Introdução à Engenharia de Dados

Agenda

- Entender o papel da Engenharia de Dados na Ciência de Dados
- Compreender tipos de dados (estruturados, semi e não estruturados)
- Realizar extração e transformação com Python
- Aplicar validações simples em dados reais (Censo Escolar)
- Entender o conceito de zonas de dados (raw, staging, serving)

Apresentação

Abílio Nogueira Barros

28 anos

Mestre em Informática Aplicada-PPGIA-UFRPE

Doutorando em Engenharia da Computação-PPGEC-UPE

Engenheiro de Dados em projetos dados abertos/dados educacionais

Apresentação da turma

- Nome
- Área de atuação
- Background em dados
- Qual sua opinião sobre dados abertos?

O que é Engenharia de Dados?

Definição

Área responsável por projetar, construir e manter sistemas de coleta, armazenamento e processamento de dados em escala.

Objetivo Principal

Garantir que os dados estejam **acessíveis**, **confiáveis** e **prontos para análise**.

O Engenheiro de Dados no Ciclo de Ciência de Dados

Infraestrutura

Cria pipelines para alimentar modelos.

Qualidade

Assegura dados limpos e consistentes.

Performance

Otimiza consultas e processamento.

Tipos de dados

Conceito: Os tipos de dados representam a forma como a informação está estruturada, o que influencia diretamente na forma como podemos coletar, processar, transformar e armazenar esses dados.

Dados Estruturados

Definição

Dados organizados em linhas e colunas fixas, como em tabelas de bancos relacionais (SQL) ou arquivos CSV.

Exemplos

Arquivos .csv, .xls
Tabelas em bancos de dados relacionais (PostgreSQL, MySQL, SQLite)

Características

Fácil de consultar com SQL
Apresentam um bom nível de padronização

Dados Semi-estruturados

Definição

Dados que possuem alguma estrutura interna (como chave-valor), mas não seguem uma tabela rígida.

Exemplos

JSONs retornados de APIs (ex: IBGE, INEP)
Arquivos XML, YAML
Logs de sistema

Características

Flexíveis e auto-descritivos
Estrutura aninhada
Falta de padronização

Dados Não Estruturados

Definição

Dados sem estrutura pré-definida, não são organizados em linhas e colunas.

Exemplos

Imagens (JPEG, PNG)
Áudios (MP3, WAV)
Vídeos, PDFs, documentos escaneados

Características

Alto volume
Dificuldade na extração dos dados
Necessitam de processamento especializado (NLP, OCR)

Comparativo dos Tipos de Dados

Tipo de Dado	Facilidade de Processamento	Exemplos
Estruturado	Alta	CSV com dados de escolas e matrículas
Semi-estruturado	Média	JSON
Não estruturado	Baixa	PDFs com relatórios

Fonte de dados

Fonte de dados

Definição: Fontes de dados são os locais ou sistemas de onde os dados são gerados, armazenados ou disponíveis para consumo. Elas representam o ponto de partida do pipeline de dados e variam amplamente em estrutura, formato, volume e frequência de atualização.

Fonte de dados comuns

Tipo	Descrição	Exemplos
Bancos Relacionais	Estruturados em tabelas com esquema fixo. Utilizam SQL para acesso.	MySQL, PostgreSQL, Oracle, SQL Server
Arquivos locais	Dados armazenados em arquivos, geralmente em formatos padrão.	CSV, Excel (XLSX), JSON, Parquet, Avro
APIs Web	Interfaces que permitem acesso a dados via requisições HTTP/REST.	Twitter API, IBGE API, OpenWeather, INEP
Data Warehouses	Armazenam grandes volumes de dados para análises e relatórios.	BigQuery, Redshift, Snowflake, Synapse

Fonte de dados comuns

Tipo	Descrição	Exemplos
Data Lakes	Armazenam dados em qualquer formato (estruturado ou não).	Amazon S3, Azure Data Lake, GCP Cloud Storage
Sistemas de Arquivos	Diretórios locais ou em nuvem com arquivos dispersos.	FTP/SFTP, HDFS, diretórios em rede
Streams/Logs	Fontes que produzem dados em tempo real.	Kafka, MQTT, Flink, Kinesis
Web Scraping	Dados extraídos diretamente de sites por meio de scripts automatizados.	BeautifulSoup, Scrapy

Formas de extração de dados

Método de Extração	Descrição	Quando usar
Extração direta por SQL	Realiza queries diretamente no banco de dados.	Bancos relacionais com acesso liberado
Leitura de arquivos locais	Leitura de CSVs, JSONs ou outros formatos com pandas, pyarrow, etc.	Quando os dados estão exportados em arquivos
Extração por API REST	Utiliza endpoints HTTP para coletar dados com autenticação e filtros.	Dados públicos ou internos expostos por API
Conectores ETL/ELT	Uso de conectores como Singer Taps, Airbyte, Talend.	Integração de múltiplas fontes de forma padronizada

Formas de extração de dados

Método de Extração	Descrição	Quando usar
Streaming de dados	Consumo contínuo de eventos em tempo real.	Casos com dados de sensores, logs, cliques em tempo real
Web Scraping	Automação para extrair dados de sites, respeitando políticas de uso.	Sites que não oferecem APIs, último recurso
Leitura em lote	Extrações periódicas com agendamentos (cron jobs, Airflow DAGs).	Bases com atualizações mensais, semanais, arquivos por período

Pipeline de dados

Pipeline de dados

Definição: Um pipeline de dados é um conjunto de processos automatizados que permitem mover dados de uma fonte até seu destino final (geralmente um repositório de dados ou ferramenta analítica), passando por etapas como extração, transformação e carregamento.

Atribuições da engenharia de dados:

- Desenvolver e manter pipelines de dados confiáveis e escaláveis;
- Garantir qualidade, integridade e performance do fluxo de dados;
- Automatizar o processamento por meio de scripts, orquestradores e ferramentas;

Tipos de pipeline: ETL (Extract – Transform – Load)

Modelo tradicional, onde os dados são extraídos, transformados localmente (normalmente com scripts Python ou ferramentas como Spark), e depois carregados já tratados no destino.

Exemplos:

- Extrair dados brutos do Censo Escolar (CSV)
- Tratar dados com pandas (corrigir nulos, converter tipos)
- Carregar para PostgreSQL, BigQuery, ou .parquet

Tipos de pipeline: ETL (Vantagens e Desvantagens)

Vantagens

Dados chegam ao destino prontos para consumo

Maior controle da qualidade e regras de negócio

Desvantagens

Processamento prévio exige infraestrutura local

Menos eficiente para grandes volumes em cloud

ELT (Extract – Load – Transform)

Estratégia moderna, comum em ambientes de cloud e data lakes, onde os dados são primeiro extraídos e carregados brutos no destino (zona raw), e transformados depois com ferramentas analíticas ou SQL.

Exemplos:

- Carregar os dados em um Data Lake
- Rodar scripts SQL ou ferramenas como o DBT para tratamento direto no banco de dados.

ELT (Vantagens e Desvantagens)

Vantagens

Ideal para grandes volumes e pipelines em cloud
Permite reaproveitar dados brutos em diferentes transformações

Desvantagens

Transformações podem ser menos controladas
Necessita mais governança para manter a qualidade

Comparativo entre ETL e ELT

Critério	ETL	ELT
Ordem das etapas	Extrai → Transforma → Carrega	Extrai → Carrega → Transforma
Onde ocorre a transformação	Local (Python, Spark)	No destino (SQL, dbt, BigQuery)
Indicado para	Dados sensíveis, lógica complexa	Alto volume, cloud, flexibilidade
Velocidade inicial	Mais lento	Mais rápido para ingestão
Exemplo prático	Python + Pandas → PostgreSQL	CSV → BigQuery → SQL Transformações

Ferramenta ETL Open Source

Ferramenta	Descrição	Linguagem/Base	Destaques principais
Apache Airflow	Orquestrador de workflows. Define pipelines como código com Python.	Python	DAGs, agendamento, integração com Spark, Bash, SQL, etc.
Kettle (Pentaho PDI)	Ferramenta gráfica para ETL, rica em conectores e transformações.	Java	Interface visual amigável para protótipos e ETLs corporativos.

Ferramentas ELT Open Source

Ferramenta	Descrição	Linguagem/Base	Destaques principais
dbt (Data Build Tool)	Ferramenta para transformar dados com SQL no data warehouse.	SQL + Jinja	Modularização, versionamento, testes e documentação automática.
Singer.io	Protocolo open-source para criar conectores de extração e carga de dados.	JSON + Python	Reutilizável via "taps" (extração) e "targets" (carga), base do Meltano.

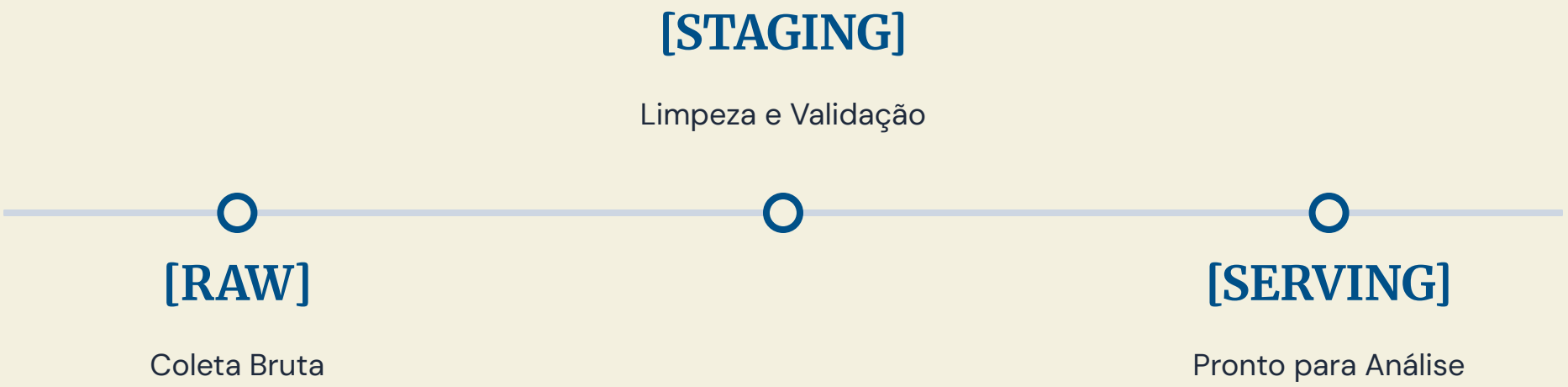
Zona de Dados: RAW, STAGING e SERVING

Contextualização: Em uma arquitetura de dados moderna, os dados passam por diferentes estágios de processamento. Esses estágios são comumente organizados em "zonas" dentro de um Data Lake ou Data Lakehouse, ajudando a manter a organização, a rastreabilidade e a governança dos dados.

O que são zonas de dados?

As zonas representam níveis de maturidade e confiabilidade dos dados. À medida que os dados passam por essas zonas, eles são limpos, transformados e preparados para consumo.

Fluxo entre as zonas



Zona RAW (Bronze)

- **Objetivo:** armazenar os dados exatamente como foram extraídos
- **Formato:** CSV, JSON, dumps, logs
- **Características:** Sem limpeza ou transformação, Reproduzível, Serve como "verdade bruta"
- **Ferramentas:** Airflow, Python

Zona STAGING (Prata)

- **Objetivo:** transformar, padronizar e validar os dados
- **Ações comuns:** Conversão de tipos, Remoção de nulos e duplicatas, Validação de schema
- **Formatos comuns:** tabelas intermediárias
- **Ferramentas:** dbt, PySpark,Airflow

Zona SERVING (Ouro)

- **Objetivo:** fornecer dados prontos e otimizados para consumo
- **Características:** Estruturados para análise e dashboards, Contêm métricas, KPIs, agregações
- **Ferramentas de consumo:** PowerBi, Tableau, Metabase

Tabela: Zonas de Dados

Zona	Descrição
Raw	Área bruta, onde os dados são armazenados exatamente como foram extraídos da fonte.
Staging	Etapa intermediária com limpeza, padronização e validação de qualidade.
Serving	Área final, com dados prontos para uso analítico, relatórios, dashboards e ML.

Por que dividir por zonas?

- **Governança e rastreabilidade:** Saiba de onde o dado veio e o que foi feito com ele.
- **Performance e escalabilidade:** Otimize o armazenamento e processamento para cada etapa.
- **Segurança e conformidade:** Aplique regras de acesso e LGPD em diferentes níveis.

Recapitulando...

Zona	Papel no pipeline
RAW	Coleta e armazenamento bruto
STAGING	Transformação e validação
SERVING	Consumo e análise de dados

A Importância do Dicionário de Dados

Por que criar um dicionário de dados?

Um dicionário de dados documenta o significado, formato, origem e regras de cada campo em um dataset. Ele é essencial para garantir **transparência, entendimento e consistência** no trabalho com dados.

Benefícios principais

- **Documentação clara:** descreve nome, tipo, unidade, regras e domínio dos dados
- **Facilita a colaboração:** ajuda equipes diferentes a entenderem os dados
- **Validações automatizadas:** permite definir padrões para validação e limpeza
- **Auditoria e rastreabilidade:** registra transformações e versões dos dados
- **Redução de erros:** previne uso indevido ou interpretações equivocadas

Dúvidas ou perguntas?

Obrigado

abilionbarros@gmail.com