

# **PCS5787 – Ciência dos Dados**

## **Ferramentas de Big Data**

Representação de Dados e Arquitetura de Banco de dados não Relacionais

Prof. Dr. Pedro Luiz Pizzigatti Corrêa

pedro.correa@usp.br

Segundo semestre 2019

# Agenda

# Agenda

- Introdução
- Estruturas não normalizadas
- Estratégias de representação
- Arquiteturas de Aplicações e de Ferramentas de Big Data.



# Introdução

# Introdução

- Qual das imagens melhor representa um dado **normalizado** e um dado **não normalizado**?



Não normalizado



Normalizado

Adaptado de: MALAIKA, S.; NICOLA, M. **Data Normalization Reconsidered**. IBM. p. 40. 2012.

# Introdução



- Não normalizado:
  - Mundo ‘real’;
  - Mais fácil entender;
  - Fácil de armazenar;
  - Pronto para o uso.



- Normalizado:
  - Visão idealizada;
  - Mais difícil de entender;
  - O armazenamento implica em decomposição;
  - Não está pronto para o uso (recomposição dos dados).







# Agenda

- Introdução
- Estruturas não normalizadas
- Estratégias de representação
- Arquiteturas de Aplicações e de Ferramentas de Big Data.



# Estruturas não normalizadas

# Estruturas não normalizadas

- Dados em estruturas não normalizadas
  - Conceito de dados não normalizados:

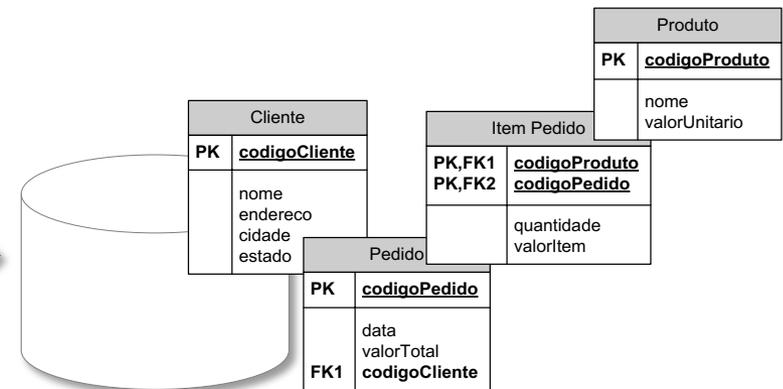
*“Abordagem na qual a normalização é omitida deliberadamente para construção de modelos cujos dados sejam representados de forma mais próxima à sua estrutura no mundo real.”*

Adaptado de: ELMASRI, R.; NAVATHE, S. B. **Fundamentals of Database Systems**. 7ª Edição, Pearson, 2015.

# Estruturas não normalizadas

- Dados em estruturas não normalizadas
  - Conceito de dados normalizados:

Data: ___ / ___ / ___ <b>PEDIDO</b> <input type="text"/>		
Cliente: _____		
Endereço: _____		
Cidade: _____		Estado: _____
CPF/CNPJ: _____		
QUANT.	PRODUTO	TOTAL R\$
ASSINATURA _____		TOTAL <input type="text"/>





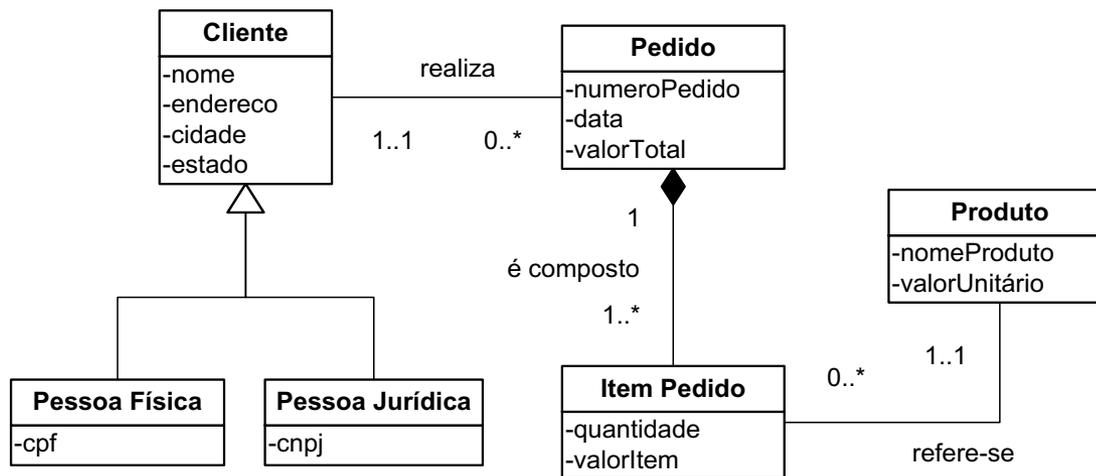




# Estruturas não normalizadas

- Dados em estruturas não normalizadas
  - Construção do Modelo de Agregados:

Modelo de Agregados



PEDIDO
Data Pedido
Numero Pedido
Valor Total
Nome Cliente
Endereço
Cidade
Estado
CPF / CNPJ
Quantidade de Itens
Item Pedido
Quantidade
Nome Produto
Valor Item

# Agenda

- Introdução
- Estruturas não normalizadas
- Estratégias de representação
- Arquiteturas de Aplicações e de Ferramentas de Big Data.



# Estratégias de representação

# Estratégias de representação

- Conceito de Modelo Canônico:
  - O modelo canônico refere-se à padronização de uma determinada **estrutura** de dados.

PEDIDO
Data Pedido
Numero Pedido
Valor Total
Nome Cliente
Endereço
Cidade
Estado
CPF / CNPJ
Quantidade de Itens
Item do Pedido
Quantidade
Nome Produto
Valor Item

Modelo Canônico

A estrutura está correta?

Quais são os atributos?

Qual é o formato dos dados?

Quais atributos são obrigatórios?

Quais são as restrições?

Adaptado de: *IBM Developers Work*. Disponível em <http://www.ibm.com/developerworks/data/library/techarticle/dm-0803sauter/>. Acesso em: 15 Julho 2019.

# Estratégias de representação

- Conceito de Modelo Canônico:

PEDIDO
Data Pedido
Numero Pedido
Valor Total
Nome Cliente
Endereço
Cidade
Estado
CPF / CNPJ
Quantidade de Itens
Item do Pedido
Quantidade
Nome Produto
Valor Item

O Modelo Canônico equivale a um  
**“Contrato de Serviço”**

Modelo Canônico – Pedido		
Atributo	Formato	Regra
Data Pedido	Data	(dd/mm/aaaa)
Numero Pedido	Inteiro (10)	Obrigatório
Valor Total	Moeda (13,2)	Obrigatório
Nome Cliente	Texto (60)	Obrigatório
...	...	...
Item do Pedido	Lista 100 produtos	
Quantidade	Inteiro	Obrigatório e maior que zero
Nome Produto	Texto (20)	Depende de um produto para existir
Valor Item	Moeda (13,2)	Obrigatório e maior que zero

# Estratégias de representação

- Estratégias de representação:
  - XML;
  - RDF;
  - JSON.

# Arquivos XML

# Estratégias de representação

- Arquivo XML (*eXtensible Markup Language*):
  - Linguagem de marcação recomendada pelo W3C para criação de documentos cujos dados estejam organizados de forma hierárquica.
  - Aplicações:
    - Banco de dados;
    - Interoperabilidade entre aplicações;
    - Utilização em WebServices;
    - Configuração de aplicativos;
    - Entre outros.



<http://www.w3.org/>

# Estratégias de representação

- Arquivo XML (*eXtensible Markup Language*):

```
<raiz>
  <nó_1.0> valor </nó_1.0>
  <nó_2.0> valor </nó_2.0>
  <nó_3.0>
    <nó_3.1.0> valor </nó_3.1.0>
    <nó_3.2.0>
      <nó_3.2.1> valor </nó_3.2.1>
      <nó_3.2.2> valor </nó_3.2.2>
    </nó_3.2.0>
  </nó_3.0>
</raiz>
```



<http://www.w3.org/>

# Estratégias de representação

- Arquivo XML (*eXtensible Markup Language*):
  - Legibilidade;
  - Flexibilidade (*tags* ilimitadas);
  - Autoexplicativa;
  - Formato hierárquico (um nó pode ter relações do tipo 1:N com outros nós).

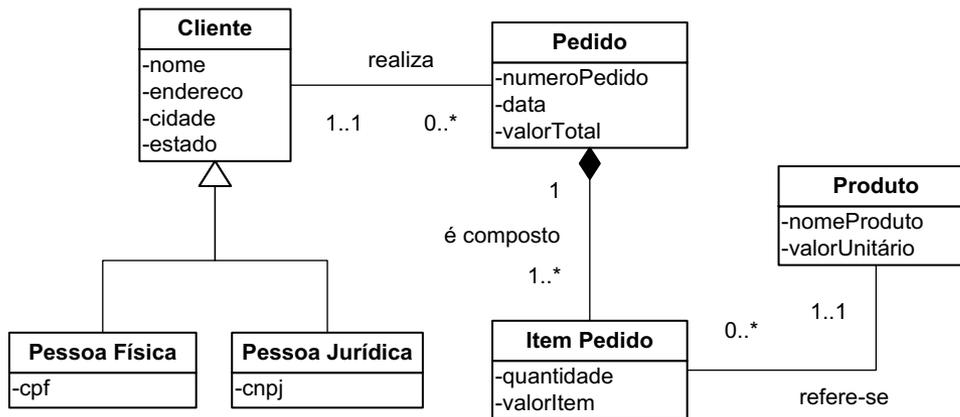


<http://www.w3.org/>

# Estratégias de representação

- Arquivo XML (*eXtensible Markup Language*):

## Modelo de Agregados



```
<pedido>
  <dataPedido> "21/12/2014" </dataPedido>
  <numeroPedido> 000001 </numeroPedido>
  <valorTotal> 224,59 </valorTotal>
  <nomeCliente> "João da Silva" </nomeCliente>
  <endereco> "Rua Trinta de Setembro, 20" </endereco>
  <cidade> "São Paulo" </cidade>
  <estado> "SP" </estado>
  <cpf> "490.005.340-60" </cpf>
  <cnpj> "" </cnpj>
  <qtdItens> 2 </qtdItens>
  <itemPedido>
    <produto>
      <codigoProduto> 0001 </codigoProduto>
      <quantidade> 1 </quantidade>
      <nomeProduto> "Calca Jeans" </nomeProduto>
      <valorItem> 134,60 </valorItem>
    </produto>
    <produto>
      <codigoProduto> 0002 </codigoProduto>
      <quantidade> 1 </quantidade>
      <nomeProduto> "Camiseta Preta" </nomeProduto>
      <valorItem> 89,99 </valorItem>
    </produto>
  </itemPedido>
</pedido>
```

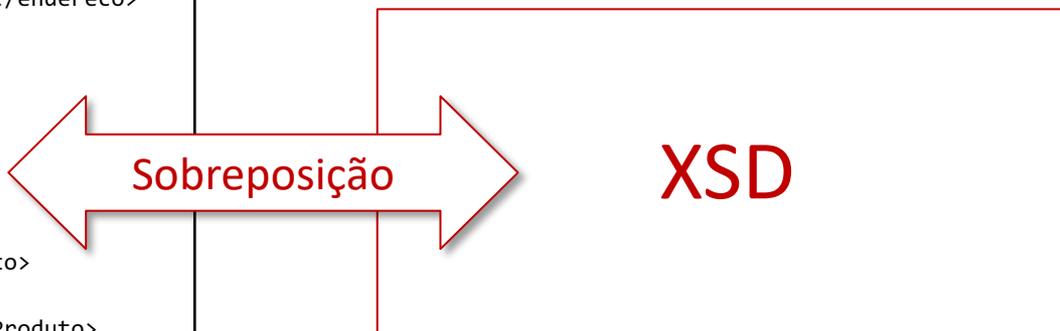
# Estratégias de representação

- Arquivo XSD (***XML Schema Definition***):
  - Permite a definição da estrutura dos dados usados em documentos XML:
    - Define quais são os nós;
    - Determina a hierarquia dos nós;
    - Especifica os tipos de valores para cada nó;
    - Entre outros.

# Estratégias de representação

- Arquivo XSD (*XML Schema Definition*):

```
<pedido>
  <dataPedido> "21/12/2014" </dataPedido>
  <numeroPedido> 000001 </numeroPedido>
  <valorTotal> 224,59 </valorTotal>
  <nomeCliente> "João da Silva" </nomeCliente>
  <endereco> "Rua Trinta de Setembro, 20" </endereco>
  <cidade> "São Paulo" </cidade>
  <estado> "SP" </estado>
  <cpf> "490.005.340-60" </cpf>
  <cnpj> "" </cnpj>
  <qtdItens> 2 </qtdItens>
  <itemPedido>
    <produto>
      <codigoProduto> 0001 </codigoProduto>
      <quantidade> 1 </quantidade>
      <nomeProduto> "Calca Jeans" </nomeProduto>
      <valorItem> 134,60 </valorItem>
    </produto>
    <produto>
      <codigoProduto> 0002 </codigoProduto>
      <quantidade> 1 </quantidade>
      <nomeProduto> "Camiseta Preta" </nomeProduto>
      <valorItem> 89,99 </valorItem>
    </produto>
  </itemPedido>
</pedido>
```



# Estratégias de representação

- Arquivo XSD (*XML Schema Definition*):

```
<pedido>
  <dataPedido> "21/12/2014" </dataPedido>
  <numeroPedido> 000001 </numeroPedido>
  <valorTotal> 224,59 </valorTotal>
  <nomeCliente> "João da Silva" </nomeCliente>
  <endereco> "Rua Trinta de Setembro, 20" </endereco>
  <cidade> "São Paulo" </cidade>
  <estado> "SP" </estado>
  <cpf> "490.005.340-60" </cpf>
  <cnj> "" </cnj>
  <qtdItens> 2 </qtdItens>
  <itemPedido>
    <produto>
      <codigoProduto> 0001 </codigoProduto>
      <quantidade> 1 </quantidade>
      <nomeProduto> "Calca Jeans" </nomeProduto>
      <valorItem> 134,60 </valorItem>
    </produto>
    <produto>
      <codigoProduto> 0002 </codigoProduto>
      <quantidade> 1 </quantidade>
      <nomeProduto> "Camiseta Preta" </nomeProduto>
      <valorItem> 89,99 </valorItem>
    </produto>
  </itemPedido>
</pedido>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="pedido">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="dataPedido"/>
        <xs:element type="xs:integer" name="numeroPedido"/>
        <xs:element type="xs:float" name="valorTotal"/>
        <xs:element type="xs:string" name="nomeCliente"/>
        ...
        <xs:element type="xs:integer" name="qtdItens"/>
        <xs:element name="itemPedido">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="produto" maxOccurs="100" minOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:integer" name="codigoProduto"/>
                    <xs:element type="xs:integer" name="quantidade"/>
                    <xs:element type="xs:string" name="nomeProduto"/>
                    <xs:element type="xs:float" name="valorItem"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  ...
```

# Estratégias de representação

- Arquivo XSD (*XML Schema Definition*):

FREEFORMATTER.COM    HTTPS    FreeDataGenerator.com    Contact    f Like 2.3K    G+1 114

**Formatters**

- » JSON Formatter
- » HTML Formatter
- » XML Formatter
- » SQL Formatter

**Validators**

- » JSON Validator
- » HTML Validator
- » XML Validator - XSD
- » XSD Generator
- » XPath Tester
- » Credit Card Number Generator & Validator
- » Regular Expression Tester

**Encoders & Decoders**

- » Url Encoder & Decoder
- » Base 64 Encoder & Decoder
- » QR Code Generator

**Code Minifiers**

- » JavaScript Minifier
- » CSS Minifier

**Converters**

- » XSLT (XSL Transformer)
- » XML to JSON Converter
- » JSON to XML Converter
- » CSV to XML Converter
- » Epoch Timestamp To Date

### XSD/XML Schema Generator

Generates an XSD (XML Schema) from a XML file. Simply copy-paste!

The generator uses a Russian-Doll approach. The Russian-Doll elements within other elements are declared locally instead of globally.

The generator will not try to generate xs:enumeration since it's too complex. The generator will try to use a 'smart' approach to figure out the data type.

\*Files bigger than 1 meg will be formatted to a new window.

**Option 1: Copy-paste your XML document here**

```
<pedido>
  <dataPedido> "21/12/2014" </dataPedido>
  <numeroPedido> 000001 </numeroPedido>
  <valorTotal> 224.59 </valorTotal>
  <nomeCliente> "João da Silva" </nomeCliente>
  <endereco> "Rua Tinta do Sotombe, 20" </endereco>
```

**Option 2: Or type in the URL to your XML file**

**Force output to new window:**

**GENERATE XSD**

**Generated XSD:**

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="pedido">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="dataPedido"/>
        <xs:element type="xs:float" name="numeroPedido"/>
        <xs:element type="xs:string" name="valorTotal"/>
        <xs:element type="xs:string" name="nomeCliente"/>
        <xs:element type="xs:string" name="endereco"/>
        <xs:element type="xs:string" name="cidade"/>
        <xs:element type="xs:string" name="estado"/>
        <xs:element type="xs:string" name="cpf"/>
        <xs:element type="xs:string" name="cnpj"/>
        <xs:element type="xs:float" name="quantidadeItens"/>
        <xs:element name="itemPedido">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="produto" maxOccurs="unbounded" minOccurs="0"/>
```

<http://www.freeformatter.com/xsd-generator.html>

# Arquivos RDF

# Estratégias de representação

- Arquivo RDF (*Resource Description Framework*):
  - Linguagem para representação e compartilhamento de informações na Web;
  - Utiliza URI (*Uniform Resource Identifier*) e XML para representar dados no formato de triplas:

(sujeito : predicado : objeto)



<http://www.w3.org/>

*URI (Uniform Resource Identifier)* – é um identificador usado para determinar a localização de um recurso na Internet. Este recurso pode ser um documento (URL), pode ser um elemento (URN) ou ambos.

# Estratégias de representação

- Arquivo RDF (*Resource Description Framework*):
  - Linguagem para representação e compartilhamento de informações na Web;
  - Utiliza URI (*Uniform Resource Identifier*) e XML para representar dados no formato de triplas:

(sujeito : predicado : objeto)
  - As triplas podem ser representadas por grafos, de forma que o Sujeito e o Objeto sejam os nós, enquanto que o Predicado seja a conexão entre os nós.



<http://www.w3.org/>

# Estratégias de representação

- Arquivo RDF (*Resource Description Framework*):

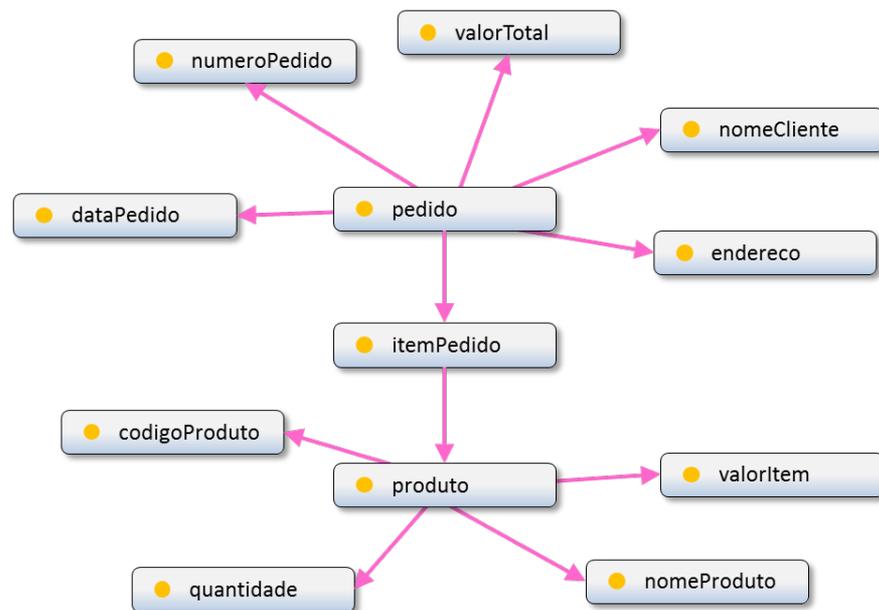
Data:    /    / <b>PEDIDO</b> <input type="text"/>		
Cliente: _____		
Endereço: _____		
Cidade: _____		Estado: _____
CPF/CNPJ: _____		
QUANT.	PRODUTO	TOTAL R\$
ASSINATURA _____		<b>TOTAL</b> <input type="text"/>

SUJEITO	PREDICADO	OBJETO
pedido	tem	numeroPedido
pedido	tem	dataPedido
pedido	tem	valorTotal
pedido	tem	nomeCliente
pedido	tem	enderecoCliente
...	...	...
pedido	tem	itemPedido
pedido	tem	qtdItens
produto	é	itemPedido
nomeCliente	temValor	"João da Silva"
dataPedido	temValor	21/12/2014
numeroPedido	temValor	0001
qtdItens	temValor	2
produto	tem	codigoProduto
codigoProduto	temValor	0001
produto	tem	nomeProduto
nomeProduto	temValor	"Calça Jeans"
...	...	...

# Estratégias de representação

- Arquivo RDF (*Resource Description Framework*):

SUJEITO	PREDICADO	OBJETO
pedido	tem	numeroPedido
pedido	tem	dataPedido
pedido	tem	valorTotal
pedido	tem	nomeCliente
pedido	tem	enderecoCliente
...	...	...
pedido	tem	itemPedido
pedido	tem	qtdItens
produto	é	itemPedido
nomeCliente	temValor	"João da Silva"
dataPedido	temValor	21/12/2014
numeroPedido	temValor	0001
qtdItens	temValor	2
produto	tem	codigoProduto
codigoProduto	temValor	0001
produto	tem	nomeProduto
nomeProduto	temValor	"Calça Jeans"
...	...	...



# Estratégias de representação

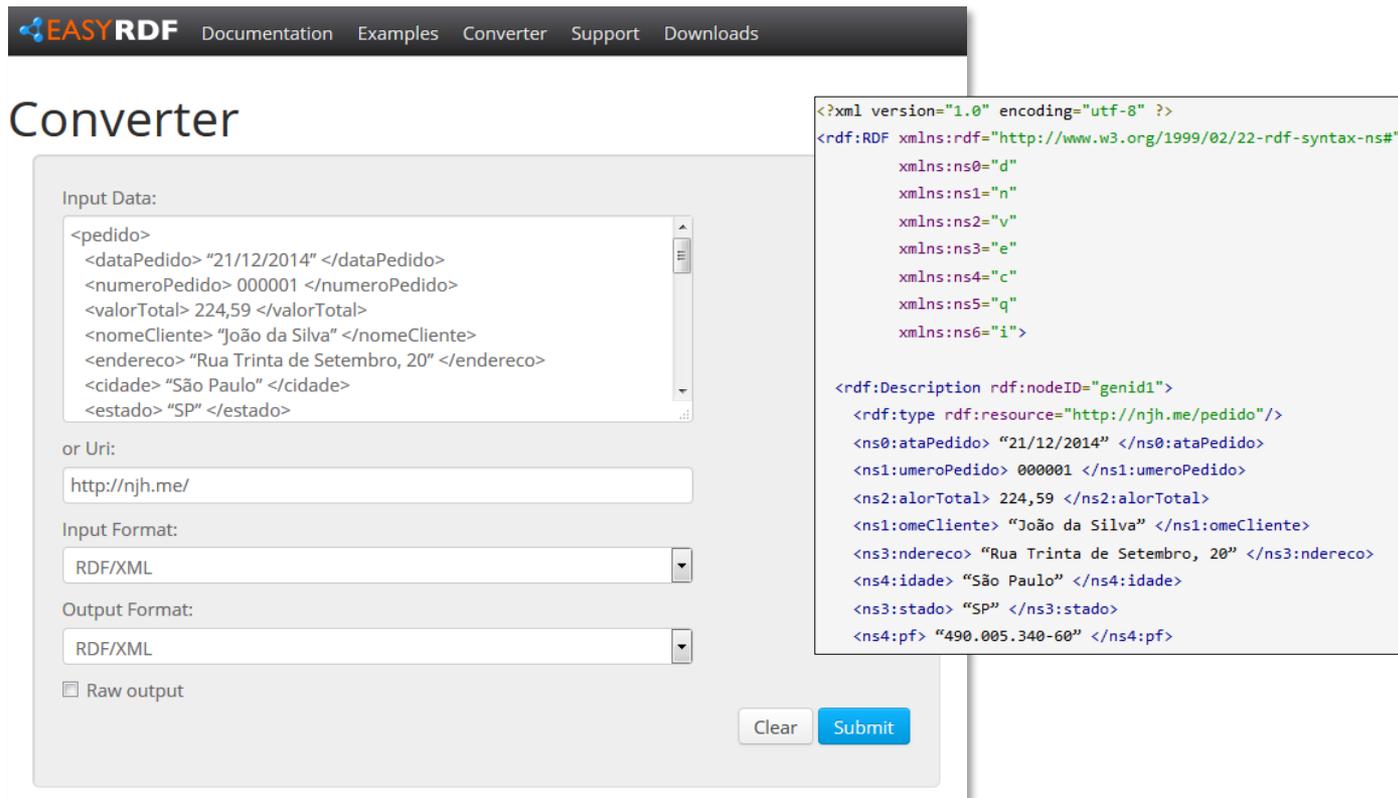
- Arquivo RDF (*Resource Description Framework*):

```
<pedido>
  <dataPedido> "21/12/2014" </dataPedido>
  <numeroPedido> 000001 </numeroPedido>
  <valorTotal> 224,59 </valorTotal>
  <nomeCliente> "João da Silva" </nomeCliente>
  <endereco> "Rua Trinta de Setembro, 20" </endereco>
  <cidade> "São Paulo" </cidade>
  <estado> "SP" </estado>
  <cpf> "490.005.340-60" </cpf>
  <cnpj> "" </cnpj>
  <qtdItens> 2 </qtdItens>
  <itemPedido>
    <produto>
      <codigoProduto> 0001 </codigoProduto>
      <quantidade> 1 </quantidade>
      <nomeProduto> "Calca Jeans" </nomeProduto>
      <valorItem> 134,60 </valorItem>
    </produto>
    <produto>
      <codigoProduto> 0002 </codigoProduto>
      <quantidade> 1 </quantidade>
      <nomeProduto> "Camiseta Preta" </nomeProduto>
      <valorItem> 89,99 </valorItem>
    </produto>
  </itemPedido>
</pedido>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ns0="dataPedido"
  xmlns:ns1="numeroPedido"
  xmlns:ns2="valorTotal"
  xmlns:ns3="nomeCliente"
  xmlns:ns4="endereco"
  ...
  xmlns:ns16="valorItem">
  <rdf:Description rdf:nodeID="genid1">
    <rdf:type rdf:resource="http://njh.me/pedido"/>
    <ns0:dataPedido> "21/12/2014" </ns0:dataPedido>
    <ns1:numeroPedido> 000001 </ns1:numeroPedido>
    <ns2:valorTotal> 224,59 </ns2:valorTotal>
    <ns3:nomeCliente> "João da Silva" </ns3:nomeCliente>
    <ns4:endereco> "Rua Trinta de Setembro, 20" </ns4:endereco>
    <ns5:cidade> "São Paulo" </ns5:cidade>
    <ns6:estado> "SP" </ns6:estado>
    <ns7:cpf> "490.005.340-60" </ns7:cpf>
    <ns8:cnpj> "" </ns8:cnpj>
    <ns9:qtdItens> 2 </ns9:qtdItens>
    <ns10:itemPedido>
      <rdf:Description>
        <rdf:type rdf:resource="http://njh.me/produto"/>
        ...
```

# Estratégias de representação

- Arquivo RDF (*Resource Description Framework*):



The screenshot displays the EASYRDF Converter web interface. The top navigation bar includes links for Documentation, Examples, Converter, Support, and Downloads. The main heading is "Converter".

**Input Data:**

```
<pedido>
<dataPedido> "21/12/2014" </dataPedido>
<numeroPedido> 000001 </numeroPedido>
<valorTotal> 224,59 </valorTotal>
<nomeCliente> "João da Silva" </nomeCliente>
<endereco> "Rua Trinta de Setembro, 20" </endereco>
<cidade> "São Paulo" </cidade>
<estado> "SP" </estado>
```

**or Uri:**

**Input Format:**

**Output Format:**

Raw output

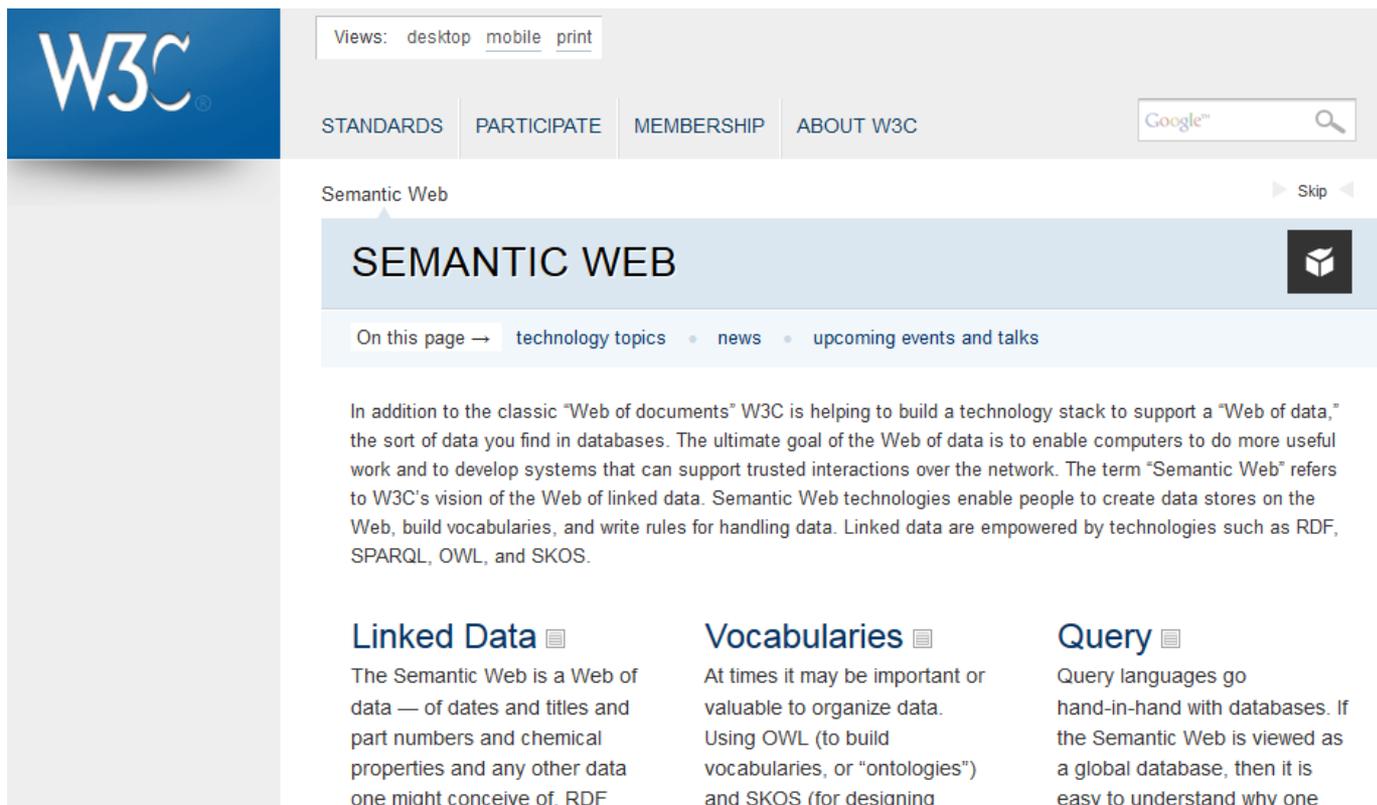
**Output (RDF/XML):**

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ns0="d"
  xmlns:ns1="n"
  xmlns:ns2="v"
  xmlns:ns3="e"
  xmlns:ns4="c"
  xmlns:ns5="q"
  xmlns:ns6="i">
  <rdf:Description rdf:nodeID="genid1">
    <rdf:type rdf:resource="http://njh.me/pedido"/>
    <ns0:ataPedido> "21/12/2014" </ns0:ataPedido>
    <ns1:umeroPedido> 000001 </ns1:umeroPedido>
    <ns2:alorTotal> 224,59 </ns2:alorTotal>
    <ns1:omeCliente> "João da Silva" </ns1:omeCliente>
    <ns3:ndereco> "Rua Trinta de Setembro, 20" </ns3:ndereco>
    <ns4:idade> "São Paulo" </ns4:idade>
    <ns3:stado> "SP" </ns3:stado>
    <ns4:pf> "490.005.340-60" </ns4:pf>
```

<http://www.easyrdf.org/convert>

# Estratégias de representação

- Arquivo RDF (*Resource Description Framework*):

A screenshot of the W3C Semantic Web page. The page features the W3C logo in the top left corner. Below the logo is a navigation menu with links for STANDARDS, PARTICIPATE, MEMBERSHIP, and ABOUT W3C. A search bar with the Google logo is located in the top right. The main content area is titled "SEMANTIC WEB" and includes a sub-navigation menu with links for "technology topics", "news", and "upcoming events and talks". The main text describes the Semantic Web as a "Web of data" and lists technologies like RDF, SPARQL, OWL, and SKOS. Three columns of text provide more details on "Linked Data", "Vocabularies", and "Query".

Views: desktop mobile print

STANDARDS PARTICIPATE MEMBERSHIP ABOUT W3C

Google™

Semantic Web Skip

## SEMANTIC WEB

On this page → technology topics • news • upcoming events and talks

In addition to the classic “Web of documents” W3C is helping to build a technology stack to support a “Web of data,” the sort of data you find in databases. The ultimate goal of the Web of data is to enable computers to do more useful work and to develop systems that can support trusted interactions over the network. The term “Semantic Web” refers to W3C’s vision of the Web of linked data. Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data. Linked data are empowered by technologies such as RDF, SPARQL, OWL, and SKOS.

### Linked Data

The Semantic Web is a Web of data — of dates and titles and part numbers and chemical properties and any other data one might conceive of. RDF

### Vocabularies

At times it may be important or valuable to organize data. Using OWL (to build vocabularies, or “ontologies”) and SKOS (for designing

### Query

Query languages go hand-in-hand with databases. If the Semantic Web is viewed as a global database, then it is easy to understand why one

<https://www.w3.org/standards/semanticweb/>

# Estratégias de representação

- Arquivo RDF (*Resource Description Framework*):
  - Flexibilidade;
  - Estrutura de grafos;
  - Interoperabilidade entre aplicações;
  - Não há separação entre dados e estruturas;
  - É mais complexo do que o XML puro;
  - Demanda mais recursos de máquina para manipulação.



<http://www.w3.org/>

# Arquivos JSON

# Estratégias de representação

- Arquivo JSON (*JavaScript Object Notation*):
  - Subconjunto da notação JavaScript, utilizado na serialização de dados entre sistemas computacionais;
  - É independente de linguagem, podendo ser analisado e processado por aplicações escritas em C, .NET, Java, entre outros;
  - Usado por APIs e Web Services para expor dados na Internet.



<http://www.json.org>

# Estratégias de representação

- Arquivo JSON (*JavaScript Object Notation*):
  - Estrutura básica:

```
“elemento_01” : valor
```

```
{  
  “elemento_01” : valor ,  
  “elemento_02” : valor ,  
  ...  
  “elemento_NN” : valor  
}
```



<http://www.json.org>

# Estratégias de representação

- Arquivo JSON (*JavaScript Object Notation*):
  - Estrutura básica:

```
{  
  "raiz" : {  
    "elemento_01" : valor ,  
    "elemento_02" : valor ,  
    ... ,  
    "elemento_NN" : valor  
  }  
}
```

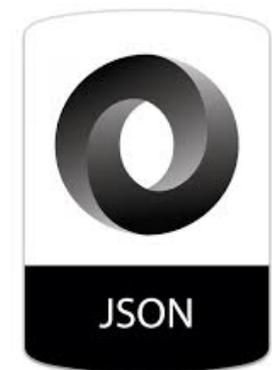


<http://www.json.org>

# Estratégias de representação

- Arquivo JSON (*JavaScript Object Notation*):
  - Estrutura básica:

```
{
  "raiz" : {
    "elemento_01" : valor ,
    "lista_01"    : [
      "subitem_01" : valor ,
      "subitem_02" : valor ,
      ...          ,
      "subitem_NN" : valor
    ]
  }
}
```



<http://www.json.org>

# Estratégias de representação

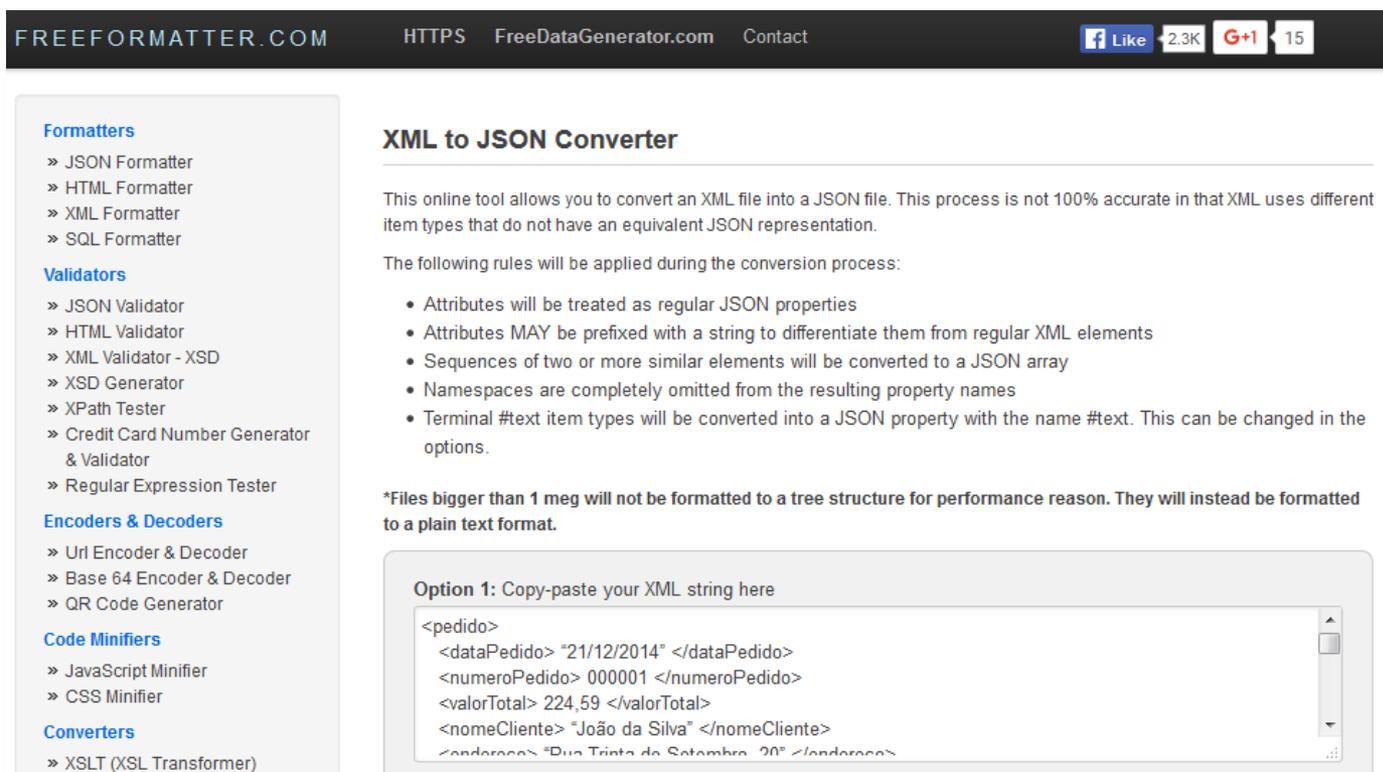
- Arquivo JSON (*JavaScript Object Notation*):

```
<pedido>
  <dataPedido> "21/12/2014" </dataPedido>
  <numeroPedido> 000001 </numeroPedido>
  <valorTotal> 224,59 </valorTotal>
  <nomeCliente> "João da Silva" </nomeCliente>
  <endereco> "Rua Trinta de Setembro, 20" </endereco>
  <cidade> "São Paulo" </cidade>
  <estado> "SP" </estado>
  <cpf> "490.005.340-60" </cpf>
  <cnpj> "" </cnpj>
  <qtdItens> 2 </qtdItens>
  <itemPedido>
    <produto>
      <codigoProduto> 0001 </codigoProduto>
      <quantidade> 1 </quantidade>
      <nomeProduto> "Calca Jeans" </nomeProduto>
      <valorItem> 134,60 </valorItem>
    </produto>
    <produto>
      <codigoProduto> 0002 </codigoProduto>
      <quantidade> 1 </quantidade>
      <nomeProduto> "Camiseta Preta" </nomeProduto>
      <valorItem> 89,99 </valorItem>
    </produto>
  </itemPedido>
</pedido>
```

```
{
  "dataPedido": "21/12/2014",
  "numeroPedido": "000001",
  "valorTotal": "224,59",
  "nomeCliente": "João da Silva",
  "endereco": "Rua Trinta de Setembro, 20",
  "cidade": "São Paulo",
  "estado": "SP",
  "cpf": "490.005.340-60",
  "cnpj": "",
  "qtdItens": "2",
  "itemPedido": [
    {
      "codigoProduto": "0001",
      "quantidade": "1",
      "nomeProduto": "Calca Jeans",
      "valorItem": "134,60"
    },
    {
      "codigoProduto": "0002",
      "quantidade": "1",
      "nomeProduto": "Camiseta Preta",
      "valorItem": "89,99"
    }
  ]
}
```

# Estratégias de representação

- Arquivo JSON (*JavaScript Object Notation*):



The screenshot shows the website FreeFormatter.com with a navigation bar at the top containing the URL, HTTPS, and social media icons. A sidebar on the left lists various tools under categories like Formatters, Validators, Encoders & Decoders, Code Minifiers, and Converters. The main content area is titled 'XML to JSON Converter' and includes a description of the tool, a list of conversion rules, and a text input field for XML data.

FreeFORMATTER.COM    HTTPS    FreeDataGenerator.com    Contact    f Like 2.3K    G+1    15

### Formatters

- » JSON Formatter
- » HTML Formatter
- » XML Formatter
- » SQL Formatter

### Validators

- » JSON Validator
- » HTML Validator
- » XML Validator - XSD
- » XSD Generator
- » XPath Tester
- » Credit Card Number Generator & Validator
- » Regular Expression Tester

### Encoders & Decoders

- » Url Encoder & Decoder
- » Base 64 Encoder & Decoder
- » QR Code Generator

### Code Minifiers

- » JavaScript Minifier
- » CSS Minifier

### Converters

- » XSLT (XSL Transformer)

## XML to JSON Converter

This online tool allows you to convert an XML file into a JSON file. This process is not 100% accurate in that XML uses different item types that do not have an equivalent JSON representation.

The following rules will be applied during the conversion process:

- Attributes will be treated as regular JSON properties
- Attributes MAY be prefixed with a string to differentiate them from regular XML elements
- Sequences of two or more similar elements will be converted to a JSON array
- Namespaces are completely omitted from the resulting property names
- Terminal #text item types will be converted into a JSON property with the name #text. This can be changed in the options.

\*Files bigger than 1 meg will not be formatted to a tree structure for performance reason. They will instead be formatted to a plain text format.

Option 1: Copy-paste your XML string here

```
<pedido>
  <dataPedido> "21/12/2014" </dataPedido>
  <numeroPedido> 000001 </numeroPedido>
  <valorTotal> 224,59 </valorTotal>
  <nomeCliente> "João da Silva" </nomeCliente>
  <endereco> "Rua Trinta de Setembro, 20" </endereco>
```

<http://www.freeformatter.com/xml-to-json-converter.html>

# Estratégias de representação

- Arquivo JSON (*JavaScript Object Notation*):

Formatted JSON:

```
"nomeCliente": "João da Silva",
"endereço": "Rua Trinta de Setembro, 20",
"cidade": "São Paulo",
"estado": "SP",
"cpf": "490.005.340-60",
"cnpj": "",
"qtdItens": "2",
"itemPedido": [
  {
    "codigoProduto": "0001",
    "quantidade": "1",
    "nomeProduto": "Calca Jeans",
    "valorItem": "134,60"
  },
  {

```

# Estratégias de representação

- Arquivo JSON (*JavaScript Object Notation*):

```
Viewer Text JSON Viewer
Paste Copy Format Remove white space Clear Load JSON data
{
  "pedido": {
    "dataPedido": "21/12/2014",
    "numeroPedido": "000001",
    "valorTotal": "224,59",
    "nomeCliente": "João da Silva",
    "endereco": "Rua Trinta de Setembro, 20",
    "cidade": "São Paulo",
    "estado": "SP",
    "cpf": "490.005.340-60",
    "cnpj": "",
    "qtdItens": "2",
    "itemPedido": [
      {
        "codigoProduto": "0001",
        "quantidade": "1",
        "nomeProduto": "Calca Jeans",
        "valorItem": "134,60"
      },
      {
        "codigoProduto": "0002",
        "quantidade": "1",
        "nomeProduto": "Camiseta Preta",
        "valorItem": "89.99"
      }
    ]
  }
}
```

<http://jsonviewer.stack.hu>

```
Viewer Text JSON Viewer
JSON
  pedido
    dataPedido : "21/12/2014"
    numeroPedido : "000001"
    valorTotal : "224,59"
    nomeCliente : "João da Silva"
    endereco : "Rua Trinta de Setembro, 20"
    cidade : "São Paulo"
    estado : "SP"
    cpf : "490.005.340-60"
    cnpj : ""
    qtdItens : "2"
    itemPedido
      0
        codigoProduto : "0001"
        quantidade : "1"
        nomeProduto : "Calca Jeans"
        valorItem : "134,60"
      1
        codigoProduto : "0002"
        quantidade : "1"
        nomeProduto : "Camiseta Preta"
        valorItem : "89.99"
```

# Estratégias de representação

- JSON Schema:
  - Assim como no XML, é possível definir qual a estrutura e o tipo dos dados usados em arquivos JSON:
    - Definição dos atributos;
    - Definição da ordem dos atributos;
    - Definição dos tipos de dados
    - Regras;
    - Entre outros.



<http://www.json.org>

# Estratégias de representação

- JSON Schema:

```
{
  "dataPedido": "21/12/2014",
  "numeroPedido": "000001",
  "valorTotal": "224,59",
  "nomeCliente": "João da Silva",
  "endereco": "Rua Trinta de Setembro, 20",
  "cidade": "São Paulo",
  "estado": "SP",
  "cpf": "490.005.340-60",
  "cnpj": "",
  "qtdItens": "2",
  "itemPedido": [
    {
      "codigoProduto": "0001",
      "quantidade": "1",
      "nomeProduto": "Calca Jeans",
      "valorItem": "134,60"
    },
    {
      "codigoProduto": "0002",
      "quantidade": "1",
      "nomeProduto": "Camiseta Preta",
      "valorItem": "89,99"
    }
  ]
}
```

```
{ "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "pedido": {
      "type": "object",
      "properties": {
        "dataPedido": { "type": "string" },
        "numeroPedido": { "type": "integer" },
        "valorTotal": { "type": "float" },
        "nomeCliente": { "type": "string" },
        "endereco": { "type": "string" },
        "cidade": { "type": "string" },
        "estado": { "type": "string" },
        "cpf": { "type": "string" },
        "cnpj": { "type": "string" },
        "qtdItens": { "type": "integer" },
        "itemPedido": { "type": "array",
          "items": { "type": "object", "properties": {
            { "codigoProduto": { "type": "integer" },
              "quantidade": { "type": "integer" },
              "nomeProduto": { "type": "string" },
              "valorItem": { "type": "float" }
            }, "required": [ "codigoProduto", "quantidade",
              "nomeProduto", "valorItem" ] } }},
        "required": ["dataPedido", "numeroPedido", "valorTotal",
          "nomeCliente", "endereco", "cidade", "estado",
          "cpf", "cnpj", "qtdItens", "itemPedido" ]
      }, "required": [ "pedido" ]
    }
  }
}
```

# Estratégias de representação

- JSON Schema:

JSONSchema.net Home About Contact Resources Previous Version [\\* Previous Version](#)

### JSON

URL

JSON 

```
{
  "pedido": {
    "dataPedido": "21/12/2014",
    "numeroPedido": "000001",
    "valorTotal": "224,59",
    "nomeCliente": "João da Silva",
    "endereco": "Rua Trinta de
Setembro, 20",
    "cidade": "São Paulo",
    "estado": "SP",
    "cpf": "490.005.340-60",
    "cnpj": "",
    "qtdItens": "2",
```

Well done! You provided valid JSON.

[Generate Schema](#) [Reset](#)

### Schema

Edit view has been disabled while bugs are fixed. Code and String views work as normal.  
Thank you for your patience.

[Code View](#) [Edit View](#) [String View](#)

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "pedido": {
      "type": "object",
      "properties": {
        "dataPedido": {
          "type": "string"
        },
        "numeroPedido": {
          "type": "string"
        },
        "valorTotal": {
          "type": "string"
        }
      }
    }
  }
}
```

<http://jsonschema.net>

# Agenda

- Introdução
- Estruturas não normalizadas
- Estratégias de representação
- Arquiteturas de Aplicações e de Ferramentas de Big Data.



# Arquitetura de Big Data

# Projeto Hadoop

2003

## The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung  
Google\*



2004

## MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat  
jeff@google.com, sanjay@google.com  
*Google, Inc.*



2006

## Bigtable: A Distributed Storage System for Structured Data

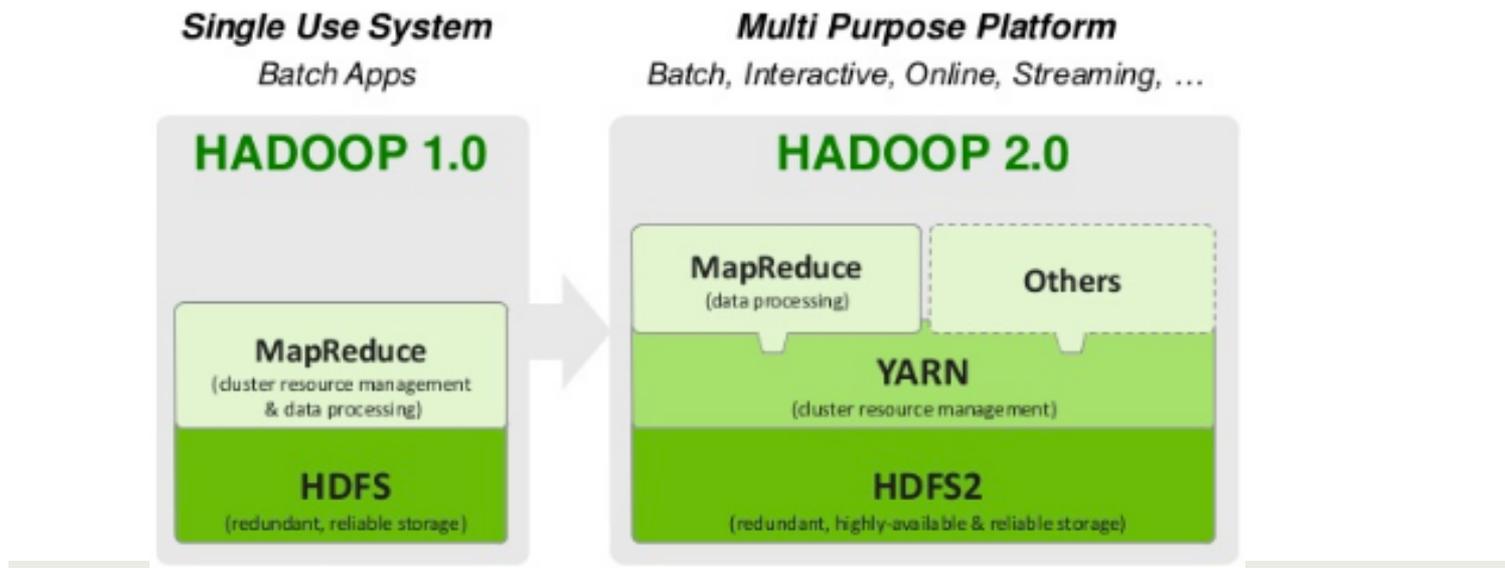
Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach  
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber  
{fay,jeff,sanjay,wilson,hkerr,m.7b,tushar,fikes,gruber}@google.com  
*Google, Inc.*

**Abstract**  
Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large  
achieved scalability and high performance, but Bigtable provides a different interface than such systems. Bigtable does not support a full relational data model; instead,

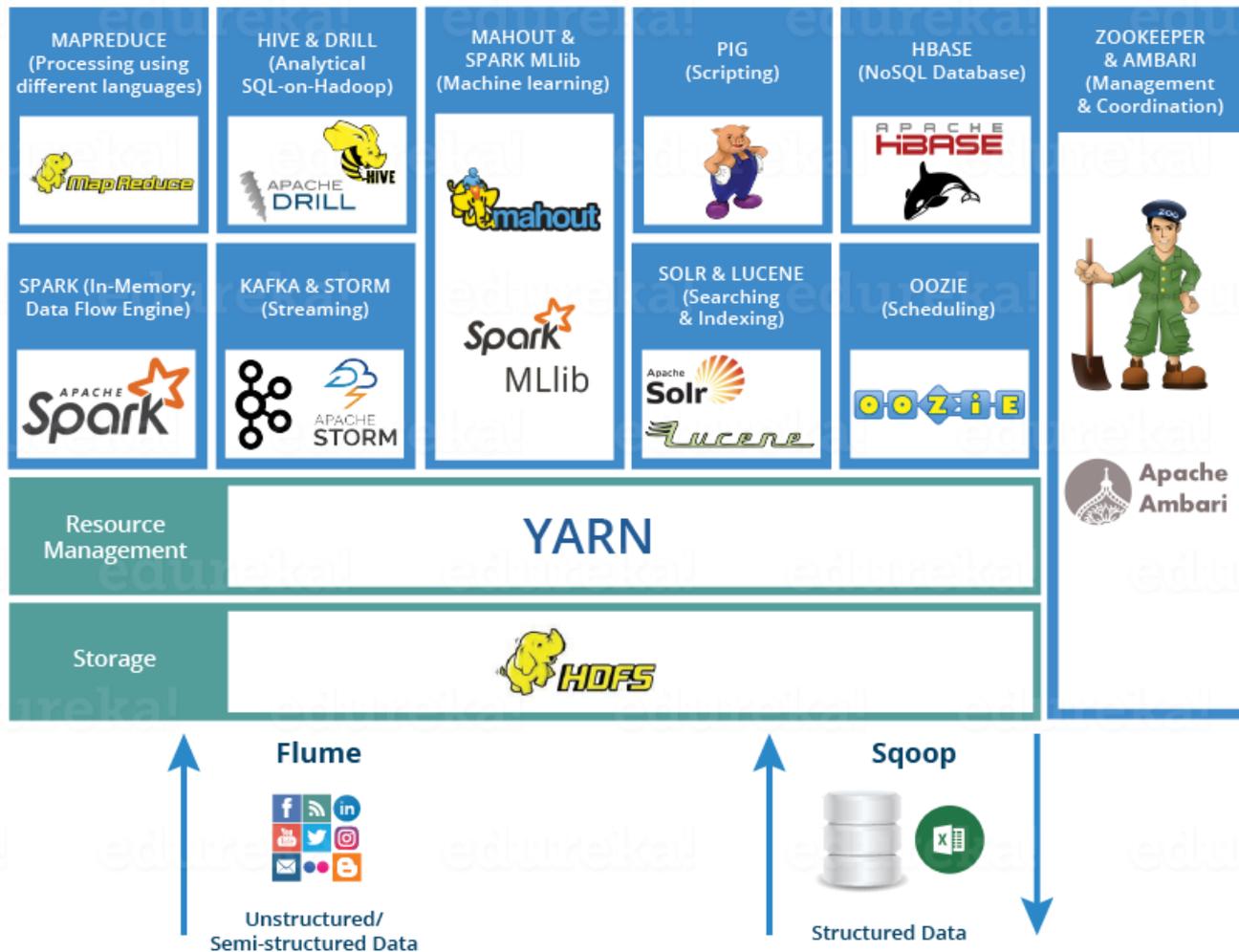


# Projeto Hadoop

- ▣ **MapReduce** é um paradigma de programação paralela para processamento distribuído proposto pelo Google.
- ▣ **MAP** é o processo de mapear/dividir a requisição
- ▣ **REDUCE** é o processo de agregação do resultado



# Arquitetura/Ecosystema Hadoop



# Distribuições Hadoop



# Agenda

- Introdução
- Estruturas não normalizadas
- Estratégias de representação
- Arquiteturas de Aplicações e de Ferramentas de Big Data.



# Aplicações Big Data.

# Exemplo de Aplicações

## Overview

- Tornar a estimativa de tempo de retorno/normalização no serviço de fornecimento de serviço mais preciso e específico para cada tipo de instalação ou tipo de equipamento.
- Priorizar os clientes através de determinadas regras para que, caso haja alguma falha no fornecimento, o tempo de retorno/normalização sejam comunicados de forma pró-ativa através de RPA (Robot Process Automation).

# Exemplo de Aplicações

- Para o tempo de retorno/normalização foram selecionados 5 (cinco) modelos de regressão;
- Para a priorização de clientes foram selecionados 2 (dois) modelos de scoring.

# Exemplo de Aplicações

## Lições aprendidas

- Aumentar janela de tempo para treinamento dos modelos;
- Aumentar a base de instalações binarizadas de acordo com as ocorrências (reduzir tempo de processamento);
- Notebook de score para re-treino automático de acordo com a performance do modelo em operação.

# Conceitos de Big Data

Representação de Dados e Arquitetura de Banco de dados não Relacionais

Prof. Dr. Pedro Luiz Pizzigatti Corrêa

[pedro.correa@usp.br](mailto:pedro.correa@usp.br)

Segundo semestre 2019