Tipos de dados: estruturados, semiestruturados e não estruturados

QXD0099 - Desenvolvimento de Software para Persistência

Universidade Federal do Ceará - *Campus* Quixadá

Prof. Francisco Victor da Silva Pinheiro
victorpinheiro@ufc.br

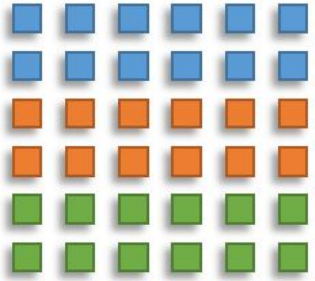
Slides baseados no material do Professor Régis Magalhães

Agenda

- Dados estruturados, semiestruturados e não estruturados
- Scraping Tools
- Extração de websites
- Extração a partir de PDF
- GED – Gestão Eletrônica de Documentos
  - Extração de dados de imagens
- Extração de documentos estruturados (JSON, XML)

Dados estruturados, semiestruturados e não estruturados

- **Dados estruturados**
  - Representados em um formato estrito.
  - Ex: registro em uma tabela relacional segue o mesmo formato dos outros registros da tabela.
  - Para dados estruturados, projeta-se cuidadosamente o esquema de banco de dados a fim de definir a estrutura do banco de dados.
  - O SGBD verifica se todos os dados seguem as estruturas e restrições especificadas no esquema.



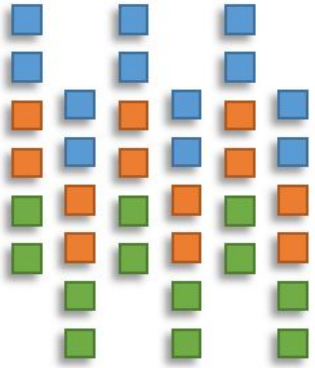
Dados estruturados, semiestruturados e não estruturados

Tabela: Clientes

ID_Cliente	Nome	E-mail	Telefone	Data_Cadastro
1	João Silva	joao.silva@gmail.com	(11) 91234-5678	2023-01-10
2	Maria Oliveira	maria.oliveira@gmail.com	(21) 98765-4321	2023-02-15
3	Ana Costa	ana.costa@gmail.com	(31) 95678-1234	2023-03-20

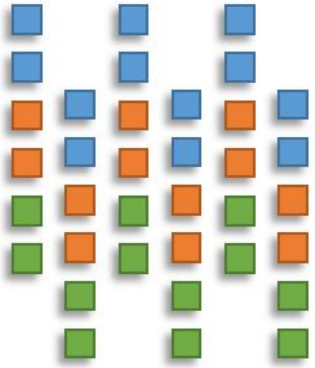
Dados estruturados, semiestruturados e não estruturados

- **Dados semiestruturados**
  - Os dados podem ter uma estrutura, mas nem toda a informação coletada terá a estrutura idêntica.
  - Alguns atributos podem ser compartilhados entre as diversas entidades, mas outros podem existir apenas em algumas entidades.
  - Atributos adicionais podem ser introduzidos em alguns dos itens de dados mais novos a qualquer momento, e não existe esquema predefinido.
  - Diversos modelos de dados foram introduzidos para representar dados semiestruturados, geralmente com base no uso de estruturas de dados de árvore ou grafo, em vez das estruturas do modelo relacional plano.



Dados estruturados, semiestruturados e não estruturados

- **Dados semiestruturados**
  - Nos dados semiestruturados, a informação do esquema é misturada com os valores dos dados, já que cada objeto de dados pode ter atributos diferentes que não são conhecidos antecipadamente.
  - Logo, esse tipo de dado às vezes é chamado de dados autodescritivos.
  - Os dados semiestruturados podem ser exibidos como um grafo direcionado.
  - Esse modelo é capaz de representar objetos complexos e estruturas aninhadas.



Dados estruturados, semiestruturados e não estruturados

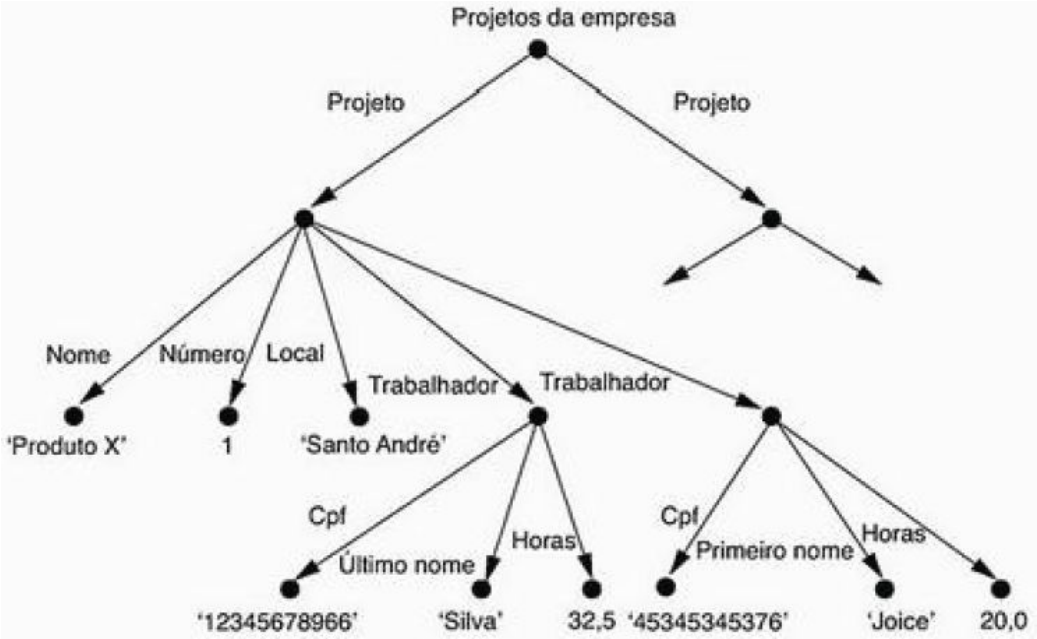
Exemplo em JSON

```
[
    {
      "id_cliente": 1,
      "nome": "João Silva",
      "email": "joao.silva@email.com",
      "pedidos": [
        {
          "id_pedido": 101,
          "data_pedido": "2023-01-12",
          "valor_total": 200.00,
          "status": "Entregue"
        }
      ]
    }
```

```
  {
      "id_cliente": 2,
      "nome": "Maria Oliveira",
      "email": "maria.oliveira@email.com",
      "pedidos": [
        {
          "id_pedido": 102,
          "data_pedido": "2023-02-20",
          "valor_total": 350.00,
          "status": "Em processamento"
        }
      ]
    }
]
```

Dados estruturados, semiestruturados e não estruturados

- **Dados semiestruturados representados como um grafo direcionado**



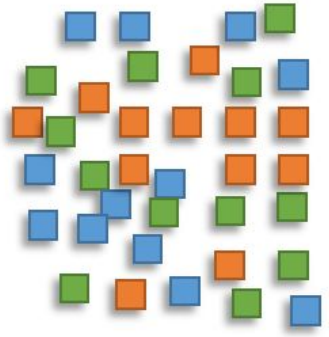
Dados estruturados, semiestruturados e não estruturados

- **Dados semiestruturados**
  - A informação do esquema (nomes de atributos, relacionamentos e classes) no modelo semiestruturado é misturado com os objetos e seus valores de dados na mesma estrutura de dados.

  - No modelo semiestruturado não existe requisito para um esquema predefinido ao qual os objetos de dados precisam se adequar, embora seja possível definir um esquema, se necessário.

Dados estruturados, semiestruturados e não estruturados

- **Dados não estruturados**
  - Além de dados estruturados e semiestruturados, existe uma terceira categoria, conhecida como dados não estruturados porque existe indicação muito limitada sobre o tipo de dados.
  - Ex: documento de texto com informações incorporadas a ele.
  - As páginas HTML que contêm alguns dados são consideradas dados não estruturados.
  - As tags HTML especificam informações, como parágrafos, níveis de cabeçalho nos documentos, etc.
  - Algumas tags oferecem estruturação de texto, como na especificação de lista numerada ou não numerada, ou de tabela.
  - Mesmo essas tags de estruturação especificam que os dados textuais devem ser exibidos de certa maneira, em vez de indicar o tipo de dado representado na tabela.



Dados estruturados, semiestruturados e não estruturados

Texto Livre (Postagem em Redes Sociais)

"Hoje foi um dia incrível! Visitei o parque com meus amigos e tiramos muitas fotos 📸. O pôr do sol estava lindo! #Natureza #Amigos #Felicidade"

Imagem

E-mail

De: maria.oliveira@email.com
Para: joao.silva@email.com
Assunto: Reunião de Projeto

Oi João,

Gostaria de confirmar nossa reunião de projeto para amanhã às 10h. Podemos nos encontrar na sala 5? Por favor, me avise se houver alguma mudança no horário.

Abraços,
Maria

Conversa em Aplicativo de Mensagens

08:50 - Maria: De nada! A propósito, já confirmou a reunião com o cliente para amanhã?
08:52 - João: Sim, está marcada para as 14h. Precisamos revisar os últimos pontos ainda hoje. Você acha que consegue participar?
08:55 - Maria: Claro, posso sim. Quer marcar para às 11h, depois do café?
08:57 - João: Perfeito! Vou preparar alguns tópicos para a nossa revisão. Mais alguma coisa que gostaria de discutir?
08:59 - Maria: Acho que só precisamos ver o orçamento detalhado. Ainda estão pendentes algumas atualizações.
09:01 - João: Verdade, estava esquecendo disso. Vou revisar a planilha de custos antes da nossa reunião.
09:03 - Maria: Combinado! Então nos vemos às 11h. Qualquer coisa, me chama antes.
09:05 - Maria: Obrigada! Até mais.

Scraping Tools - web

```python
from bs4 import BeautifulSoup
import requests

# Conexão com a página e extração do título
response = requests.get("https://quotes.toscrape.com/")
doc = BeautifulSoup(response.content, "html.parser")
title = doc.title.string

# Leitura de arquivo HTML local
with open("./input.html", encoding="utf-8") as file:
    doc = BeautifulSoup(file, "html.parser")

# Seleção de links e imagens com extensão .png
links = doc.select("a[href]")
pngs = doc.select("img[src$='.png']")

# Seleção do primeiro elemento com a classe 'masthead'
masthead = doc.select_one("div.masthead")

# Seleção de links de resultados
result_links = doc.select("h3.r > a")

# Exemplo de impressão do título
print("Título:", title)
```

- BeautifulSoup é uma biblioteca Python amplamente usada para extrair e manipular dados de arquivos HTML e XML.
- Ela é popular para web scraping, permitindo que desenvolvedores extraiam informações específicas de páginas da web ao navegar e analisar a estrutura do HTML.
- Esse código ele lê e extrai informações de uma página HTML.
- *pip install beautifulsoup4*
- *pip install requests*

Scraping Tools - PDF

```python
import tabula

# Extrai todas as tabelas de um PDF para uma lista de DataFrames
tables = tabula.read_pdf("example.pdf", pages="all")

# Exibe a primeira tabela
print(tables[0])
```

- Tabula-py
  - É uma interface Python para a biblioteca Tabula usada em Java.
  - É ideal para extrair tabelas de PDFs.
  - Para usá-la, você precisa ter o Java instalado.
- *pip install tabula-py*

Scraping Tools - PDF

```python
import tabula

# Caminho para o arquivo PDF
file_path = "./Relatorio_anual_dados.pdf"

# Extrai tabelas de todas as páginas e armazena como uma lista de DataFrames
tables = tabula.read_pdf(file_path, pages="all", multiple_tables=True)

# Exibe o conteúdo de cada tabela extraída
for i, table in enumerate(tables):
    print(f"Tabela {i+1}:")
    print(table)
    print("\n")

tabula.convert_into(file_path, "output.csv", output_format="csv",
pages="all")
```

- Pega os dados de uma tabela de um arquivo pdf, a tabela formatada e salva em um arquivo csv.

Scraping Tools - PDF

```python
from PyPDF2 import PdfReader

reader = PdfReader("example.pdf")
for page in reader.pages:
    print(page.extract_text())
```

- PyPDF2
  - Excelente para manipulação básica de PDFs, como extrair texto, dividir, combinar ou girar páginas.
  - Funciona bem com PDFs baseados em texto, mas não é ideal para PDFs com tabelas complexas ou gráficos.
  - *pip install PyPDF2*

Scraping Tools - PDF

```python
import pdfplumber

with pdfplumber.open("example.pdf") as pdf:
    for page in pdf.pages:
        print(page.extract_text())
```

- pdfplumber
  - É uma das melhores opções para extrair dados estruturados de PDFs, como tabelas e texto.
  - Tem suporte mais avançado para tratamento de PDFs complexos
- *pip install pdfplumber*

Scraping Tools - PDF

```python
from pdfminer.high_level import extract_text

text = extract_text("example.pdf")
print(text)
```

- pdfminer.six
  - Muito poderosa para extração de texto e análise detalhada do layout.
  - Permite extração de texto em blocos e reconhecimento de layouts complexos.

Scraping Tools - PDF

Comparação entre as bibliotecas

- **Tabula-py:** Melhor para PDFs que contêm tabelas.
- **PyPDF2:** Ideal para tarefas básicas, mas com limitações na extração de textos complexos.
- **pdfplumber:** Excelente para extração de texto e tabelas, com suporte robusto para PDFs complexos.
- **pdfminer.six:** Oferece mais controle sobre a estrutura e layout, sendo útil para PDFs complexos.
- Para tarefas principalmente com tabelas em PDFs, tabula-py ou pdfplumber são as melhores opções.
- Para outras tarefas básicas e manipulação, PyPDF2 é uma opção leve e fácil de usar.

GED – Gestão Eletrônica de Documentos

- O Tesseract OCR é uma biblioteca de Reconhecimento Óptico de Caracteres (OCR), usada para extrair texto de imagens e PDFs digitalizados.
- Originalmente desenvolvida pela HP, foi aberta como software de código aberto em 2005 e, de 2006 até 2018, teve continuidade com o suporte do Google.
- Principais Características do Tesseract OCR
  - **Reconhecimento de Texto em Vários Idiomas:** Suporta vários idiomas e pode ser treinado para reconhecer novos idiomas ou fontes específicas.
  - **Identificação de Textos em Imagens e PDFs Digitalizados:** É ideal para transformar conteúdo de imagens em texto editável, especialmente útil em sistemas de GED (Gestão Eletrônica de Documentos), onde documentos em papel precisam ser convertidos para formatos digitais pesquisáveis.
  - **Integração com Python e Outras Linguagens:** Existe uma biblioteca chamada pytesseract que permite usar o Tesseract facilmente com Python, simplificando a extração de texto em imagens e PDFs.

GED – Gestão Eletrônica de Documentos

- Para utilizar o Tesseract em Python, você pode instalar o ***pytesseract*** e o próprio Tesseract OCR:

```
pip install pytesseract
sudo apt-get install tesseract-ocr  # para Linux

# ou, no Windows, faça o download do executável do Tesseract e adicione-o ao PATH

# no colab
!apt-get update
!apt-get install -y tesseract-ocr
!pip install pytesseract
```

GED – Gestão Eletrônica de Documentos

- Este exemplo abre uma imagem (document_image.png) e usa o Tesseract para extrair o texto.

```
import pytesseract
from PIL import Image

# Carregar a imagem
image = Image.open("document_image.png")

# Extrair texto da imagem
text = pytesseract.image_to_string(image)
print("Texto extraído:", text)
```

GED – Gestão Eletrônica de Documentos

Aplicações Comuns

- **Digitalização de Documentos:** Convertendo documentos físicos para formatos digitais editáveis.
- **Indexação de Arquivos:** Criação de índices pesquisáveis em sistemas de GED.
- **Extração de Dados em Processos de Automação:** Automatizando processos que precisam extrair texto de documentos escaneados ou capturas de tela.
- https://github.com/tesseract-ocr/tesseract

Extração de Dados de JSON

- O formato JSON é muito comum e fácil de manipular em Python usando a biblioteca padrão json.

```python
import json

# Leitura do arquivo JSON
with open("dados.json", "r") as file:
    data = json.load(file)

# Exemplo de acesso aos dados
print(data["chave_principal"])  # substitua por uma chave real no JSON
```

Extração de dados de XML

```python
import xml.etree.ElementTree as ET

# Parse do arquivo XML
tree = ET.parse("./dados.xml")
root = tree.getroot()

# Acessando dados por tags
for cliente in root.findall("cliente"):
    nome = cliente.find("nome").text
    email = cliente.find("email").text
    telefone = cliente.find("telefone").text
    print(f"Nome: {nome}, Email: {email}, Telefone: {telefone}")

    # Acessando compras de cada cliente
    for compra in cliente.find("compras").findall("compra"):
        id_compra = compra.find("id_compra").text
        data = compra.find("data").text
        total = compra.find("total").text
        print(f"  Compra ID: {id_compra}, Data: {data}, Total: {total}")

        # Acessando itens de cada compra
        for item in compra.find("itens").findall("item"):
            produto = item.find("produto").text
            quantidade = item.find("quantidade").text
            preco_unitario = item.find("preco_unitario").text
            print(f"    Produto: {produto}, Quantidade: {quantidade}, Preço Unitário:
{preco_unitario}")
```

- Para XML, Python possui algumas bibliotecas populares, como xml.etree.ElementTree (padrão) e BeautifulSoup (com o parser lxml) para facilitar a navegação em estruturas complexas.

Extração de XML Complexo com BeautifulSoup e lxml

- Para arquivos XML mais complexos, BeautifulSoup com lxml pode oferecer mais flexibilidade:

```
from bs4 import BeautifulSoup

with open("dados.xml", "r") as file:
    soup = BeautifulSoup(file, "xml")  # o "xml" parser é específico para XML

# Buscando e extraindo dados
for elemento in soup.find_all("tag_principal"):
    valor = elemento.find("subtag").text
    print(valor)
```

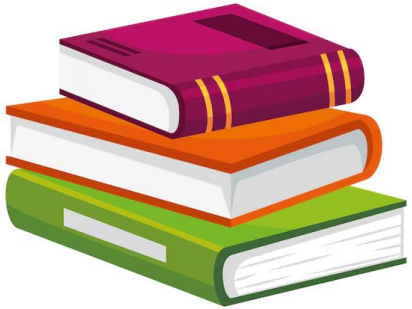
pip install lxml

Bibliografia Básica

- SADALAGE, P. J. E FOWLER, M. NoSQL Essencial. Editora Novatec, São Paulo, 2013.
- REDMOND, E.; WILSON, J. R. Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement. 1ª edição, 2012. The Pragmatic Programmers.
- ULLMAN, J.D.; WIDOW, J. First Course in Database Systems. 3a edição, 2007. Prentice Hall.
- HAMBRICK, G. et al. Persistence in the Enterprise: A Guide to Persistence Technologies; 1ª edição, 2008. IBM Press.
- ELMASRI, R.; NAVATHE, S. B. Sistemas de banco de dados. 4ª edicao, 2009. Pearson/Addison-Wesley.

Bibliografia Complementar

- WHITE, Tom. Hadoop: the definitive guide. California: O´Reilly, 2009. xix, 501 p. ISBN 9780596521974 (broch.).
- AMBLER, S.W., SADALAGE, P.J. Refactoring Databases: Evolutionary Database Design. 1a edição, 2011. Addison Wesley.
- SILBERSCHATZ, A.; SUDARSHAN, S. Sistema de banco de dados. 2006. Campus.
- LYNN, B. Use a cabeça! SQL. 1ª edição, 2008. ALTA BOOKS.
- SMITH, Ben. JSON básico: conheça o formato de dados preferido da web. São Paulo: Novatec, 2015. 400 p. ISBN 9788575224366 (broch.).
- HITZLER, P., KRÖTZSCH, M., and RUDOLPH, S. (2009). Foundations of Semantic Web Technologies. Chapman & Hall/CRC.
- ANTONIOU, G. and HARMELEN, F. (2008). A Semantic Web Primer. Second Edition,Cambridge, MIT Press, Massachusetts.
- HEATH, T. and BIZER, C. (2011). Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool, 1st edition.



Obrigado!

Dúvidas?

Universidade Federal do Ceará - *Campus* Quixadá

Prof. Francisco Victor da Silva Pinheiro
victorpinheiro@ufc.br