ETLs, ELTs e Pipelines de Dados

## Introdução

Olá, seja bem-vindo a este módulo sobre extrações automatizadas de dados entre sistemas!

Nos dias de hoje, as empresas lidam com grandes quantidades de dados em diferentes sistemas. Para tomar decisões informadas e eficientes, é crucial ter acesso a todos esses dados de forma integrada e atualizada. Aqui estão algumas razões pelas quais as extrações automatizadas de dados são necessárias:

1. Aumento da eficiência: extrair manualmente dados de diferentes sistemas é um processo demorado e propenso a erros. Com extrações automatizadas, é possível obter dados rapidamente e com precisão, permitindo uma tomada de decisão mais rápida e eficiente;
2. Redução de erros: as extrações automatizadas removem a possibilidade de erros humanos, como digitação incorreta ou omissão de informações;
3. Integração de dados: as extrações automatizadas permitem integrar dados de diferentes sistemas, fornecendo uma visão mais completa e precisa do negócio; e
4. Tempo e recursos: automatizar o processo de extração de dados economiza tempo e recursos, permitindo que os funcionários se concentrem em tarefas mais importantes.

Em resumo, as extrações automatizadas de dados são fundamentais para obter dados precisos, integrados e atualizados, aumentando a eficiência e efetividade da tomada de decisão.

## Arquitetura de soluções de extração de dados

A arquitetura de soluções de extração de dados pode envolver o uso de APIs (Application Programming Interfaces), webhooks, planilhas, etc. para obter dados de diferentes fontes de forma programática. Neste tipo de solução, a extração de dados é automatizada usando scripts escritos em alguma linguagem de programção (como por exemplo, Python). De forma simples, podemos definir um processo de extração de dados em 5 partes:

1. Fonte de dados: a primeira etapa é identificar a fonte de dados desejada. Isso pode incluir bancos de dados, sistemas de gerenciamento de conteúdo (CMS), plataformas de e-commerce, entre outros;
2. API e Webhooks: a próxima etapa é acessar sua fonte de dados. A API é um conjunto de regras que permite que você interaja com a fonte de dados de forma programática, enquanto que os webhooks possibitam o recebimento assíncrono dos seus dados. Algumas fontes de dados possuem APIs públicas, enquanto outras exigem autenticação ou pagamento por volume de dados consumido;
3. Requisição API: uma vez que você tenha acesso à API, podemos enviar requisições para obter os dados desejados. Em Python, isso pode ser feito usando bibliotecas como Requests ou httplib2;
4. Tratamento de dados: após a recepção dos dados, precisamos tratá-los antes de serem usados. Isso pode incluir a limpeza dos dados, a conversão para um formato desejado e sua posterior análise. Em Python, podemos utilizar bibliotecas como Pandas ou Numpy para realizar tais tarefas.
5. Armazenamento de dados: por fim, os dados extraídos precisam ser armazenados em algum local para uso futuro. E, isso pode ser feito em bancos de dados locais ou na nuvem, como o Amazon S3 ou o Google Cloud Storage.

É importante destacar que a soma destas partes permite a automatização da extração de dados, tornando-a mais eficiente e confiável. Além disso, a utilização de Python como linguagem de programação permite a facilidade de uso e a ampla disponibilidade de bibliotecas para tratamento e armazenamento de dados.

## ETLs & ELTs

ETL, ELT e pipeline de dados são conceitos importantes na gestão de dados. E, podemos defini-los da seguinte forma:

1. ETL (Extração, Transformação e Carga): ETL é uma sigla que se refere ao processo de extração de dados de várias fontes, as transformações utilizadas para torná-los compatíveis com o sistema de destino e ao seu carregamento no sistema de destino. Esse processo é importante para garantir que os dados sejam integrados corretamente em um sistema único e possam posteriormente serem utilizados para a tomada de decisão informada.
2. ELT (Extração, Carga e Transformação): a ELT é semelhante ao ETL, mas a ordem dos passos é diferente. Em uma ELT, os dados são extraídos de fontes externas, carregados no sistema de destino e, em seguida, transformados para serem compatíveis com o sistema de destino. A ELT é uma abordagem popular quando se tem um hardware poderoso disponível para transformação de dados no lado do sistema de destino.
3. Pipeline de dados com extração ativa ou passiva: pipeline de dados se referem ao processo de fluxo contínuo de dados de uma fonte para um sistema de destino. Elas podem ser divididas em duas categorias:
   * Pipelines de extração ativa: são aquelas em que os dados são extraídos da fonte em intervalos regulares de tempo; e
   * Pipelines de extração passiva: são aquelas em que os dados são enviados assincronamente pela sistema originador da informação;

Em suma, ETL, ELT e pipeline de dados são conceitos importantes para a gestão de dados que ajudam a garantir que os dados sejam integrados corretamente em um único sistema e possam ser usados para tomar decisões informadas. A escolha entre ETL e ELT depende de fatores como a quantidade de dados e o poder de processamento disponível, enquanto que a escolha entre extração ativa ou passiva irá depender de outros fatores como a volumetria dos dados e o tempo de latência requisitado.

## Exemplos Básicos

Para tornarmos as coisas mais divertidas, vamos ver alguns exemplos simples de ETL (Extract, Transform, Load) e ELT (Extract, Load, Transform) em Python:

1. ETL

# Importar biblioteca para ler arquivos CSV

import csv

# Extrair dados de um arquivo CSV

with open("dados\_origem.csv", "r") as arquivo:

dados = list(csv.reader(arquivo))

# Transformar os dados lidos para a forma desejada

dados\_transformados = []

for linha in dados:

nome, idade = linha[0], int(linha[1])

dados\_transformados.append({"nome": nome, "idade": idade})

# Carregar os dados transformados em um banco de dados

import sqlite3

conexao = sqlite3.connect("dados\_destino.db")

cursor = conexao.cursor()

cursor.execute("CREATE TABLE IF NOT EXISTS pessoas (nome TEXT, idade INTEGER)")

for pessoa in dados\_transformados:

cursor.execute("INSERT INTO pessoas (nome, idade) VALUES (?, ?)", (pessoa["nome"], pessoa["idade"]))

conexao.commit()

conexao.close()

1. ELT

# Importar biblioteca para ler arquivos CSV

import csv

# Importar biblioteca para conexão com banco de dados

import sqlite3

# Extrair dados de um arquivo CSV e carregá-los diretamente no banco de dados

conexao = sqlite3.connect("dados\_destino.db")

cursor = conexao.cursor()

cursor.execute("CREATE TABLE IF NOT EXISTS pessoas (nome TEXT, idade INTEGER)")

with open("dados\_origem.csv", "r") as arquivo:

dados = list(csv.reader(arquivo))

for linha in dados:

cursor.execute("INSERT INTO pessoas (nome, idade) VALUES (?, ?)", linha)

conexao.commit()

conexao.close()

# Transformar os dados carregados no banco de dados

conexao = sqlite3.connect("dados\_destino.db")

cursor = conexao.cursor()

cursor.execute("UPDATE pessoas SET idade = idade + 1")

conexao.commit()

conexao.close()

Esses são exemplos simples que ilustram a diferença entre ETL e ELT, mostrando como a ordem das etapas pode afetar a eficiência e a escalabilidade desses processos. No ETL, primeiro extraímos os dados, depois os transformamos e, por fim, são persistidos no banco de dados.

## Instalando o Jupyter Notebook

Caso deseje montar um ambiente de estudo exclusivamente local, siga os passos a seguir para instalar o Jupyter Notebook na sua máquina:

1. Instale o Python em seu computador. Você pode baixá-lo no site oficial (<https://www.python.org/downloads/>). Verifique se a opção "Adicionar Python ao PATH" está selecionada durante a instalação.
2. Instale o gerenciador de pacotes pip. Ele vem instalado por padrão em versões recentes do Python, mas caso contrário, você pode baixá-lo no site (<https://pip.pypa.io/en/stable/installation/>).
3. Instale o Jupyter Notebook usando o pip. Abra o prompt de comando ou terminal e digite:

pip install jupyter

1. Verifique se a instalação foi bem-sucedida digitando:

jupyter notebook

Se tudo estiver correto, o Jupyter Notebook será aberto em seu navegador padrão.

1. (Opcional) Instale pacotes adicionais para ampliar as funcionalidades do Jupyter Notebook, como o NumPy, Pandas ou Matplotlib. Para isso, basta usar o pip novamente, por exemplo:

pip install numpy pandas matplotlib

Agora você está pronto para começar a usar o Jupyter Notebook! Crie um novo notebook e comece a programar já!

# Referências

* Artigo sobre Coleta de Dados: <https://towardsdatascience.com/the-art-of-data-engineering-1a6e0ba7f303>
* Tutorial sobre ETL com AWS Glue: <https://aws.amazon.com/pt/blogs/big-data/getting-started-with-aws-glue-data-quality-for-etl-pipelines/> e <https://medium.com/analytics-vidhya/etl-data-pipeline-in-aws-150acd6fee60>
* Documentação Oficial Python: <https://docs.python.org/pt-br/3/>
* Guia completo: <https://automatetheboringstuff.com/>
* Playlist Python no canal da Ada no Youtube: <https://www.youtube.com/watch?v=cDqMbI02hRs&list=PLcmVV8telDGzH6wFY_9h_KZ3Sn7EfjT1D>
* Livro: REIS, Joe & HOUSLEY, Matt. Fundamentals of Data Engineering, O'Reilly, 2022.
* Curso digital Ada: [Git e versionamento.](https://cursos.letscode.com.br/curso-digital/37f4b5d2-dbab-4c45-ab61-aac1ba2c7d19)