Introdução ao Processamento de Dados

Integração e Armazenamento de Dados

Professor Douglas Castro

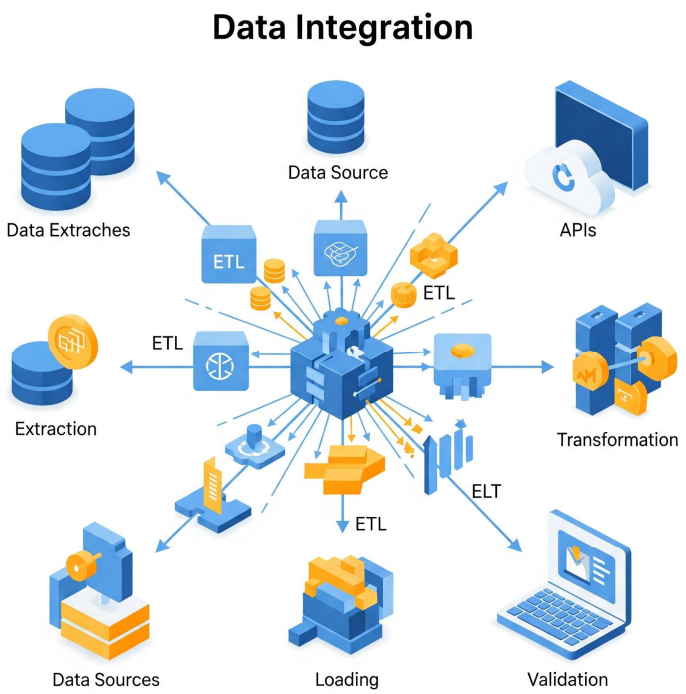
O que é Integração de Dados?

Integração de dados: processo de combinar dados de diferentes fontes e formatos em um conjunto único e consistente.

Objetivo: disponibilizar dados completos, confiáveis e prontos para análise.

Exemplos comuns:

- Unir dados de escolas com dados de estudantes.
- Combinar bases de vendas de diferentes filiais.
- Integrar dados internos com informações públicas (IBGE, Receita Federal).



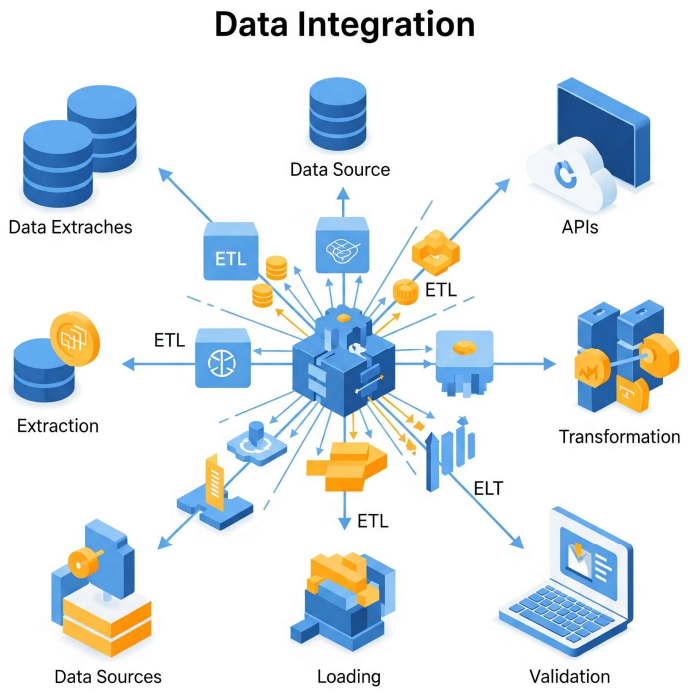
Desafios típicos na integração de dados

Diferença de formatos (CSV, Excel, bancos de dados, APIs).

Dados duplicados ou conflitantes.

Padronização de nomes e tipos de variáveis.

Necessidade de "chave" para fazer o relacionamento (Ex: código da escola).



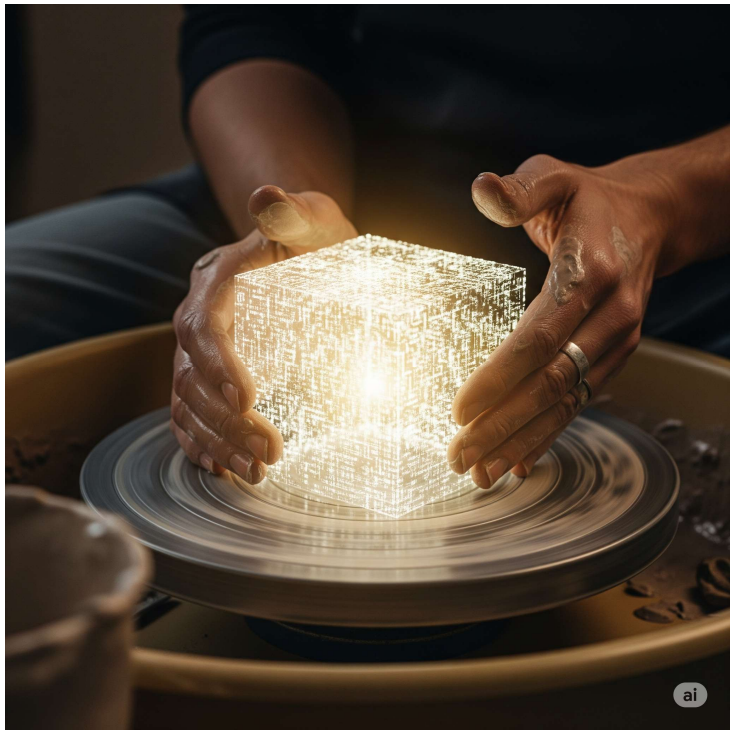
Introdução à modelagem de dados

Modelagem de dados: processo de organizar e estruturar os dados de uma organização.

Define **entidades** (Ex: Estudante, Escola) e **relacionamentos** entre elas.

Ajuda a manter a consistência, evitar redundância e melhorar o desempenho.

Tipos principais:
Modelagem relacional (tabelas ligadas por chaves)
Modelagem dimensional (para Data Warehouses: fatos e dimensões)



Formatos de armazenamento de dados

Arquivos tabulares: CSV, Excel — simples, mas limitados para grandes volumes.

JSON/XML: estruturados, ideais para integração com sistemas e APIs.

Bancos de dados relacionais (PostgreSQL, MySQL): escaláveis, com integridade e flexibilidade.

NoSQL (MongoDB, Cassandra): dados semi-estruturados ou não estruturados.

Data Lakes: armazenamento de grandes volumes em formatos brutos (usados em Big Data).

Por que armazenar bem importa?

Facilita análises, dashboards e relatórios.

Garante a integridade e segurança dos dados.

Permite escalabilidade e reuso dos dados em diferentes projetos.

O que é um pipeline de dados?

Pipeline de dados: sequência de etapas automáticas para coletar, transformar, integrar, armazenar e disponibilizar dados.

Ajuda a garantir dados atualizados e consistentes para o negócio.

Exemplo:

Coleta dos dados brutos

Limpeza e transformação

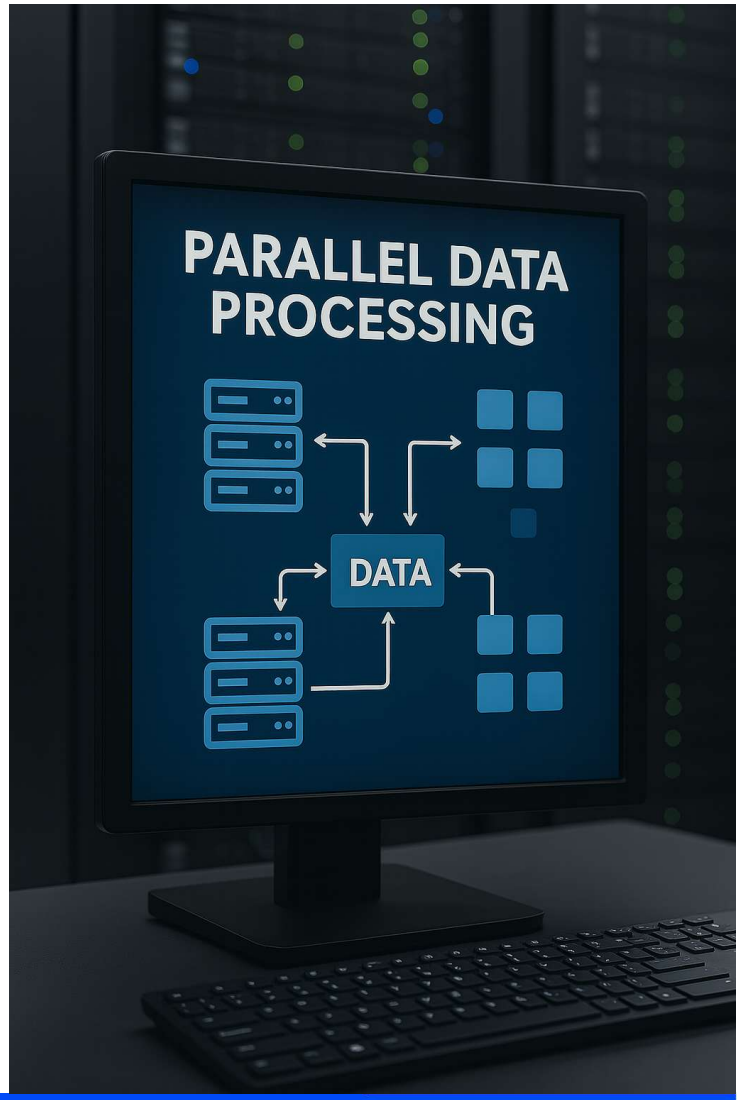
Integração de múltiplas fontes

Armazenamento

Entrega para análise ou BI

Notebook 2

PROCESSAMENTO EM PARALELO



Por que usar processamento paralelo?

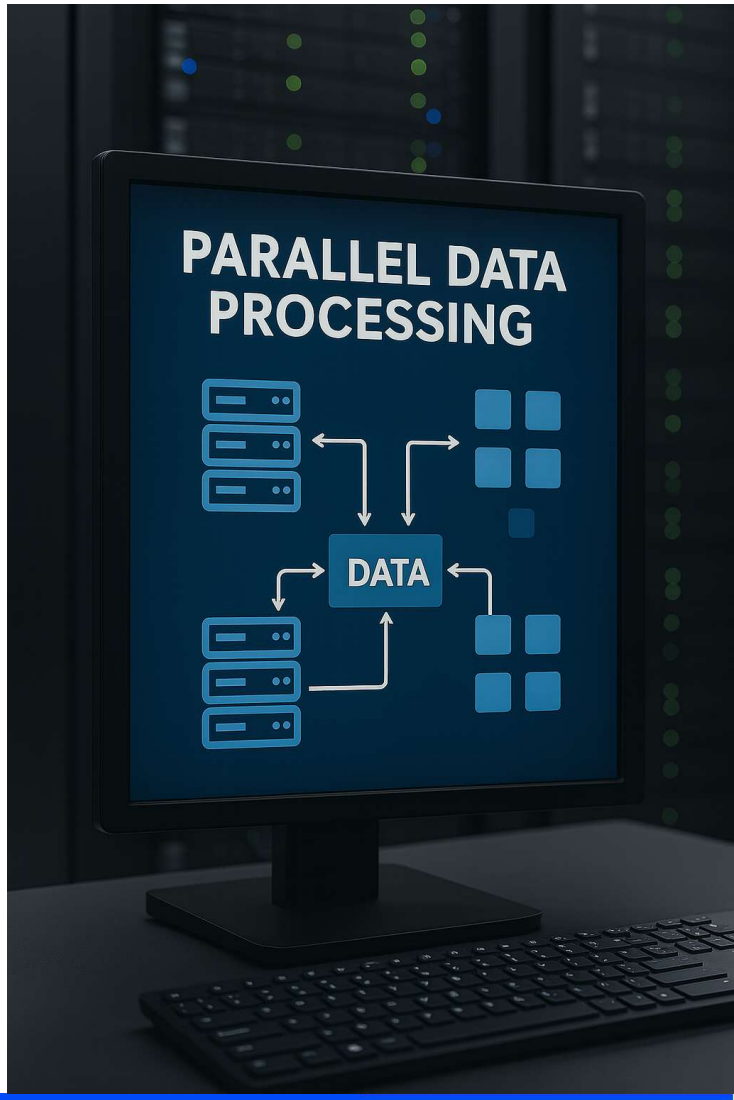
Grandes volumes de dados consomem muito tempo para processar em modo sequencial.

Computadores modernos têm múltiplos núcleos de CPU: é possível distribuir tarefas e acelerar o processamento.

Exemplo prático:

Aplicar transformações pesadas em milhões de linhas de um DataFrame.

Realizar cálculos em lote (por exemplo: notas, status de aprovação, indicadores, etc).

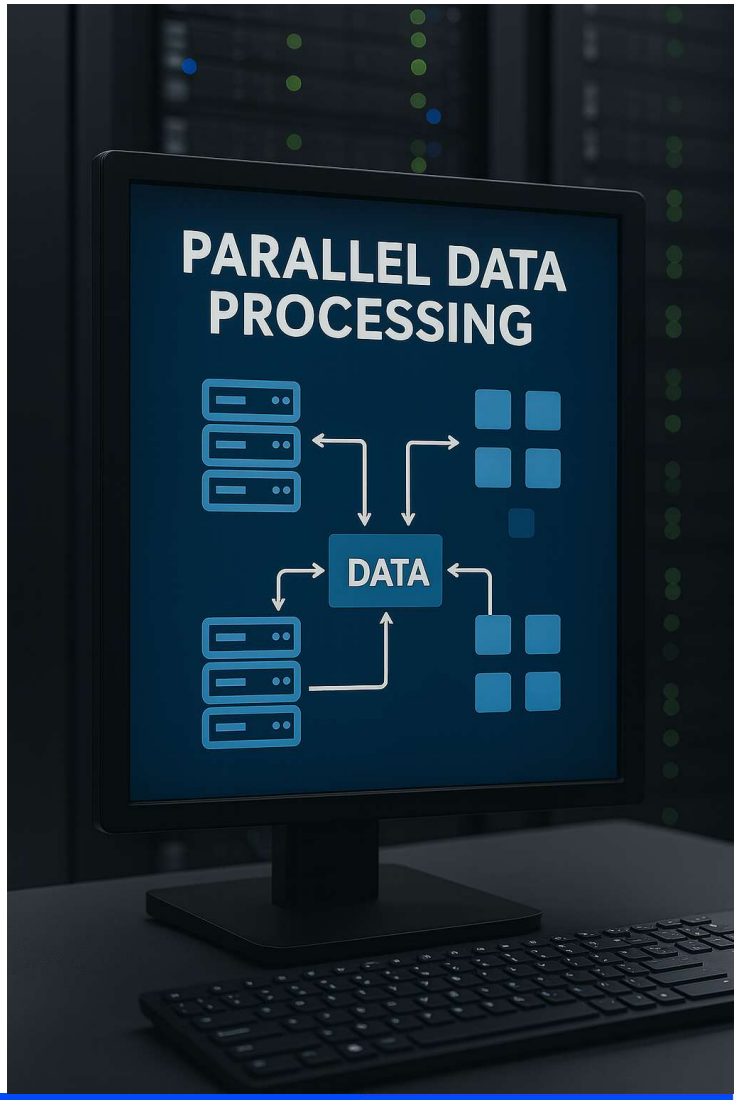


O que é o módulo multiprocessing?

Multiprocessing é uma biblioteca padrão do Python para criar múltiplos processos de forma simples.

Permite que diferentes partes do código rodem simultaneamente, cada uma em um núcleo da CPU.

Ideal para tarefas que exigem muito do processador (CPU-bound).



Vantagens e cuidados do multiprocessing

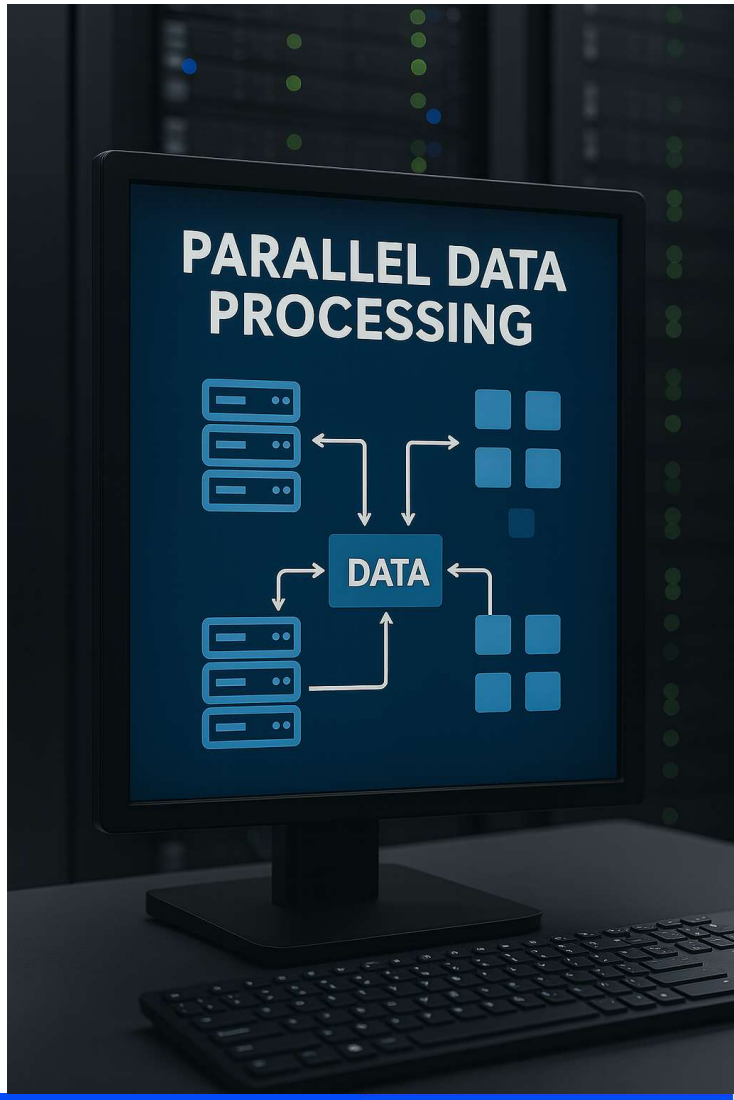
Redução do tempo total de execução.

Uso eficiente dos recursos do computador.

Dividir os dados corretamente para cada processo.

Atenção ao uso de variáveis globais: cada processo tem sua própria memória.

Processos consomem mais memória do que threads.



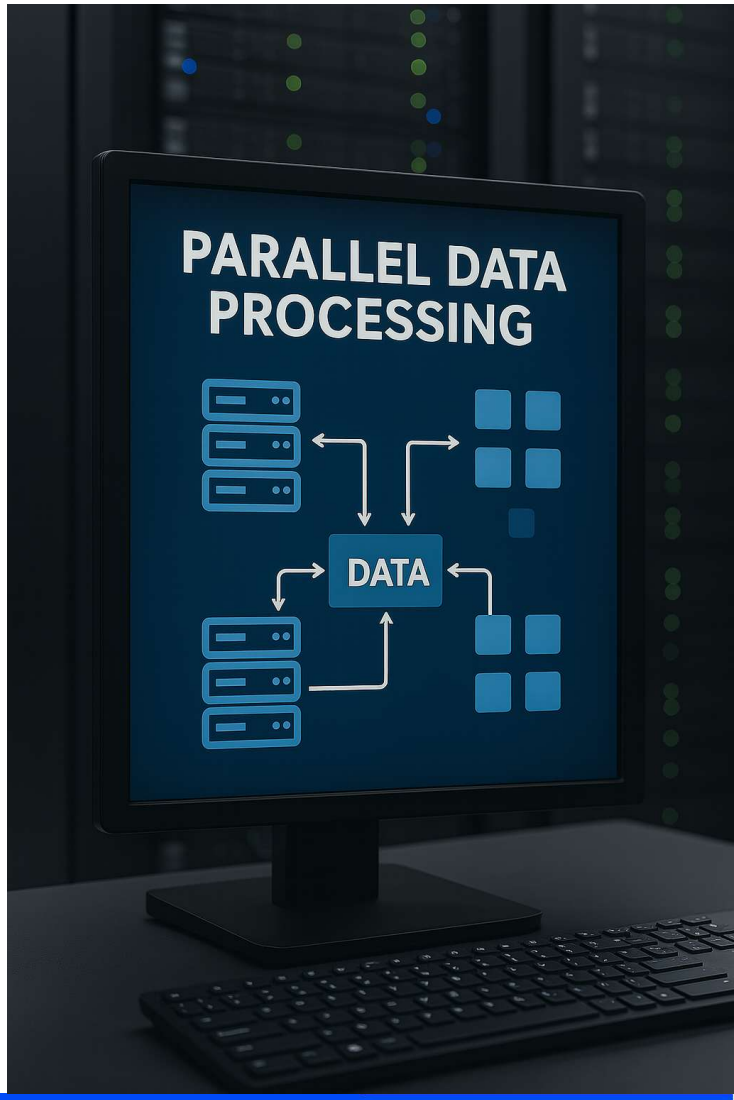
Como funciona o ProcessPoolExecutor?

ProcessPoolExecutor faz parte do módulo concurrent.futures (Python 3).

Permite executar funções em paralelo, gerenciando um "pool" de processos.

Mais simples e moderno do que trabalhar diretamente com multiprocessing.Process.

Facilita o envio de funções e dados para vários processos.

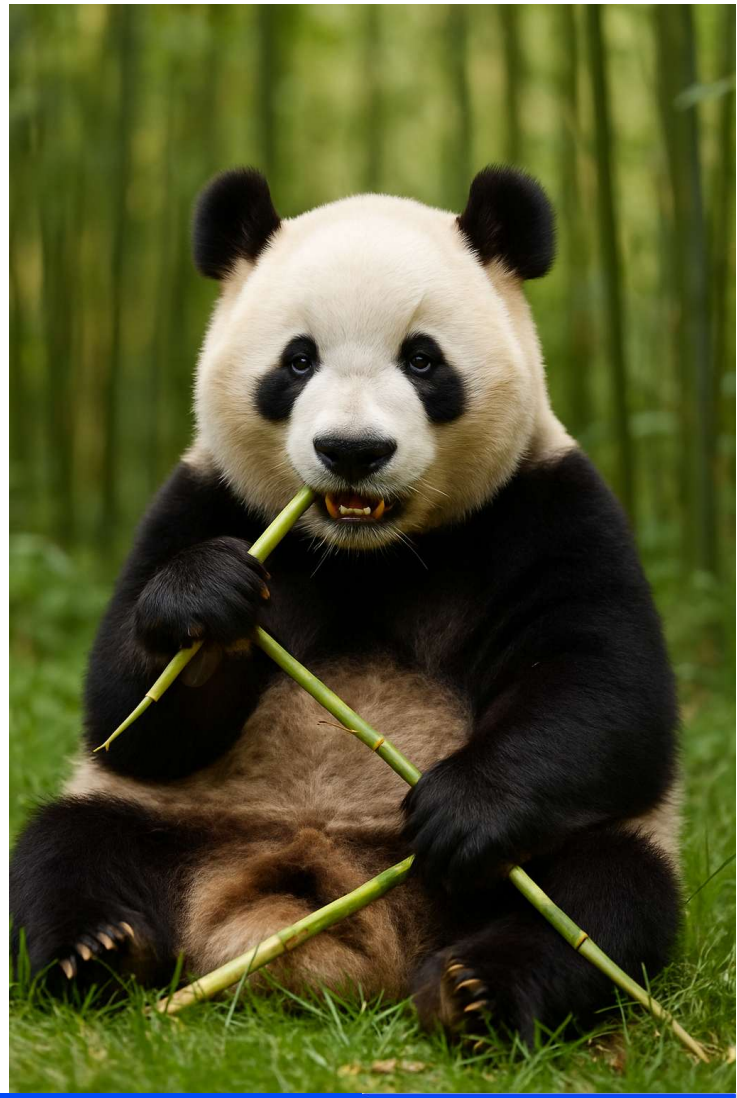


Aplicação com Pandas

DataFrames grandes podem ser divididos em partes menores.

Cada processo pode aplicar uma função diferente em um pedaço do DataFrame.

Ao final, os pedaços são reunidos novamente com pd.concat.



Notebook 2