Arquitetura de dados

Prof. M.Sc Weslley Moura

Prazer!

https://www.linkedin.com/in/weslleymoura/
weslleymoura@gmail.com

Lisboa, Portugal
Antes: Dublin, Irlanda - Osasco, São Paulo

Consensys, EUA
Head of Data

Mestre em Engenharia de Software, IPT
Doutorando Universidade do Minho, Portugal
Pós graduação em data science, Atlântica University, Lisbon
MBA in Business Intelligence, FIAP, São Paulo
Bacharelado em sistemas de informação, FIAP, São Paulo

Olá pessoal! Trabalho na área de tecnologia há 20 anos, exercendo funções como engenheiro de software e cientista de dados. Desde 2016, venho atuando em cargos de liderança na área de dados.

Empresas: Citibank, Itaú, Banco BV, NET, Hospital Albert Einstein, Loggi, Consensys

O que venho fazendo de bom?

Machine Learning

Objetivos da disciplina

Neste curso, você irá explorar a evolução das arquiteturas de dados e entender como as práticas modernas, como o padrão ELT e o desacoplamento entre armazenamento e processamento, tornam as plataformas mais escaláveis e eficientes.

Vamos aprofundar o conceito das camadas de armazenamento (Bronze, Silver e Gold) e discutir estratégias de modelagem para garantir dados organizados e de fácil acesso. Além disso, abordaremos a orquestração de tarefas, garantindo pipelines de dados automatizados e confiáveis.

A disciplina também destaca a importância da governança na plataforma de dados, assegurando segurança, qualidade e conformidade. Para consolidar o aprendizado, os alunos terão acesso a um pipeline de dados que utiliza as principais ferramentas de mercado.

Ao final do curso, você terá um entendimento sólido dos fundamentos e práticas essenciais da engenharia de dados, pronto para enfrentar desafios reais na área! 🚀

Conteúdo programático

Evolução das arquiteturas de dados
Dos monolíticos aos lakehouses: descubra como as arquiteturas de dados evoluíram para atender às demandas do mundo moderno.

Desacoplamento do armazenamento e processamento de dados
Escalabilidade e flexibilidade! Saiba como separar o armazenamento do processamento pode revolucionar sua arquitetura.

Camadas de armazenamento de dados
Bronze, Prata e Ouro: entenda como organizar dados de forma estratégica para confiabilidade e performance.

Padrão Extract, Load and Transform (ELT)
Mude o jogo do processamento de dados! Entenda por que o ELT se tornou a abordagem preferida em grandes volumes de dados.

Estratégias para modelagem de dados
Modelagem importa! Descubra como estruturar seus dados para atender às necessidades analíticas e operacionais.

Orquestração de tarefas
Automação inteligente! Aprenda a coordenar pipelines de dados de forma eficiente e escalável.

Governança da plataforma de dados
Segurança, conformidade e qualidade! Conheça as melhores práticas para manter um ambiente de dados confiável.

Metodologia aplicada

Aulas expositivas e práticas
Os conceitos apresentados serão demonstrados em um ambiente realista, utilizando ferramentas como Apache Spark, Delta Lake, Airflow e S3.

Avaliação de desempenho em formato de certificação em engenharia de dados
Questões em formato múltipla escolha (nota mínima 70%). Conteúdo da avaliação: todo material apresentado em aulas + artigos sugeridos para leitura

Evolução das arquiteturas de dados

Arquiteturas Monolíticas

Bancos de dados centralizados e acoplados às aplicações

Baixa escalabilidade e dificuldade na integração de dados

Data Warehouses

Foco em relatórios e BI

Estrutura rígida, alto custo de manutenção

Data Lakes

Armazenamento escalável e flexível

Desafios com governança e qualidade dos dados

Lakehouse

Combina a flexibilidade do Data Lake com a confiabilidade do Data Warehouse

Formato aberto, processamento otimizado e governança aprimorada

Evolução das arquiteturas de dados

Arquiteturas Monolíticas

Bancos de dados centralizados e acoplados às aplicações

Baixa escalabilidade e dificuldade na integração de dados

O que mais?

Difícil escalar com o crescimento dos dados
Alta dependência entre armazenamento e processamento
Custos elevados para manter desempenho

Evolução das arquiteturas de dados

Data Warehouses

Foco em relatórios e BI

Estrutura rígida, alto custo de manutenção

O que mais?

Modelagem complexa (ex.: Star Schema, Snowflake)
Processamento caro e muitas vezes lento
Dificuldade em lidar com dados não estruturados

Evolução das arquiteturas de dados

Data Lakes

Armazenamento escalável e flexível

Desafios com governança e qualidade dos dados

O que mais?

Falta de governança pode transformar o repositório em um "pântano de dados"
Dificuldade em garantir qualidade e consistência dos dados

Evolução das arquiteturas de dados

Lakehouse

Combina a flexibilidade do Data Lake com a confiabilidade do Data Warehouse

Formato aberto, processamento otimizado e governança aprimorada

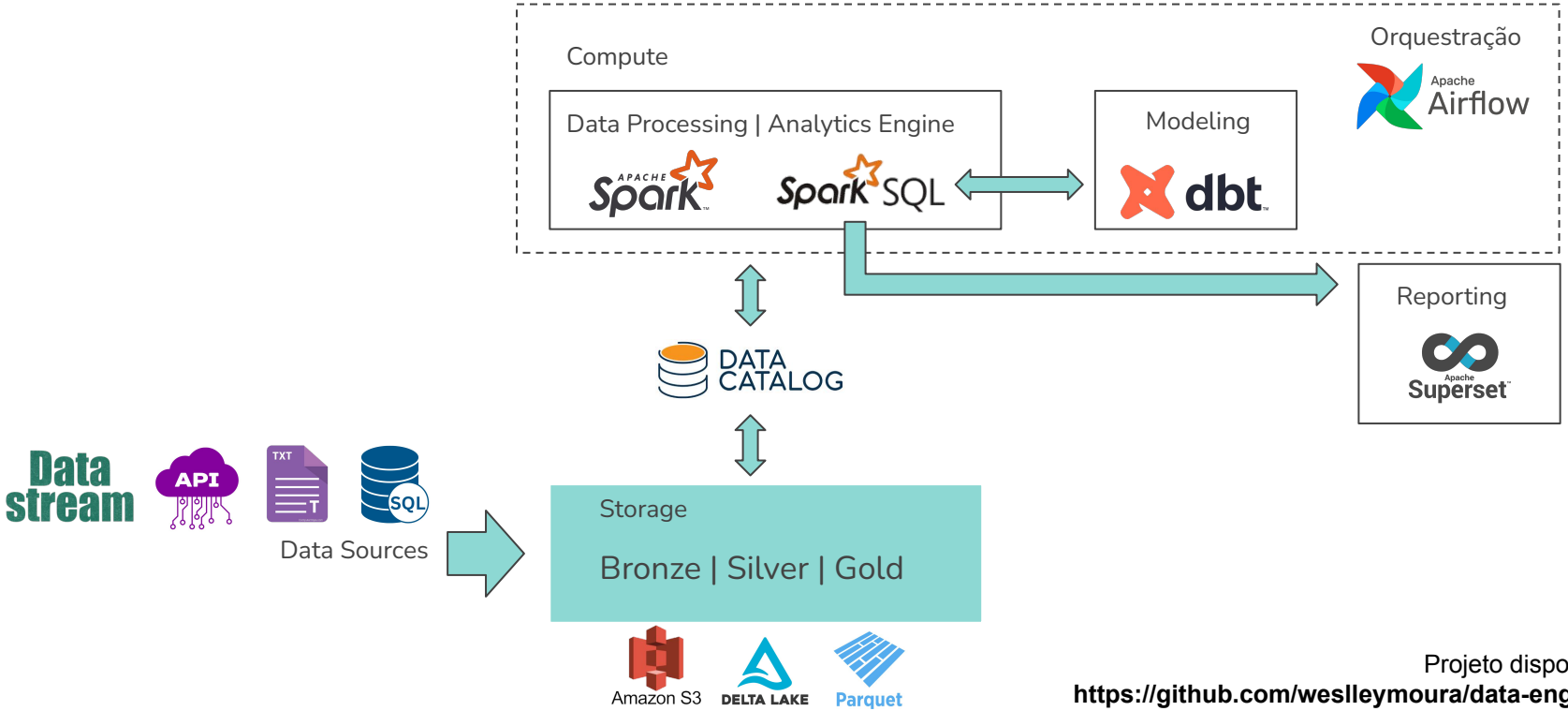
O que mais?

🏆 **Lakehouse: A Evolução das Arquiteturas de Dados**

Uma arquitetura que une a escalabilidade dos **Data Lakes** com a governança e confiabilidade dos **Data Warehouses**.

✅ **Formato Aberto** – Suporte a dados estruturados e não estruturados
✅ **Armazenamento Desacoplado** – Dados ficam em camadas independentes
✅ **Transações ACID** – Garantia de consistência e confiabilidade dos dados
✅ **Performance Otimizada** – Indexação, caching e otimizações nativas
✅ **Governança Aprimorada** – Controle de acesso, versionamento e qualidade dos dados

Exemplo de uma arquitetura de dados moderna (nosso projeto)



Projeto disponível em:
https://github.com/weslleymoura/data-engineering

Exemplo de uma arquitetura de dados moderna (nosso projeto)

LAKEHOUSE

Sugestão de leitura:
https://atlan.com/know/iceberg/apache-iceberg-vs-delta-lake/

Sugestão de leitura:
https://atlan.com/know/iceberg/apache-iceberg-vs-delta-lake/

Sugestão de leitura:
https://www.onehouse.ai/blog/comprehensive-data-catalog-comparison
https://lakefs.io/blog/hive-metastore-vendor-lock/

Desacoplamento do Armazenamento e Processamento

Separar **onde os dados são armazenados** de **como eles são processados**, garantindo mais flexibilidade, escalabilidade e eficiência.

🔹 **Antes: Arquiteturas Acopladas**
❌ Bancos de dados tradicionais integravam armazenamento e processamento
❌ Dificuldade de escalar de forma independente
❌ Custos elevados para upgrades de hardware

🔹 **Agora: Arquiteturas Desacopladas**
✅ Dados armazenados em **camadas abertas** (ex.: S3, ADLS, GCS)
✅ Processamento independente com **Spark, Trino, Dremio**
✅ **Escalabilidade elástica** – Processamento cresce conforme a demanda

Com esse modelo, conseguimos **armazenar grandes volumes de dados de forma barata** e **processá-los sob demanda**, sem necessidade de infraestrutura monolítica.

Camadas de armazenamento de dados

Bronze
Dados Brutos

Dados não tratados ou parcialmente tratados.

Armazenamento de dados na forma original, com pouca ou nenhuma transformação.

Ex: Logs, arquivos de eventos, dados de sensores.

Preservar os dados em seu estado mais puro para futuras análises e processamento

Silver
Dados Limpos e Curados

Dados transformados e limpos para análise.

Dados integrados, validados e organizados em formatos estruturados (tabelas, CSV, Parquet).

Ex: Tabelas de transações, dados de clientes filtrados e agregados.

Organizar os dados para facilitar consultas analíticas e integrar fontes diferentes

Gold
Dados Preparados para Negócio

Dados totalmente transformados, prontos para decisões estratégicas.

Dados enriquecidos, agregados e modelados para fácil acesso e análise.

Ex: Relatórios analíticos, dashboards de KPI.

Prover dados confiáveis e de alta qualidade para decisões empresariais

Padrão Extract, Load and Transform (ELT)

ELT é um processo onde **os dados são extraídos (Extract)**, **carregados (Load)** no repositório de dados e, só então, **transformados (Transform)** para atender às necessidades analíticas.

Diferente de ETL!

- **ETL (Extract, Transform, Load)**: Os dados são transformados antes de serem carregados no repositório.
- **ELT (Extract, Load, Transform)**: Os dados são carregados primeiro, e as transformações são feitas **no repositório**, usando a capacidade de processamento da plataforma.

✅ **Menor Latência**: Carregar dados primeiro e transformar depois permite **acesso mais rápido aos dados brutos** para consultas imediatas.
✅ **Flexibilidade**: Você pode transformar os dados **no momento necessário** e ajustar as transformações conforme as demandas analíticas.
✅ **Escalabilidade**: Com o ELT, você pode **aproveitar a capacidade de processamento da plataforma** (ex.: Apache Spark, Databricks) para transformar grandes volumes de dados de forma eficiente.
✅ **Eficiência em Grandes Volumes**: Reduz a necessidade de etapas intermediárias e facilita a manipulação de grandes volumes de dados.

Estratégias para modelagem de dados

Principais padrões de modelagem de dados em Lakehouses

Modelo de Dados de Tabelas Amplas (Wide Tables)

- **O que são?**: Tabelas grandes que armazenam dados amplos e detalhados, com muitas colunas.
- **Características**:
  - **Benefícios**: Simplificação no gerenciamento de dados e consultas.
  - **Desafios**: Pode gerar problemas de performance em grandes volumes de dados, especialmente nas transformações.
  - **Exemplo**: Tabela única de transações de clientes, produtos, e dados de campanhas de marketing.

Modelo Dimensional (Star Schema, Snowflake)

- **O que é?**: Estruturação dos dados em **fatos** e **dimensões** para facilitar consultas analíticas e BI.
- **Características**:
  - **Fatos**: Tabelas que armazenam dados quantitativos, como vendas ou transações.
  - **Dimensões**: Tabelas que fornecem contexto para os dados de fatos, como tempo, produto e região.
- **Exemplo**: Fato de vendas, dimensão de tempo, dimensão de produto.

Estratégias para modelagem de dados

Otimização de modelos de dados

Partition Keys (Chaves de Partição)

As **partition keys** são usadas para dividir os dados em **partições menores** dentro de uma tabela. Em um lakehouse, particionar uma tabela significa **organizar os dados em arquivos ou diretórios menores**, o que ajuda a distribuir os dados e melhorar a performance de **leitura** e **escrita**.

🔹 **Objetivo das Partition Keys:**

- **Eficiência nas leituras**: Quando você consulta dados em uma tabela particionada, o sistema pode **ignorar partições inteiras** se elas não forem necessárias para a consulta, melhorando a **performance**.
- **Escalabilidade**: Ao particionar os dados, você pode distribuir o trabalho entre múltiplos nós de computação, tornando o processo de **leitura e escrita mais escalável**.
- **Gerenciamento de dados históricos**: Em sistemas de lakehouse, particionar os dados por tempo (como ano, mês ou dia) é uma prática comum para organizar grandes volumes de dados históricos.

Estratégias para modelagem de dados

Otimização de modelos de dados

Em vez de armazenar todos os dados em uma única partição (ou arquivo), você pode definir uma **chave de partição** para dividir os dados em várias pastas, conforme algum critério. Isso pode ser feito por atributos como **data**, **categoria de produto**, **localização geográfica**, etc.

🔹 **Exemplos de Partition Keys:**

- `partitioned_by('year')`: Particiona os dados por ano. Isso é útil quando você tem dados históricos e deseja consultar rapidamente os dados de um ano específico.
- `partitioned_by('date')`: Particiona os dados por data, como ano/mês/dia. Isso é útil em cenários de dados transacionais, como logs de atividades.

🔹 **Vantagens das Partition Keys:**

- **Redução de custos de leitura**: Menos dados precisam ser lidos para consultas que envolvem apenas uma partição.
- **Consultas mais rápidas**: O mecanismo de leitura pode filtrar e buscar dados dentro de uma partição sem ler a tabela inteira.
- **Facilidade na manutenção**: Quando você precisa apagar ou arquivar dados antigos, você pode simplesmente remover ou mover uma partição inteira.

Estratégias para modelagem de dados

Otimização de modelos de dados

Clustering Keys (Chaves de Agrupamento)

As **clustering keys** (ou **chaves de agrupamento**) são usadas para organizar os dados **dentro de uma partição**. Enquanto as **partition keys** dividem os dados em partições separadas, as **clustering keys** são responsáveis por **organizar o ordenamento dos dados** dentro dessas partições. Isso é importante porque pode acelerar as **consultas de leitura** que fazem uso de **filtros** ou **joins**.

Objetivo das Clustering Keys:

- **Melhorar a performance de consulta**: Se você faz frequentemente **consultas com filtros** ou ordenações em uma coluna específica, usar uma chave de agrupamento pode otimizar o processo, porque os dados estarão **fisicamente organizados** de forma eficiente.
- **Reduzir o número de arquivos lidos**: Ao definir uma chave de agrupamento, você pode reduzir a quantidade de dados que precisam ser lidos dentro de uma partição.

Estratégias para modelagem de dados

Otimização de modelos de dados

A ideia é ordenar os dados dentro da partição de acordo com uma ou mais colunas. Isso melhora a **eficiência de buscas e operações de leitura** em dados que são frequentemente filtrados ou agregados com base nessas colunas.

- **Exemplos de Clustering Keys:**
  - `clustered_by('product_id')`: Quando você agrupa os dados por ID do produto dentro de cada partição. Isso ajuda em consultas rápidas de vendas por produto.
  - `clustered_by('date', 'region')`: Quando você agrupa os dados de vendas por data e região dentro de cada partição. Esse tipo de clustering melhora as consultas filtrando por ambos os campos ao mesmo tempo.
- **Vantagens das Clustering Keys:**
  - **Consultas mais rápidas**: Ao acessar dados em grandes conjuntos de dados particionados, o clustering permite que as consultas sejam mais rápidas, pois os dados são armazenados em uma ordem otimizada.
  - **Redução de fragmentação**: Evita que os dados fiquem desordenados ou dispersos dentro das partições, garantindo uma **leitura sequencial** mais eficiente.

Estratégias para modelagem de dados

Otimização de modelos de dados

Exemplo Prático em Lakehouses:

Imaginando que você tem um dataset de vendas com os seguintes atributos: **id_venda**, **data**, id_**produto**, **quantidade**, **valor_total**, **regiao**.

- **Partition Key**: Vamos dizer que você particione os dados pela **data** (ano, mês). Isso significa que você dividirá o dataset em várias partições, com cada partição contendo dados de um ano/mês em específico.
  - `partitioned_by('year', 'month')`
- **Clustering Key**: Dentro de cada partição (ano/mês), você pode querer organizar os dados por **id_produto** para otimizar consultas que frequentemente filtram por esse atributo.
  - `clustered_by('id_product')`

Isso permitiria que consultas como **“quais foram as vendas de um produto específico no mês de janeiro de 2024”** sejam extremamente rápidas, pois os dados estariam organizados de maneira eficiente dentro das partições de mês e produto.

Estratégias para modelagem de dados

Otimização de modelos de dados

Aspecto	Partition Keys	Clustering Keys
Função	Dividir os dados em **partições** físicas distintas.	Organizar os dados **dentro de uma partição**.
Objetivo	Melhorar a **performance de leitura** em grandes volumes de dados.	Melhorar a **performance de leitura** em consultas filtradas.
Exemplo	partitioned_by('year', 'month')	clustered_by('product_id')
Impacto	Reduz o número de **partições** lidas.	Melhora a ordenação e **acesso sequencial** de dados dentro de uma partição.
Uso	Ideal para grandes datasets históricos ou por **data**.	Ideal para dados com **consultas frequentes** em um campo específico.

Boas práticas para modelagem de dados

Discussões recentes sobre o uso de partition keys e clustering keys

Problemas com o particionamento tradicional

- **Over-partitioning (Particionamento excessivo)**:
  - Em modelos tradicionais, o particionamento de dados pode levar a um grande número de partições pequenas. Isso ocorre quando os dados são particionados com base em uma coluna que tem muitos valores distintos, como **data** ou **ID do cliente**.
  - Quando isso acontece, o **número de arquivos** aumenta consideravelmente, resultando em **ineficiência nas consultas**. Isso ocorre porque a leitura de muitas partições pequenas pode ser mais lenta e mais cara do que ler poucas partições grandes.
- **Manutenção e Gerenciamento de Partições**:
  - Gerenciar muitas partições pode ser difícil, especialmente em grandes datasets. Adicionar, remover ou atualizar partições pode gerar **sobrecarga operacional**. Além disso, partições mal projetadas podem afetar negativamente o desempenho de leitura e escrita.
- **Não há garantia de boa distribuição dos dados**:
  - Em alguns casos, o particionamento pode não garantir uma **distribuição uniforme** dos dados, o que pode levar a **hot spots** em algumas partições, gerando gargalos no processo de leitura e escrita.

Boas práticas para modelagem de dados

Discussões recentes sobre o uso de partition keys e clustering keys

Vantagens do Clustering sobre o Partitioning

🔹 **Melhor distribuição dos dados**:

- O **clustering** permite organizar os dados fisicamente de maneira **otimizada** dentro das partições sem criar múltiplas partições pequenas. Em vez de dividir os dados em muitas pastas, o clustering organiza os dados dentro de cada partição, tornando o processo de leitura mais eficiente.

🔹 **Consultas mais rápidas com ordenação e filtragem eficientes**:

- O clustering organiza os dados de acordo com uma ou mais **colunas-chave**, o que permite **consultas mais rápidas**. Quando você filtra por uma coluna usada no clustering, o Databricks pode localizar rapidamente os dados relevantes dentro da partição sem a necessidade de varrer todos os dados na tabela.
- Por exemplo, se você clusteriza uma tabela por `produto_id` ou `cliente_id`, consultas filtrando essas colunas serão mais rápidas, pois os dados estarão armazenados de forma sequencial.

Boas práticas para modelagem de dados

Discussões recentes sobre o uso de partition keys e clustering keys

Vantagens do Clustering sobre o Partitioning

- **Menos arquivos para ler**:
  - Como os dados são agrupados de maneira eficiente dentro de uma única partição, o número de **arquivos** lidos durante a consulta é reduzido. Isso reduz o tempo de leitura e melhora a performance de consultas.
- **Evita o problema de Over-partitioning**:
  - O uso de clustering não sofre com o problema de particionamento excessivo, pois você não está criando **muitas partições pequenas**. Ao invés disso, você organiza os dados de forma mais inteligente dentro de um número controlado de partições maiores.
- **Flexibilidade e escalabilidade**:
  - O clustering é mais **flexível** e se adapta melhor a **consultas diversas**. Enquanto o particionamento pode ser rígido (você precisa decidir como particionar os dados de acordo com um critério específico), o clustering permite que você organize os dados de acordo com múltiplos critérios e otimize as leituras baseadas em diferentes padrões de acesso.

Boas práticas para modelagem de dados

Padrão de nomenclatura de tabelas - Modelo DBT

🔹 `stg_` **– Staging Tables (Tabelas de Preparação)**

- **O que são?**
  Tabelas de **pré-processamento** ou **staging** são usadas para **armazenar dados brutos** logo após a ingestão. Esses dados podem ser provenientes de várias fontes, como APIs, bancos de dados, arquivos, etc. Essas tabelas servem como um ponto de entrada onde os dados são carregados e podem ser rapidamente transformados ou limpos antes de serem movidos para o próximo estágio.
- **Objetivo:**
  - Limpeza inicial dos dados, remoção de duplicatas e padronização.
  - Transformações simples, como **conversões de tipos** ou **remoção de colunas desnecessárias**.
- **Exemplo de Tabela `stg_`:**
  - `stg_customers` - Tabela com os dados brutos dos clientes, como nome, endereço e e-mail.
  - `stg_orders` - Tabela com dados de pedidos extraídos de uma fonte externa.

Boas práticas para modelagem de dados

Padrão de nomenclatura de tabelas - Modelo DBT

`int_` – Intermediate Tables (Tabelas Intermediárias)

- **O que são?**
  Tabelas intermediárias são usadas para **transformações mais complexas** que preparam os dados para os modelos finais. Elas geralmente servem como uma camada de **intermediação entre as tabelas de preparação (`stg_`) e as tabelas de fatos ou dimensões**. A ideia aqui é **refinar** ainda mais os dados antes que eles sejam organizados para análise.
- **Objetivo:**
  - Realizar transformações mais profundas, como **joins** entre tabelas, **agregações** ou criação de **novas métricas**.
  - Tornar os dados mais úteis para **consultas analíticas**.
- **Exemplo de Tabela `int_`:**
  - `int_sales` - Tabela intermediária que agrega as vendas por produto, mês e região.
  - `int_customers` - Tabela que limpa e formata os dados dos clientes, removendo inconsistências e criando campos calculados.

Boas práticas para modelagem de dados

Padrão de nomenclatura de tabelas - Modelo DBT

`dim_` – Dimension Tables (Tabelas de Dimensões)

- **O que são?**
  As tabelas de dimensões armazenam **informações qualitativas** e **descritivas** que ajudam a fornecer contexto para os dados em tabelas de fatos. Essas tabelas são essenciais para a modelagem **dimensional**, como o **Star Schema** ou **Snowflake Schema**.
- **Objetivo:**
  - Armazenar informações que podem ser usadas para **filtrar, agrupar ou detalhar** as métricas nas tabelas de fatos.
  - Exemplos de dimensões incluem **tempo**, **cliente**, **produto**, **região**, etc.
- **Exemplo de Tabela `dim_`:**
  - `dim_product` - Tabela que contém informações sobre os produtos, como categoria, marca, e descrição.
  - `dim_time` - Tabela de dimensão de tempo, com informações sobre dias, semanas, meses e anos.

Boas práticas para modelagem de dados

Padrão de nomenclatura de tabelas - Modelo DBT

`fct_` – Fact Tables (Tabelas de Fatos)

- **O que são?**
  As tabelas de **fatos** são o coração da modelagem dimensional. Elas armazenam **dados quantitativos**, como **métricas, agregações e transações**, e se relacionam com as **tabelas de dimensões** para fornecer contexto às métricas. As tabelas de fatos geralmente contêm dados históricos ou transacionais que são analisados para gerar insights.
  - **Exemplo de métricas**: Vendas, receitas, custos, contagens, etc.
- **Objetivo:**
  - Armazenar **métricas e medidas** numéricas, como volume de vendas ou lucros.
  - **Relacione-se com as tabelas de dimensões** para fornecer contexto analítico.
- **Exemplo de Tabela `fct_`:**
  - `fct_sales` - Tabela de vendas, com informações sobre transações, como quantidade vendida, preço e descontos aplicados.
  - `fct_transactions` - Tabela de transações financeiras, com métricas sobre pagamentos e encargos.

Boas práticas para modelagem de dados

Padrão de nomenclatura de tabelas - Modelo DBT

`mart_` – Data Marts (Marts de Dados)

- **O que são?**
  Data marts são **conjuntos de dados otimizados** para um grupo específico de usuários ou para uma **área de negócios** em particular, como **finanças**, **vendas** ou **marketing**. Essas tabelas geralmente contêm **dados agregados** ou **curados**, preparados para análise e visualização.
- **Objetivo:**
  - **Otimizar** as consultas analíticas para um caso de uso específico.
  - Fornecer **relatórios e dashboards** específicos para departamentos ou equipes.
- **Exemplo de Tabela `mart_`:**
  - `mart_sales_summary` - Tabela agregada com vendas totais por região e produto, projetada para análises rápidas.
  - `mart_financials` - Tabela contendo métricas financeiras agregadas para a equipe de finanças, como receita, lucro e custo.

Boas práticas para modelagem de dados

Padrão de nomenclatura de tabelas - Modelo DBT

Prefixo	Tipo de Tabela	Objetivo	Exemplo
stg_	Staging Tables (Pré-processamento)	Dados brutos, preparados para limpeza e transformação.	stg_customers
int_	Intermediate Tables (Intermediárias)	Transformações intermediárias, agregações e junções.	int_sales
dim_	Dimension Tables (Dimensões)	Contextualizar os dados de fatos para análise.	dim_product, dim_time
fct_	Fact Tables (Fatos)	Métricas e dados quantitativos, como transações e agregações.	fct_sales, fct_transactions
mart_	Data Marts (Marts de Dados)	Conjuntos otimizados de dados para áreas específicas.	mart_sales_summary

Orquestração de tarefas

A orquestração de tarefas é o coração da automação de pipelines de dados. Ela permite que você coordene a execução de processos complexos de forma eficiente, garantindo que cada etapa do pipeline seja realizada na ordem correta e no momento certo.

🔹 **Antes: Orquestração Manual e Ad-Hoc**
❌ Processos manuais e scripts isolados para orquestrar tarefas
❌ Dificuldade em garantir a **sequência correta** de execução
❌ Monitoramento e alerta dependiam de ferramentas externas
❌ Desafios para escalabilidade e flexibilidade em grandes volumes de dados

🔹 **Agora: Orquestração Automática e Inteligente**
✅ **Airflow** e outras ferramentas para automatizar e coordenar workflows
✅ Definição clara de **dependências** entre tarefas usando DAGs
✅ Execução **paralela ou sequencial** de tarefas conforme necessidade
✅ **Monitoramento** e **alertas automáticos** para falhas em tempo real
✅ **Escalabilidade**: Orquestração que cresce conforme os volumes de dados aumentam

Com esse modelo de orquestração inteligente, você ganha **eficiência operacional**, **flexibilidade** e **visibilidade** em todo o pipeline de dados, melhorando o desempenho e a confiabilidade de cada tarefa!

Governança da plataforma de dados

Definição

Governança de dados refere-se ao conjunto de processos, políticas e normas que garantem a **segurança**, **qualidade** e **conformidade** dos dados em uma plataforma.

Ela assegura que os dados sejam acessíveis, consistentes e confiáveis, além de garantir que o uso e o compartilhamento dos dados sigam as regulamentações e as melhores práticas.

Governança da plataforma de dados

Pilares da governança de dados

Segurança de Dados

- Proteção contra acessos não autorizados.
- Implementação de **criptografia**, **controle de acessos** e **autenticação** robustos.

Conformidade Regulatória

- Adequação a regulamentações como **LGPD**, **GDPR**, e outras leis de privacidade e proteção de dados.
- **Auditoria** e **rastreabilidade** para garantir a transparência no uso dos dados.

Qualidade de Dados

- Garantia de que os dados sejam **precisos**, **completos** e **confiáveis** para tomada de decisão.
- Implementação de processos para **monitoramento contínuo da qualidade** e correção de problemas de dados.

Gestão de Metadados

- Organização e catalogação dos **metadados** para facilitar a **descoberta** e o **uso eficiente** dos dados.
- Criação de uma **linguagem comum** para descrever dados e aumentar a **compreensão** entre diferentes equipes.

Data Mesh — Arquitetura de Dados Descentralizada

O **Data Mesh** é um modelo de arquitetura de dados **distribuído** e **organizado por domínios de negócio**, que busca aumentar a **flexibilidade**, a **escalabilidade** e a **eficiência** no uso de dados em grandes empresas.

Em vez de centralizar os dados em um único repositório, ele propõe que **cada equipe de domínio seja responsável** pelos dados que gera, tratando-os como **ativos valiosos** e bem definidos.

Data Mesh — Arquitetura de Dados Descentralizada

Princípios:

- **Responsabilidade por Domínio:**
  Cada área (como Marketing, Vendas, Logística) gerencia seus próprios dados — garantindo controle, qualidade e compartilhamento com outras áreas.

- **Dados como Produtos:**
  Os dados são disponibilizados com documentação, regras de uso, indicadores de qualidade e disponibilidade, facilitando seu reuso por toda a organização.

- **Acesso Autosserviço:**
  Plataformas facilitam que qualquer equipe descubra e use dados sem depender de times centralizados de engenharia.

- **Arquitetura Federada e Processamento Distribuído:**
  A infraestrutura pode ser heterogênea e descentralizada, respeitando padrões de governança e aproveitando tecnologias de processamento escalável.

Obrigado!