Introdução

As organizações modernas estão cada vez mais orientadas por dados, e a capacidade de coletar, integrar, governar e transformar essas informações tornou-se um diferencial competitivo. No entanto, as empresas não nascem maduras em dados, elas evoluem por fases distintas, que refletem o nível de centralização, qualidade, acessibilidade e capacidade analítica de seus ambientes de dados. Essa jornada envolve não apenas infraestrutura e tecnologia, mas também governança, processos e cultura.

Ao longo das últimas décadas surgiram diversas arquiteturas:data warehouses, data lakes, lakehouses, data fabric e data mesh, cada uma tentando resolver limitações das anteriores e responder às novas demandas de escala, diversidade de dados, tempo real e autonomia do usuário final. Além disso, conceitos fundamentais como **OLTP(Processamento Transacional Online), OLAP((Processamento Analítico Online), virtualização de dados, ODS e hubs** de dados ajudam a contextualizar como diferentes camadas e sistemas interagem para suportar tanto operações do dia a dia quanto análises estratégicas.

Este material apresenta um resumo dessa evolução, começando pelos estágios de maturidade em dados das empresas, passando pela comparação entre arquiteturas modernas, e explorando componentes como governança, catálogo de metadados, MDM e processamento em tempo real. O objetivo é oferecer uma visão clara do ecossistema de dados atual, destacando como cada abordagem contribui para tornar decisões baseadas em dados mais rápidas, precisas e acessíveis.

Este resumo é baseado no Livro: Decifrando Arquiteturas de Dados: Escolhendo entre data warehouse moderno, data fabric, data lakehouse e data mesh

Estágios de Maturidade em Dados das Empresas:

Estágio 1: Reativo

No primeiro estágio, uma empresa tem dados espalhados por todos os lados, provavelmente em um monte de planilhas do Excel e/ou bancos de dados locais em muitos sistemas de arquivos diferentes, sendo enviados por e-mail para todos os lugares. Os arquitetos de dados chamam isso de **spreadmart**: uma coleção informal e descentralizada de dados, geralmente encontrada dentro de uma organização, que usa planilhas para armazenar, gerenciar e analisar dados .Indivíduos ou equipes normalmente criam e mantêm *spreadmarts* independentemente do sistema centralizado de gerenciamento de dados da organização ou de seu data warehouse oficial.

Os *spreadmarts* sofrem de:

- inconsistência de dados,
- falta de governança,
- escalabilidade limitada, e
- ineficiência (já que frequentemente resultam em muito esforço duplicado).

Estágio 2: Informativo

As empresas chegam ao segundo estágio de maturidade quando começam a centralizar seus dados, tornando a análise e a geração de relatórios muito mais fáceis. Os estágios 1 e 2 são voltados para **relatórios históricos**, ou seja, para identificar tendências e padrões do passado. Por isso, a Figura 1-3 os chama de **"retrovisor"**. Nesses estágios, você está apenas reagindo ao que já aconteceu. No estágio 2, a solução construída para coletar os dados geralmente não é muito escalável. Normalmente, o tamanho e os tipos de dados que ela pode lidar são limitados, e a ingestão de dados só ocorre de forma pouco frequente (por exemplo, a cada noite).

A maioria das empresas está no estágio 2, especialmente se sua infraestrutura ainda é on-premises.

Estágio 3: Preditivo

No estágio 3, as empresas migraram para a nuvem e construíram um sistema capaz de lidar com maiores quantidades de dados, diferentes tipos de dados e dados ingeridos com mais frequência (de hora em hora ou por *streaming*). Elas também melhoraram sua tomada de decisão ao incorporar **machine learning** (*advanced analytics*) para tomar decisões em tempo real.

Por exemplo, enquanto um usuário está em uma livraria online, o sistema pode recomendar livros adicionais na página de checkout com base nas compras anteriores do usuário.

Estágio 4: Transformador

Finalmente, no estágio 4, a empresa construiu uma solução capaz de lidar com **qualquer dado**, não importando seu tamanho, velocidade ou tipo. É fácil integrar novos dados com um tempo de preparação reduzido, porque a arquitetura pode lidar com isso e tem capacidade de infraestrutura para suportar. Essa é uma solução que permite a usuários finais **não técnicos** criarem facilmente relatórios e dashboards com as ferramentas de sua escolha. Os estágios 3 e 4 são o foco deste livro. Em particular, quando os usuários finais fazem seus próprios relatórios, essa atividade é chamada de **business intelligence de autoatendimento** (*self-service BI*), que é o tema da próxima seção.

Desta forma, o objetivo de qualquer solução de arquitetura de dados que você construa deve ser tornar rápido e fácil para qualquer usuário final, independentemente de suas habilidades técnicas, consultar os dados e criar relatórios e dashboards.

O principal objetivo de uma arquitetura de dados é permitir que uma organização gerencie e aproveite seus ativos de dados de forma eficaz, de modo a apoiar seus objetivos de negócios e processos de tomada de decisão.

Com a abordagem **schema-on-write**, o esquema é definido e aplicado no momento em que os dados são gravados ou ingeridos no banco de dados. **Os dados precisam obedecer ao esquema pré-definido** (tipos de dados, restrições, relacionamentos) antes de poderem ser armazenados.

Por contraste, na abordagem **schema-on-read, o esquema é aplicado quando os dados são lidos ou acessados, em vez de quando são gravados.** Os dados podem ser ingeridos no sistema de armazenamento sem precisar se conformar a um esquema rígido, e a estrutura só é definida quando os dados são consultados ou consumidos.

Comparação de arquiteturas de dados

Característica	Data Warehouse Relacional	Data Lake	Data Warehouse Moderno	Data Fabric	Data Lakehouse	Data Mesh
Ano de introdução	1984	2010	2011	2016	2020	2019
Centralizado/ Descentralizado	Centralizado	Centralizado	Centralizado	Centralizado	Centralizado	Descentralizado
Tipo de armazenamento	Relacional	Objeto	Relacional e objeto	Relacional e objeto	Objeto	Específico de domínio
Tipo de esquema	*Schema-on-write*	*Schema-on-read*	*Schema-on-read* e *Schema-on-write*	*Schema-on-read* e *Schema-on-write*	*Schema-on-read*	Específico de domínio
Segurança de dados	Baixa a média	Baixa	Média a alta	Alta	Média	Específico de domínio
Latência de dados	Alta	Baixa	Baixa a alta	Baixa a alta	Média a alta	Específico de domínio
Tempo para gerar valor	Médio	Baixo	Baixo	Baixo	Médio a alto	Alto
Custo total da solução	Alto	Baixo	Médio	Médio a alto	Médio a alto	Alto
Casos de uso suportados	Baixo	Baixo a médio	Médio	Médio a alto	Baixo a médio	Alto
Dificuldade de desenvolvimento	Baixa	Média a alta	Média	Média a alta	Média a alta	Alta
Maturidade da tecnologia	Alta	Média a alta	Média a alta	Média a alta	Média a alta	Baixa
Nível de habilidade necessário da empresa	Baixo	Baixo a médio	Médio a alto	Médio a alto	Médio a alto	Alto

Comparação entre sistemas OLTP e OLAP

Característica	OLTP	OLAP / Tabular
Tipo de processamento	Transacional	Analítico
Natureza dos dados	Dados operacionais	Dados consolidados
Orientação	Orientado a aplicações	Orientado a assuntos (temas)
Propósito	Executa tarefas operacionais em andamento	Apoia a tomada de decisões
Frequência de transações	Transações frequentes	Transações ocasionais
Tipo de operação	Transações online curtas: *Insert*, *Update*, *Delete*	Muitas leituras e varreduras de dados
Design dos dados	Banco de dados normalizado (3FN)	Banco de dados desnormalizado
Uso comum	Utilizado em sistemas de vendas no varejo e transações financeiras	Usado em *data mining*, vendas e marketing
Tempo de resposta	Tempo de resposta instantâneo	Tempo de resposta pode variar de segundos a horas
Complexidade das consultas	Consultas simples e instantâneas	Consultas complexas
Padrão de uso	Uso repetitivo	Uso ad hoc (sob demanda)
Natureza da transação	Transações curtas e simples	Consultas complexas
Tamanho do banco de dados	Tamanho em gigabytes	Tamanho em terabytes

Data Warehouse

As limitações dos bancos de dados relacionais levaram ao desenvolvimento dos data warehouses relacionais (RDWs), que se tornaram amplamente populares no final da década de 1980, cerca de 15 anos após o surgimento dos bancos de dados relacionais.

Um ***relational data warehouse*** **(RDW) é um tipo específico de banco de dados relacional projetado para aplicações de *data warehousing* e *business intelligence*, com desempenho de consulta otimizado e suporte para análise de dados em larga escala.**

Embora tanto os *data warehouses relacionais* quanto o processamento transacional usem o modelo relacional para organizar dados, um RDW é tipicamente maior em escala e é otimizado para consultas analíticas.

Relação entre OLAP e Data Warehouse

Os bancos de dados OLAP e os data warehouses (DWs) são conceitos relacionados, mas distintos, e geralmente são usados em conjunto.

Por meio de um modelo multidimensional ou tabular, os bancos de dados OLAP oferecem uma maneira mais flexível e interativa de analisar os dados armazenados no DW, em comparação com a execução de consultas SQL tradicionais diretamente sobre um DW que contém grandes volumes de dados.

Os modelos multidimensionais e tabulares são frequentemente considerados camadas ou modelos semânticos, o que significa que eles fornecem um nível de abstração sobre os esquemas existentes no data warehouse.

Resumo

Conceito	O que é	Função principal
Data Warehouse (DW)	Banco de dados centralizado e estruturado, que armazena dados históricos e integrados de múltiplas fontes.	Consolidar, organizar e armazenar dados (camada de armazenamento).
OLAP Database (banco OLAP)	Camada analítica construída em cima do DW, que modela os dados para análise rápida e multidimensional.	Explorar, consultar e visualizar dados de forma otimizada para decisão.

- O DW é o repositório base (armazenamento físico e histórico).
- O OLAP é a camada semântica construída sobre esse DW, que transforma os dados em estruturas analíticas (cubos ou modelos tabulares): permitindo análises interativas, rápidas e multidimensionais.

Operational Data Stores VS. DW

O propósito de um ODS é integrar dados corporativos de diferentes fontes para habilitar relatórios operacionais em tempo real (ou quase em tempo real). Ele não é um data warehouse. Como o próprio nome indica, ele armazena dados operacionais, e não dados analíticos. O ODS fornece dados muito mais rapidamente do que um DW, a cada poucos minutos, em vez de diariamente.

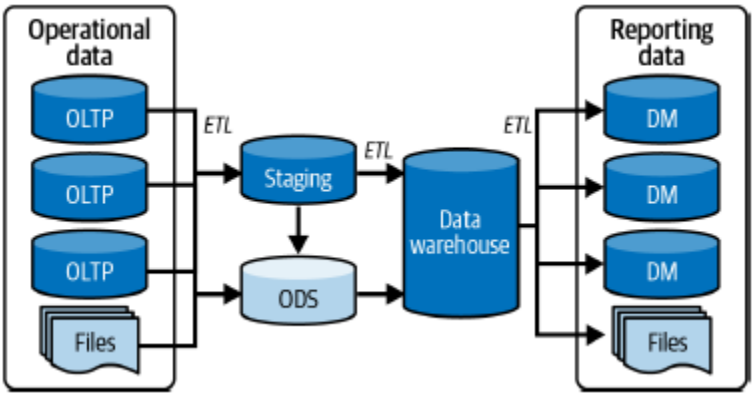
Diferenças entre um ODS e um Data Warehouse

Aspecto	Operational Data Store (ODS)	Data Warehouse (DW)
Mais adequado para	Consultas granulares e de baixo nível sobre dados operacionais detalhados	Consultas complexas sobre dados analíticos resumidos ou agregados
Propósito	Relatórios operacionais; relatórios em tempo real ou quase em tempo real	Relatórios de análise histórica e de tendências sobre grandes volumes de dados
Duração dos dados	Contém uma janela curta de dados	Contém todo o histórico dos dados da organização
Tomada de decisão	Suporta decisões operacionais e táticas baseadas em dados atuais ou quase em tempo real	Fornece feedback sobre decisões estratégicas, levando a melhorias no sistema como um todo
Frequência de carga de dados	Pode carregar dados a cada poucos minutos ou de hora em hora	Pode carregar dados diariamente, semanalmente, mensalmente ou trimestralmente

Os dados no ODS estão muito mais próximos do tempo real do que os dados exigidos pelo data warehouse. Você pode usar um ODS como uma das áreas de staging.

É recomendado que primeiro, reconcilie e limpe esses dados, preenchendo o ODS em tempo quase real para entregar valor aos tomadores de decisão operacionais e táticos que precisam dessas informações. Em seguida, você também pode utilizar o ODS para alimentar o data warehouse, que normalmente possui ciclos de carga menos exigentes.

Ainda será necessário manter uma área de staging para os dados exigidos pelo DW que não estão hospedados no ODS, mas, ao seguir essa recomendação, você reduz o número de fluxos ETL e o tamanho das áreas de staging. Isso melhora o desempenho sem sacrificar o valor nem a funcionalidade.

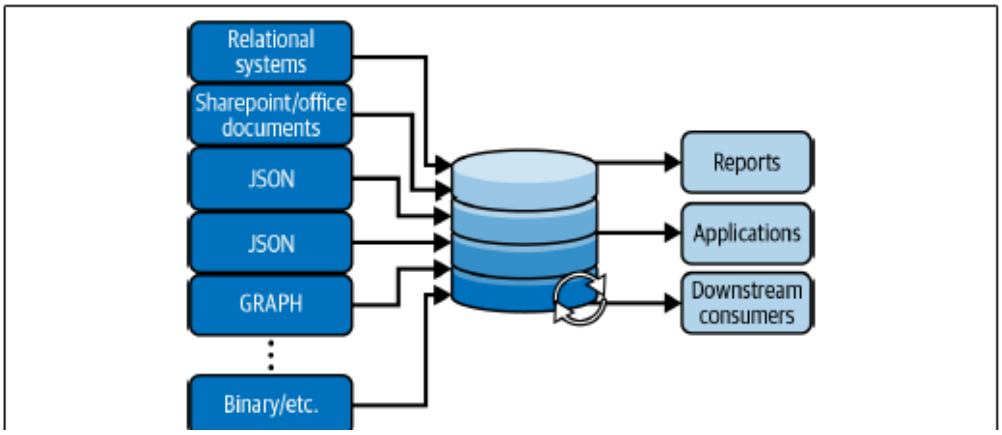


Um DW essencialmente deve conter:

- Separação em camadas lógicas:
    - *Bronze*: replicação bruta do banco de produção.
    - *Silver*: normalização, joins, limpeza, consistência.
    - *Gold*: modelagem analítica (tabelas fato/dimensão).
- Implementação de controle de histórico (SCDs:Slowly Changing Dimensions).
- Data marts ou esquemas analíticos voltados ao negócio (vendas, clientes, pedidos, etc.).
- Aplicação de modelagem dimensional (Kimball) ou Data Vault.

Data Hubs

Um hub de dados é um sistema centralizado de armazenamento e gerenciamento de dados que ajuda uma organização a coletar, integrar, armazenar, organizar e compartilhar dados de várias fontes, além de fornecer acesso facilitado a ferramentas de análise, relatórios e tomada de decisão. O aspecto mais importante é que os data hubs eliminam a necessidade de utilizar centenas ou milhares de interfaces ponto a ponto para realizar a troca de dados entre sistemas. Enquanto os data warehouses e data lakes têm como objetivo principal (e frequentemente exclusivo) fornecer dados para análise, os data hubs também são valiosos como meio de comunicação e troca de dados entre sistemas operacionais.



Um data hub armazena cópias dos dados de origem e é focado em gerenciar e distribuir esses dados, enquanto um catálogo de dados (data catalog) não armazena dados reais e tem como principal foco o gerenciamento de metadados, linhagem de dados, aplicação de políticas de governança de dados e facilitação da descoberta de dados.

Um Data Hub integra e sincroniza sistemas, por exemplo, quando um cliente atualiza seu endereço no ERP, o hub replica automaticamente essa informação para

CRM, faturamento e logística. Já um Data Warehouse consolida e armazena anos de dados limpos e estruturados para análise, permitindo calcular vendas, churn, KPIs e tendências. Enquanto o Data Hub garante consistência operacional entre sistemas, o DW fornece visão histórica e insights para tomada de decisão.

Já os data lakes normalmente usam armazenamento de objetos (object storage) para armazenar os dados, enquanto os data warehouses relacionais (RDWs) usam armazenamento relacional; já os data hubs podem combinar sistemas de armazenamento, como bancos de dados relacionais, bancos NoSQL e data lakes.

Comparação das características das soluções de dados

Característica	**Data Hub / Hub de Dados**	**Data Lake / Lago de Dados**	**Relational Data Warehouse / Data Warehouse Relacional (RDW)**	**Data Catalog / Catálogo de Dados**	**Data Marketplace / Mercado de Dados**
Foco principal	Armazenamento, integração e distribuição de dados	Armazenamento, processamento e análises de dados	Armazenamento, processamento e análises de dados	Gerenciamento de metadados e descoberta de dados	Compra, venda e compartilhamento de dados
Armazenamento de dados	Dados brutos ou levemente processados	Dados brutos e processados	Dados altamente estruturados e processados	Somente metadados	Vários formatos, dependendo do provedor
Estrutura dos dados	Estruturados, semiestruturados e não estruturados	Estruturados, semiestruturados e não estruturados	Estruturados	Somente metadados	Estruturados, semiestruturados e não estruturados
Processamento dos dados	Processamento limitado, focado em integração e distribuição	Processamento sob demanda e análises	Extenso ETL e consultas complexas	Nenhum	Dependente do provedor
Compartilhamento de dados	Integrado *(Built-in)*	Terceirizado (via ferramentas externas)	Terceirizado (via ferramentas externas)	Nenhum	Integrado *(Built-in)*
Propriedade dos dados	Dentro da organização ou de grupos relacionados	Dentro da organização ou de grupos relacionados	Dentro da organização ou de grupos relacionados	Dentro da organização ou de grupos relacionados	Múltiplas organizações
Casos de uso	Ingestão, integração e distribuição de dados	Análises de big data, machine learning e inteligência artificial	Análises complexas, relatórios e tomada de decisão	Descoberta de dados, linhagem de dados e aplicação de políticas de governança	Monetização de dados, aquisição de novos conjuntos de dados e descoberta de dados

Data Virtualization

Data Virtualization é uma camada de acesso lógico que permite consultar e combinar dados de várias fontes diferentes (bancos relacionais, APIs, arquivos, data lakes, etc.) como se estivessem todos em um só lugar, sem precisar mover ou copiar os dados fisicamente.

Analogia prática:

Pense que você tem vários armários diferentes com documentos, um na sua casa, outro no escritório e outro na casa dos seus pais. Com ETL, você copiaria todos esses documentos para um só lugar antes de começar a ler. Com data virtualization, você não copia, você cria um índice digital que sabe onde está cada documento e consegue mostrar tudo junto quando você faz uma busca.

Ou seja, ela cria uma visão unificada dos dados, mas os dados continuam morando em suas origens.

Como funciona tecnicamente

A arquitetura de data virtualization normalmente tem:

1. Conectores para as fontes de dados
   → Ex.: MySQL, Snowflake, MongoDB, APIs REST, arquivos CSV, etc.

2. Camada lógica de integração
   → Define visões (views) e modelos virtuais unificando as estruturas (ex.: juntar tabelas de clientes de diferentes bancos).

3. Camada de segurança e governança
   → Controla quem pode acessar quais dados.

4. Mecanismo de otimização de consultas
   → O query engine divide a consulta em partes, envia para cada fonte e combina os resultados (federated query).

5. Interface única (geralmente SQL)
   → O usuário (ou dashboard, ou API) consulta tudo via um endpoint SQL único.

Quando usar Data Virtualization

Use quando:

- você precisa integrar múltiplas fontes rapidamente, sem construir um ETL;
- as consultas são leves ou intermediárias (não intensivas);
- precisa de dados sempre atualizados em tempo real;
- quer agilizar prototipagem e exploração de dados antes de modelar um DW definitivo;
- quer criar uma camada única de acesso governado (por exemplo, para APIs ou dashboards).

Evite quando:

- há grande volume de joins ou agregações pesadas;
  o objetivo é histórico / analítico (melhor DW ou lakehouse);
- Você precisa armazenar snapshots ou versionamento de dados.

Motivos pelos quais você pode escolher a virtualização de dados:

- Existem restrições regulatórias sobre movimentação de dados
    - As regulamentações de soberania de dados da China determinam que os dados localizados na China devem permanecer na China e só podem ser consultados por uma pessoa localizada dentro do país.Mas imagine que você esteja nos Estados Unidos e tente consultar dados de clientes na China a partir do seu aplicativo.O aplicativo pode recuperar metadados sobre os dados de clientes na China, mas não pode ingerir (baixar) esses dados.Você poderia usar software de virtualização de dados para manter os dados dentro do país, permitindo que somente uma pessoa localizada na China possa consultá-los. Qualquer pessoa fora da China que tente acessar esses dados receberia uma mensagem de erro ou não veria nenhum resultado.
- Há dados de referência em um banco fora do DW que mudam com frequência:
    - Em vez de copiar esses dados repetidamente para dentro do data warehouse, você pode usar virtualização de dados para consultar diretamente os dados na origem sempre que precisar fazer *join* com os dados do DW.
- Você quer construir uma solução rapidamente para mover dados de origem:
    - Em vez de usar uma ferramenta ETL tradicional para copiar dados de origem para um local central e depois criar relatórios a partir disso, você pode criar relatórios diretamente usando o software de virtualização. Ele executará consultas sobre os dados de origem e fará a centralização automaticamente.

Isso significa que você pode construir uma solução muito mais rápido, já que não precisa usar uma ferramenta ETL, que pode ser muito demorada.

MDW (Modern Data Warehouse) Vs. Lakehouse

Um **Modern Data Warehouse (MDW)** é uma arquitetura que combina um data lake para ingestão e processamento flexível de dados com um data warehouse relacional para análises estruturadas. Ele difere de um DW convencional porque não depende apenas de um banco relacional rígido: no MDW, parte dos dados fica no lake (arquivos, semiestruturados, streaming) e só o que precisa ser analisado é replicado para o DW, permitindo maior escala, variedade de dados e integração com machine learning.

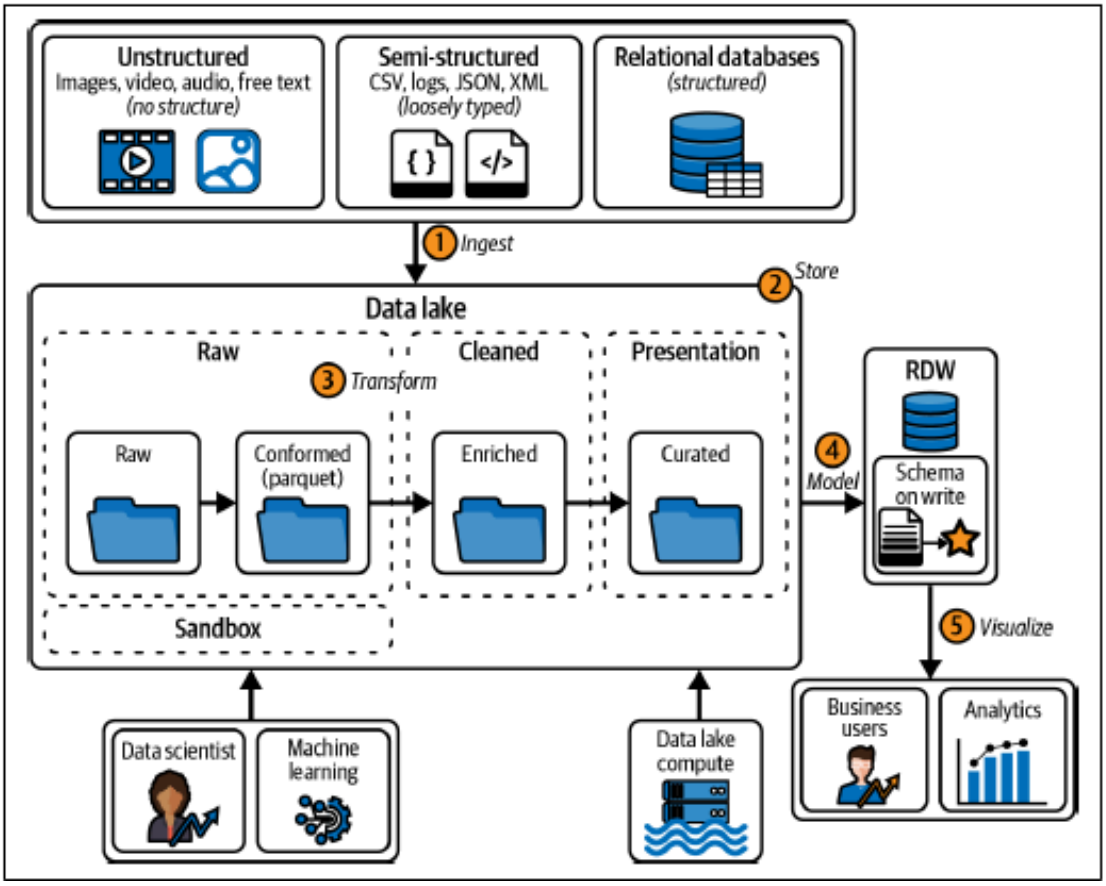
Já a principal diferença entre a arquitetura **MDW (Modern Data Warehouse**) e a arquitetura **Lakehouse** está no fato de que, no MDW, parte dos dados armazenados e processados no data lake é replicada para um data warehouse relacional (RDW), enquanto no Lakehouse essa duplicação não existe.

No MDW, o data lake funciona como a camada de ingestão e transformação inicial, armazenando dados brutos e semiestruturados de maneira barata e escalável, e permitindo operações de big data, machine learning e análises exploratórias. Entretanto, para atender à necessidade de relatórios corporativos, BI tradicional e consultas altamente otimizadas, os dados são copiados para um warehouse relacional, onde são organizados em esquemas mais rígidos e otimizados para performance, qualidade e governança. É exatamente essa replicação do lake → warehouse que caracteriza o MDW: duas plataformas diferentes cumprindo papéis distintos, cada uma com sua especialização. Tecnologias típicas dessa arquitetura incluem Amazon S3, Azure Data Lake Storage ou Google Cloud Storage como data lake, e Snowflake, BigQuery, Redshift, Azure Synapse SQL ou Teradata como o warehouse relacional para BI.

Por outro lado, na arquitetura Lakehouse, não existe a necessidade de copiar dados para um segundo sistema. Todo o armazenamento e processamento ocorre diretamente no data lake, mas com tecnologias que adicionam funcionalidades tradicionalmente encontradas em data warehouses, como transações ACID, índices, caching avançado, otimização de arquivos e catálogos unificados. Assim, o lakehouse combina a flexibilidade e o baixo custo de um data lake com a confiabilidade e a performance de um warehouse, evitando redundância de dados e simplificando a arquitetura. Nesse modelo, o próprio data lake fornece governança, performance e consistência suficientes para BI e análises corporativas. Exemplos típicos de tecnologias de Lakehouse incluem Databricks Delta Lake, Apache Iceberg, Apache Hudi, Snowflake com Iceberg Tables, BigLake no Google Cloud e Unity Catalog como camada de governança.

Em resumo, o MDW usa dois sistemas separados, com dados replicados entre eles, enquanto o Lakehouse usa um sistema só, com o data lake desempenhando o papel de armazenamento analítico completo. Esta diferença diferencia fortemente as arquiteturas e determina a escolha de tecnologias e padrões de operação em cada uma.

MDW:



Obs.: Num MDW, nem todos os dados da fonte precisam ser copiados para o data lake. Em alguns casos, especialmente novos projetos construídos na nuvem, é mais rápido copiar dados diretamente da fonte para o RDW, ignorando o data lake, especialmente para dados estruturados (relacionais). Porém, dados de origem que ignoram o data lake não recebem alguns de seus benefícios: especialmente backup, útil caso seja necessário reexecutar um ETL. Além disso, isso pode sobrecarregar o RDW, que precisará fazer a limpeza dos dados. E para quem usa o data lake, esses dados ficarão ausentes, impedindo o lake de ser a *single source of truth*

Data Fabric

A arquitetura de ***data fabric*** é uma evolução da arquitetura de data warehouse moderno (MDW): uma camada avançada construída sobre o MDW para aprimorar acessibilidade, segurança, capacidade de descoberta e disponibilidade dos dados. O aspecto mais importante da filosofia de *data fabric* é que uma solução de *data fabric* pode consumir qualquer e todos os dados dentro da organização.

O *data fabric* é a arquitetura mais amplamente utilizada para novas soluções construídas na nuvem, especialmente para grandes volumes de dados (acima de 10 terabytes). **Muitas empresas estão "atualizando" sua arquitetura MDW para um *data fabric* ao adicionar tecnologias adicionais,** especialmente processamento em tempo real, políticas de acesso a dados e um catálogo de metadados.

A arquitetura de data fabric contém todos os componentes do MDW:

1. ***ingestão***
2. ***armazenamento***
3. ***transformação***
4. ***modelagem***
5. ***visualização***

Mas apresenta recursos adicionais, pode se considerar um Data Fabric se pelo menos 3 desses recursos estão presentes:

6. **Políticas de Acesso a Dados (Data Access Policies):** As políticas de acesso a dados são fundamentais para a governança de dados. Elas consistem em um conjunto de diretrizes, regras e procedimentos que controlam quem tem acesso a quais informações, como essas informações podem ser usadas e quando o acesso pode ser concedido ou negado dentro de uma organização.
7. **Catálogo de Metadados (*Metadata Catalog*):** Os *data fabrics* incluem um catálogo de metadados: um repositório que armazena informações sobre ativos de dados, incluindo sua estrutura, relacionamentos e características. Ele fornece uma forma centralizada e organizada de gerenciar e descobrir dados, tornando mais fácil para os usuários encontrar e compreender os dados de que precisam.
8. **MDM(Master Data Management (MDM):** é o processo que garante que dados essenciais da empresa, como clientes, produtos e fornecedores, existam em uma versão única, correta e consistente. Ele elimina duplicidades e conflitos entre sistemas, criando um *Golden Record* confiável usado por toda a organização. Assim, decisões, análises e integrações passam a operar sobre dados padronizados e de alta qualidade.
9. **Virtualização de Dados (*Data Virtualization*):** virtualização de dados é uma arquitetura de software que permite que aplicações e usuários finais acessem dados de múltiplas fontes como se estivessem armazenados em um único local.

10. **Processamento em Tempo Real (*Real-Time Processing*)**: refere-se ao processamento de dados e produção de resultados imediatos assim que os dados se tornam disponíveis, sem atraso perceptível. Isso permite tomar decisões com base em informações atualizadas (por exemplo, informações de tráfego enquanto se está dirigindo).
11. **APIs:** Em vez de depender de *connection strings*, APIs fornecem dados de maneira padronizada a partir de várias fontes, como um data lake ou o RDW, sem expor detalhes sobre onde os dados estão fisicamente armazenados.Se o dado for movido, basta atualizar o código interno da API; nenhuma das aplicações que consomem a API é afetada. Isso torna mais fácil aproveitar novas tecnologias.
12. **Serviços:** O *data fabric* pode ser construído em "blocos", o que torna mais fácil reutilizar código. Por exemplo, pode haver outras equipes dentro da empresa que não precisam do *data fabric* completo, mas precisam do código que você criou para limpar dados ou ingeri-los.
13. **Produtos:** Um *data fabric* inteiro pode ser empacotado como um produto e vendido. Isso seria particularmente atraente se fosse feito especificamente para um setor, como o de saúde.

Por que Fazer a Transição de um MDW para uma Arquitetura de Data Fabric?

A transição de um MDW para uma arquitetura de *data fabric* ocorre porque o MDW é mais rígido e difícil de escalar, enquanto o *data fabric* é flexível, suporta múltiplos tipos de dados e oferece uma visão unificada das fontes. **Ele possibilita processamento em tempo real, melhora segurança e governança, e facilita conformidade regulatória, especialmente em empresas multinacionais**. Também é ideal para setores que exigem respostas imediatas, como e-commerce e mercados financeiros.

Como desvantagens, a migração pode ser complexa, cara e exigir treinamento e integração extensiva. Pequenas empresas talvez não precisem dessas capacidades avançadas, e a complexidade do *data fabric* pode dificultar troubleshooting. Cada organização deve avaliar custo, maturidade técnica e retorno antes de adotar essa arquitetura.

Exemplo de Uso: Numa empresa global com dados espalhados pela Europa, China, Estados Unidos e Brasil, um MDW exige copiar dados entre regiões, criar ETLs complexos e gerenciar permissões separadas. Já o Data Fabric cria uma camada inteligente que sabe onde cada dado está, aplica automaticamente políticas como GDPR (Europa) e LGPD (Brasil), permite consultas sem mover dados da China, entrega tudo em tempo real e expõe informações via APIs. Assim, reduz integrações manuais, automatiza governança e facilita o acesso aos dados em toda a empresa.

Diferença MDW e Data Fabric:

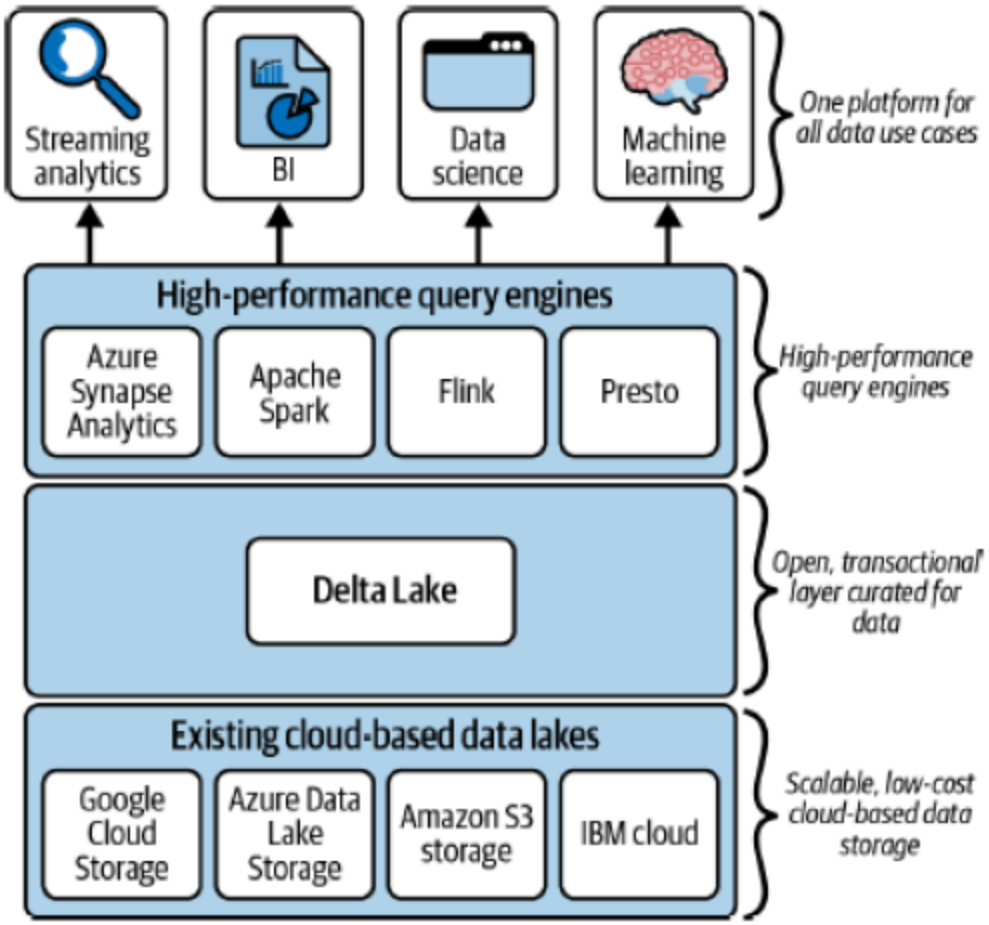
O Modern Data Warehouse (MDW) é uma arquitetura centrada em mover e consolidar dados em um único repositório, onde eles são transformados e organizados para análises estruturadas. **Já o Data Fabric cria uma camada inteligente acima das fontes, capaz de conectar, governar e disponibilizar dados distribuídos sem exigir sua centralização física.** Enquanto o MDW depende de ETL/ELT e armazenamento central, o Data Fabric utiliza metadados ativos, automação, virtualização e APIs para unificar acesso, segurança e descoberta de dados em todo o ecossistema. Em essência, o MDW armazena dados para análise; o Data Fabric integra, orquestra e governa dados de toda a organização, independentemente de onde estejam.

Data Lakehouse

A ideia por trás de um *data lakehouse* é simplificar as coisas usando apenas um *data lake* para armazenar todos os seus dados, em vez de também ter um *data warehouse* relacional separado. Para isso, o *data lake* precisa de mais funcionalidades para substituir os recursos de um RDW. É aí que entra, por exemplo, o **Delta Lake da Databricks.**

Delta Lake é uma camada de software de armazenamento transacional que roda sobre um *data lake* existente e adiciona recursos semelhantes aos de um RDW, melhorando a confiabilidade, segurança e desempenho do lake. Ele incorpora os controles, a gestão de dados e as estruturas de dados encontradas em um data warehouse, enquanto ainda armazena dados em armazenamento de objetos e suporta uma variedade de mecanismos de consulta e transformação.

Delta Lake não é a única opção para fornecer funcionalidades adicionais a um *data lake*; duas outras escolhas populares são Apache Iceberg e Apache Hudi, que possuem recursos muito semelhantes. Esses frameworks atuam como o "motor" que transforma um data lake simples em um lakehouse robusto e funcional. Sem eles, a arquitetura lakehouse perde muitos de seus benefícios e se aproxima mais de um data lake tradicional



Como em uma arquitetura *data lakehouse*, existe apenas um repositório para os seus dados (o *data lake*), em vez de dois (um data lake e um RDW). Isso resolve seis problemas comumente vistos nas arquiteturas MDW e *data fabric*:

1. **Confiabilidade:** Manter um *data lake* e um RDW consistentes pode ser um problema, especialmente se grandes quantidades de dados precisam ser copiadas frequentemente do data lake para o RDW. Se os jobs de cópia falharem, não copiarem os dados com precisão ou colocarem os dados no local errado no data lake, isso pode causar problemas de confiabilidade. Por exemplo, executar relatórios no RDW pode retornar resultados diferentes dos obtidos ao consultar os mesmos dados no data lake. Com um *data lakehouse*, como não existe um RDW para o qual copiar dados, isso não ocorre.
2. **Desatualização dos dados:** Os dados em um RDW serão mais antigos do que os dados equivalentes no data lake. O quanto mais antigos dependerá da frequência com que os dados são copiados do lake para o RDW e, para evitar afetar o desempenho de consultas e relatórios, você não quer executar esses jobs com muita frequência. Assim como nos problemas de confiabilidade acima, isso pode resultar em relatórios do RDW

e do data lake retornando resultados diferentes. No *lakehouse*, como não existe RDW, isso não é um problema.

3. **Suporte limitado a análises avançadas:** Poucos sistemas e ferramentas de IA/ML funcionam bem em RDWs, porque cientistas de dados normalmente preferem trabalhar com arquivos em um data lake. Eles obtêm isso com uma arquitetura *data lakehouse*.
4. **Custo total de propriedade:** Embora o armazenamento seja relativamente barato, há custos adicionais para o processamento necessário para copiar dados para um RDW. Além disso, isso resulta em duas cópias dos dados, aumentando custos de armazenamento. Por contraste, em um lakehouse há apenas uma cópia dos dados. Além disso, o processamento usado para consultas e relatórios no RDW geralmente custa muito mais do que o processamento usado em um data lake. Administrar um data lake e um RDW também envolve habilidades diferentes, e contratar para ambas as áreas pode significar custos extras.
5. **Governança de dados:** Ter duas cópias dos dados em dois ambientes de armazenamento diferentes, possivelmente com dois tipos de segurança distintos, aumenta o risco de alguém visualizar dados que não deveria ver. Também é difícil garantir que ambos os sistemas sigam as mesmas regras de qualidade e transformação dos dados. Com um lakehouse e sua única cópia de dados, esses problemas não existem.
6. **Complexidade:** Administrar tanto um data lake quanto um RDW pode ser complexo, exigindo habilidades e recursos especializados. Como um lakehouse não tem RDW, menos habilidades especializadas são necessárias.

Lakehouse trade-offs

- Um *data lakehouse* sem um RDW (Relational Data Warehouse) traz perdas de desempenho porque consultas SQL não contam com otimizações típicas de bancos relacionacionais MPP (Massively Parallel Processing), como índices avançados, estatísticas, *materialized views* e planos de execução otimizados.

- O Delta Lake depende de Spark SQL, exigindo que usuários deixem o SQL ANSI (American National Standards Institute) tradicional e aprendam o modelo de execução distribuída do Spark.

- Também faltam recursos maduros de governança e segurança, como row-level security (segurança por linha), column-level security (segurança por coluna), TDE (Transparent Data Encryption), *data masking* e auditoria forte.

- Além disso, o modelo baseado em arquivos torna os metadados instáveis e suscetíveis a inconsistências quando alterados fora do Delta.

- Como o Delta Lake funciona em *schema-on-read* e armazena tudo como arquivos e pastas, ele não possui contexto relacional nem metadados fixos como um RDW. Cada arquivo é isolado, sem relações pré-definidas, por isso é necessário criar uma Relational Serving Layer, uma camada que recria a visão

relacional usando *SQL views*, tabelas Hive, datasets de BI ou consultas SQL que definem *joins* e relacionamentos. Essa camada faz com que o usuário final consulte o Lakehouse “como se fosse um RDW”, mesmo que por baixo sejam apenas arquivos Delta. Porém, ela pode gerar inconsistências, porque metadados no Delta Lake não são garantidos, podem ficar diferentes entre camadas ou até desatualizados se alguém alterar os arquivos diretamen

Obs.: Estruture seu MDW (ou data lakehouse) de acordo com as necessidades do negócio. Não simplesmente espelhe, de forma automática, a estrutura dos dados de origem.

Data Mesh

Um *data mesh* é uma arquitetura de dados descentralizada com quatro características específicas.

- Primeiro, exige que equipes independentes dentro de domínios designados sejam responsáveis pelos seus dados analíticos.
- Segundo, em um data mesh, os dados são tratados e oferecidos como um produto, ajudando o consumidor de dados a descobrir, confiar e utilizá-los para qualquer finalidade que desejar.
- Terceiro, ele depende de provisionamento automatizado de infraestrutura.
- E quarto, utiliza governança para garantir que todos os produtos de dados independentes sejam seguros e sigam regras globais.

É muito importante entender que *data mesh é um conceito, não uma tecnologia*. Trata-se de uma mudança organizacional e cultural dentro das empresas. A tecnologia usada para construir um data mesh pode seguir a arquitetura de data warehouse moderno, data fabric ou data lakehouse, ou os domínios podem até seguir arquiteturas diferentes.

Uma grande diferença entre o data mesh e outras arquiteturas já mencionadas é que o data mesh é *descentralizado*. Isso significa que os dados, juntamente com tudo o que é usado para gerenciá-los, são de propriedade e administrados por equipes individuais ou “domínios” dentro de uma organização, agrupados por domínio de negócio, em vez de uma autoridade central ou uma única equipe de TI.

Essas equipes de domínio são responsáveis por coletar, processar e gerenciar seus próprios dados, e têm autonomia para tomar decisões sobre como esses dados são usados e compartilhados. Os dados permanecem dentro do domínio; usuários finais podem acessá-los e consultá-los *onde eles residem*, sem copiá-los para um local centralizado. Isso gera maior responsabilização e garante que os dados sejam administrados e mantidos por aqueles com maior expertise sobre eles.

Isso resolve o principal problema das arquiteturas centralizadas, que é o escalonamento organizacional e técnico. À medida que uma empresa cresce e mais dados precisam ser centralizados rapidamente, o time de TI costuma ficar sobrecarregado e se torna um gargalo.

Arquiteturas centralizadas não abrangem apenas uma equipe central de TI que possui e controla os dados, mas essa equipe também é responsável por:

- Integrar dados de diferentes fontes
- Armazenar dados em um local centralizado que ela controla
- Construir modelos de dados e mantê-los em um repositório central
- Governança e conformidade, incluindo definição e aplicação de padrões de qualidade, segurança e privacidade
- Criar e manter pipelines de dados e lógica de transformação
- Aplicar qualidade de dados via *profiling*, validação, limpeza e outros processos
- Otimizar desempenho
- Gerenciar toda a tecnologia de computação
- Administrar ferramentas de análise e relatórios
- Armazenar e gerenciar metadados
- Recuperação de desastres e backup

Além disso, em uma arquitetura centralizada, o hardware e o poder de processamento são projetados para escalar verticalmente.

O Hype do Data Mesh:

Muitos que promovem o data mesh afirmam que projetos de data warehouse falham em taxas muito altas, são insustentáveis e não conseguem lidar com big data porque não escalam. Porém, embora esses projetos realmente possam falhar, isso raramente ocorre por limitações de escala ou incapacidade tecnológica; na maioria das vezes, os problemas estão ligados a pessoas, processos ou à escolha inadequada de tecnologia.

Nesse contexto, o data mesh ganhou hype ao se apresentar como a solução para esses gargalos, propondo uma arquitetura descentralizada e baseada na responsabilidade por domínio. No entanto, muitos especialistas argumentam que ele é uma solução procurando um problema, já que arquiteturas centralizadas modernas já escalam muito bem. Além disso, a complexidade organizacional necessária para implementar um data mesh “puro” faz com que poucas empresas realmente consigam adotá-lo, e grande parte do que hoje é chamado de data mesh é, na prática, apenas data fabric ou lakehouse rebatizados com buzzwords.

4 Princípios do Data Mesh:

Domain Ownership (Propriedade por Domínio): Os dados deixam de ser responsabilidade exclusiva de um time central de TI e passam a ser gerenciados pelos próprios domínios de

negócio que os geram. Isso aumenta a responsabilidade, melhora a qualidade e elimina o gargalo do time central. Cada domínio entende seu contexto e, portanto, toma melhores decisões sobre seus dados.

Data as a Product (Dados como Produto): Cada domínio deve tratar seus dados analíticos como um produto completo: fácil de descobrir, acessível, confiável, seguro e bem documentado. Os consumidores desses dados são tratados como clientes, o que faz com que o domínio se preocupe com qualidade, SLAs, usabilidade e valor entregue.

Self-Serve Data Infrastructure (Infraestrutura Self-Service): Para que domínios consigam operar seus data products sem depender de TI, uma plataforma central fornece infraestrutura automatizada: provisionamento de armazenamento, computação, pipelines, logging e controle de acesso. Isso reduz a complexidade e evita que cada domínio "reinvente a roda".

Federated Computational Governance (Governança Federada): A governança não é totalmente centralizada, nem totalmente descentralizada. A equipe central define padrões globais (qualidade, segurança, interoperabilidade), enquanto os domínios implementam e monitoram essas regras localmente. Assim busca-se equilíbrio entre autonomia e consistência.

Muitas empresas dizem ter um data mesh ao implementar apenas alguns princípios, ao separar dados por departamentos ou ao simplesmente conectar seus data lakes e data warehouses por meio de virtualização. Porém, isso não é um data mesh real, já que o conceito exige uma mudança organizacional e cultural, não apenas tecnológica. Além disso, existem inúmeras exceções possíveis a cada princípio, o que torna difícil determinar quando uma solução ainda pode ser chamada de data mesh. Sem consenso sobre os quatro princípios como um conjunto obrigatório, nem é possível definir um *minimum viable mesh*.

Data Mesh versus Data Fabric

Data Mesh é um conceito organizacional que descentraliza a responsabilidade pelos dados, fazendo com que cada domínio os trate como produtos e cuide de sua qualidade, governança e compartilhamento. Já o Data Fabric é uma arquitetura tecnológica que automatiza integração, metadados e acesso aos dados, servindo como infraestrutura que pode ser usada dentro de um Mesh. Ou seja, o Mesh define *como a empresa se organiza*, enquanto o Fabric define *como a tecnologia conecta e automatiza os dados*. Eles não competem, são complementares, e um Data Mesh pode conter vários Data Fabrics, um para cada domínio.

Uma reflexão essencial é distinguir quando estamos apenas realizando *data federation*, isto é, integrando dados provenientes de múltiplas fontes sem centralizá-los e quando de fato estamos buscando implementar um data mesh. A federation, embora útil, cumpre somente a parte do princípio de **propriedade de dados por domínio**, mas não

incorpora os demais pilares que caracterizam um mesh completo. Por isso, soluções baseadas apenas nessa integração distribuída são mais bem descritas como um enterprise data lakehouse ou como uma arquitetura data fabric com workspaces independentes por domínio. Mesmo que cada domínio mantenha seu próprio ambiente, essa abordagem não atende plenamente aos requisitos de um data mesh verdadeiro.

Resumo Arquiteturas de Dados e Suas Aplicações

Modern data warehouse: Os MDWs são mais adequados para empresas que lidam com uma pequena quantidade de dados e já estão acostumadas com data warehouses relacionais, tornando a transição suave.

Data fabric: À medida que as empresas crescem e se diversificam, muitas vezes se veem lidando com dados provenientes de inúmeras fontes, cada uma diferindo em tamanho, velocidade e tipo. O data fabric é a escolha arquitetural para quem precisa de uma forma contínua e integrada de unir esses fluxos de dados tão diversos.

Data lakehouse: Pode ser visualizado como uma solução intermediária. Você deve utilizar um data lakehouse como sua solução principal até encontrar suas limitações. Quando isso acontecer, você pode transferir conjuntos de dados específicos para um RDW (Relational Data Warehouse) para melhor atender às necessidades da sua organização.

Data mesh: Projetado para empresas muito grandes e orientadas a domínios, o data mesh é destinado a organizações que estão enfrentando desafios significativos de escalabilidade e possuem os recursos e o tempo para investir em uma solução robusta.

Essas arquiteturas não são mutuamente exclusivas. O mundo real raramente trabalha com absolutos. A maioria das organizações descobrirá que suas necessidades únicas exigem um mosaico de recursos oriundos de múltiplas arquiteturas. Uma solução personalizada pode combinar aspectos de qualquer uma ou de todas essas arquiteturas, adaptadas aos casos de uso específicos de dados da organização e às capacidades de negócio em evolução.