Introdução a Pipeline de dados com Duckdb

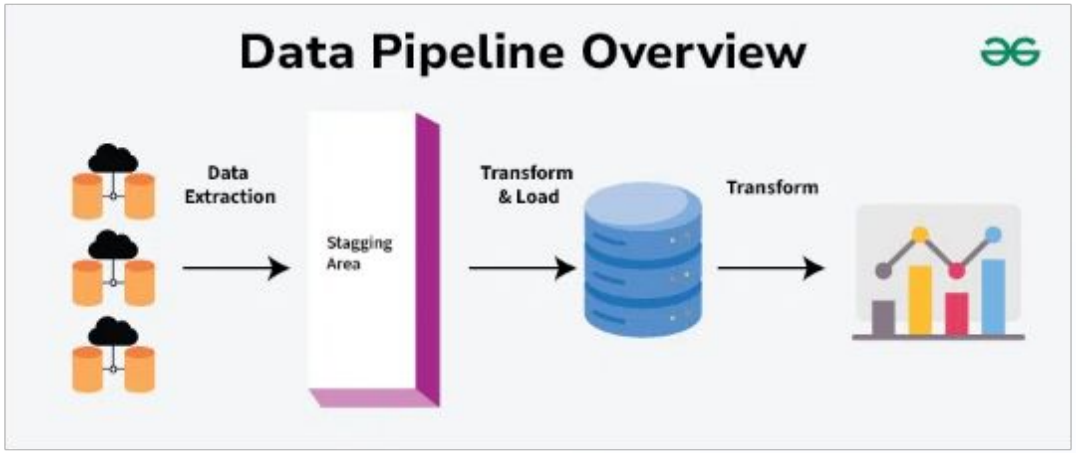
Conceitos de uma Stack Moderna para Pipeline de Dados

Pipeline de Dados

Um pipeline de dados é uma sequência de processos automatizados que captura, transforma e carrega dados de diversas fontes para um destino, normalmente com o objetivo de análise, visualização ou tomada de decisão.

Etapas comuns:

- **Ingestão:** captura de dados (de APIs, bancos, arquivos etc.)
- **Transformação:** limpeza, enriquecimento e reestruturação dos dados
- **Armazenamento e consulta:** envio para bancos de dados, data warehouses ou data lakes
- **Consumo:** dashboards, modelos de ML, relatórios



ELT vs ETL

ETL (Extract, Transform, Load)

- **Extração:** obtém dados da fonte
- **Transformação:** limpa e organiza antes de armazenar
- **Carga:** envia os dados prontos ao destino

Usado quando: há necessidade de transformar os dados antes de armazenar.

ELT (Extract, Load, Transform)

- **Extração:** obtém dados da fonte
- **Carga:** carrega os dados brutos no destino
- **Transformação:** transforma os dados **dentro do banco** (ex: via SQL)

Usado quando: o banco de destino é potente e permite transformação rápida (ex: Snowflake, BigQuery, DuckDB).

Ferramentas de Inserção

Essas ferramentas automatizam a **ingestão** de dados, conectando-se a APIs, arquivos ou bancos de dados.

Airbyte

Código aberto, permite conectar diversas fontes e destinos.

Fivetran

Solução paga com foco em ELT.

Singer

Especificação para ingestão de dados via arquivos "tap" e "target".

Python personalizado

Scripts para ingestão sob medida.

Ferramentas de Pipelines de Dados

São usadas para **orquestrar** as etapas do pipeline (agendamento, monitoramento, dependências).

- **Prefect:** pipelines com foco em simplicidade e observabilidade
- **Dagster:** design modular, com foco em desenvolvimento e validação
- **Airflow:** robusto e altamente configurável, ideal para ambientes complexos
- **Luigi:** bom para pipelines locais e dependências simples
- **Make / bash / crontab:** soluções caseiras e simples para fluxos pequenos

Banco de Dados Colunares vs Relacionais

Diferente dos bancos tradicionais (linha a linha), os bancos colunares armazenam os dados **por coluna**, o que permite otimizações para consultas analíticas.

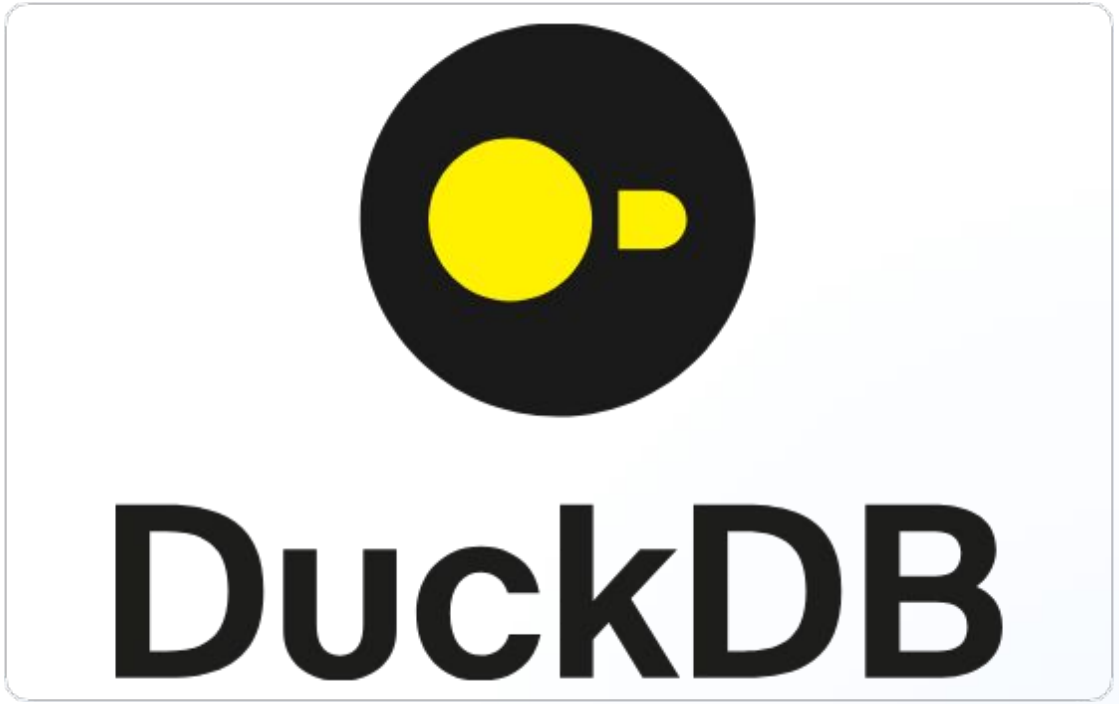
Característica	Relacional (row-based)	Colunar (columnar)
Armazenamento	Linha por linha	Coluna por coluna
Otimizado para	Escrita, OLTP	Leitura, OLAP
Exemplos	PostgreSQL, MySQL	DuckDB, ClickHouse, BigQuery

DuckDB

DuckDB é um banco de dados analítico **colunar**, projetado para rodar **localmente**, dentro do seu processo Python (sem servidor).

Vantagens:

- Extremamente leve (sem necessidade de servidor)
- Suporte nativo a formatos como Parquet e CSV
- Integração direta com pandas, Arrow e NumPy
- Alta performance em queries analíticas
- Ideal para prototipação e ambientes de desenvolvimento



Formas de Uso do DuckDB

- Como banco **embutido** em scripts e notebooks
- Com arquivos locais **.csv, .parquet, .json**
- Para consultar **DataFrames diretamente** (sem gravar no disco)
- Como camada intermediária de análise antes de enviar para um datawarehouse

```
import duckdb
import pandas as pd
# Carrega dados em um DataFrame
df = pd.read_csv("dados.csv")
# Executa uma query SQL diretamente no DataFrame
resultado_df = duckdb.query(""" SELECT * FROM df WHERE valor > 100
""").to_df()
```

Perguntas?