



# Representando Dados

- Antes da invenção do computador, a grande maioria dos dados eram representados em papel
- No papel, **texto**, **figuras**, **tabelas**, etc. são representados com tinta e cabe ao leitor identificar os diferentes elementos e entender o conteúdo
- No computador, precisamos saber de antemão com que **tipo de dados** estamos lidando para representá-los adequadamente

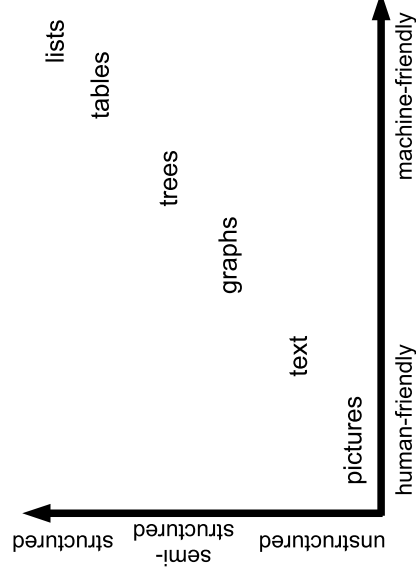
## Processando Dados

- Quanto mais “organizados” (estruturados) os dados, mas simples é o processamento
- Exemplo de dados **estruturados**: listas, tabelas, matrizes
- Exemplo de dados **não-estruturados**: texto, imagens, sons
- Exemplo de dados **semiestruturados**: árvores, grafos

## Principais modelos de dados

- Modelo Relacional (tabelas - foco do curso)
- Modelo hierárquico (árvores - foco desta aula)
- Modelo de grafos (veremos no fim do curso)

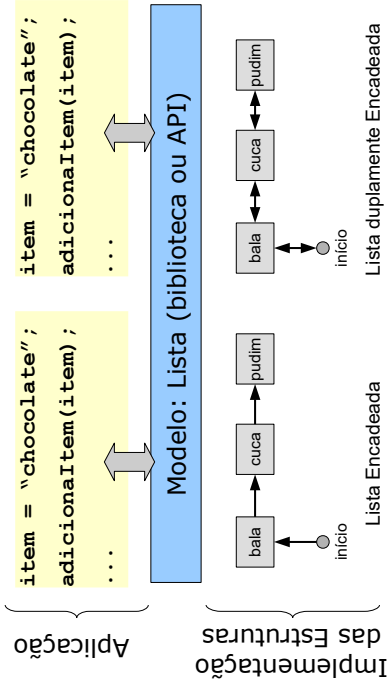
## Processando Dados



## Importância dos modelos

- Permitem a separação entre representação dos dados e a implementação física das estruturas
- Exemplo de ED1: um programador pode implementar um programa que usa uma lista encadeada e no futuro mudar a implementação das bibliotecas para uma lista duplamente encadeada sem precisar alterar o programa principal

## Importância dos modelos



## Modelo Hierárquico

- Usado para representar diversos tipos de dados
- Exemplos: Documentos, Categorias de Produtos, Modelos Orientados a Objeto, Disciplinas (tópicos, conteúdo, tarefas)
- Relativamente **flexível** e processamento relativamente **eficiente** (no meio do caminho entre tabelas e grafos)
- **XML** é o principal padrão para representação e compartilhamento de dados

## Importância dos modelos

- Em grandes empresas, os mecanismos de armazenamento mudam frequentemente para atender às demandas e isto não pode afetar as aplicações
- Exemplos: atualização de versão do SGBD, mudança de fornecedor de SGBD, upgrade de SGBD centralizado para SGBD distribuído

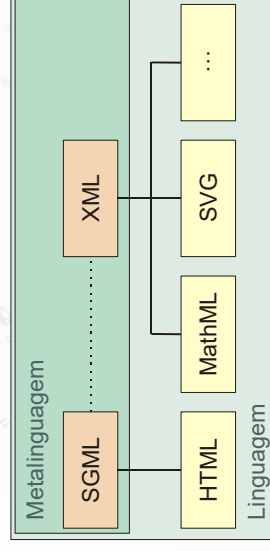
## XML - eXtensible Markup Language

## XML

- Lançada em 1996 como uma versão simplificada da SGML (*Standard Generalized Markup Language*), para ser utilizada na Web.

## Metalinguagem

- Tal como SGML, XML é uma metalinguagem.
- HTML ao contrário, foi escrita em SGML.



## Linguagem de Marcação

- Utiliza marcadores para agregar informações adicionais a documentos.
- Tomemos como exemplo a seguinte frase:  
Horácio escreveu o livro Vida dos Dinossauros.
- Desejamos agregar informações que identifiquem quem é o autor e qual a ação realizada.

## Estrutura Hierárquica

- Marcação podem ser agrupadas hierarquicamente.
- A interpretação de cada marcador está subordinada a seu contexto.

```
<sentença>  
  <autor>Horácio</autor>  
  <ação>escreveu o  
    <publicação>  
      <tipo>livro</tipo>  
      <título>Vida dos Dinossauros</título>  
    </publicação>  
  </ação>  
</sentença>
```

## Elementos e Atributos

- Atributos:

```
<autor cpf="487.526.548-74" nascimento="12/5/1960">  
  Horácio  
</autor>
```

- Elementos vazios:

```
<esgotado/>
```

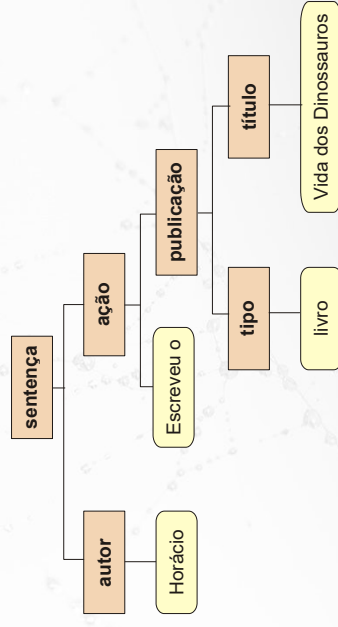
## Linguagem de Marcação

- Os marcadores se diferenciam do conteúdo pelos símbolos “<” e “>” (seguem o mesmo princípio de HTML):

```
<autor>Horácio</autor> <ação>escreveu o livro Vida dos Dinossauros</ação>
```

- Os marcadores delimitam unidades estruturais denominadas **elementos**.

## Modelo de Dados XML



## Validação de Documentos

- Documento bem formado:
  - atende às regras de construção XML
- Documento válido:
  - bem formado
  - atende a um esquema
    - DTD
    - XML Schema

DTD

- O documento XML pode se basear em uma gramática definida através de uma DTD (*Document Type Definition*).

```
<ELEMENT documento (topico+)>
<ELEMENT topico (titulo, subtopico*)>
<ELEMENT titulo (#PCDATA)>
<ELEMENT subtopico (titulo, #PCDATA)>
```

# Tipos Simples

```
<xs:element name="business">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="30"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

(Wilde, 2006)

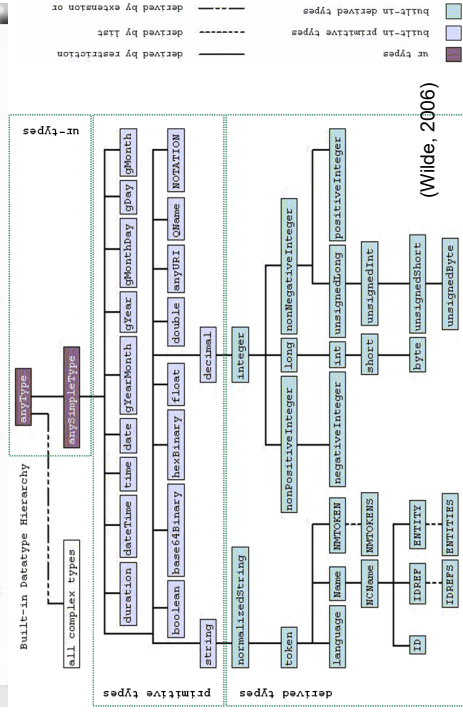
## Tipo Composto

```
<xs:schema>
  <xs:element name="billingAddress" type="addressType"/>
  <xs:element name="shippingAddress" type="addressType"/>
  <xs:complexType name="addressType">
    <xs:sequence>
      <xs:element name="street" type="xs:string"/>
      <xs:element name="city" type="xs:string"/>
      <xs:element name="state" type="xs:string"/>
      <xs:element name="zip" type="xs:decimal"/>
      <xs:sequence>
        <xs:attribute name="country" type="xs:NMTOKEN"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
```

(Wilde, 2006)

## XWL Schema

- Padrão para definição de esquemas XML
- Mais poderoso



## Query

- **XPath**
  - Especifica expressões na forma de caminhos que atendem padrões para alcançar nós específicos (elementos ou atributos)
- **XQuery**
  - Queries para XML (usam XPath)



## URI

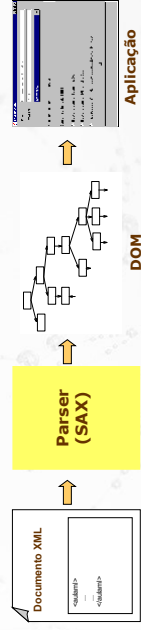
- A identificação de um recurso é feita através de um URI - Uniform Resource Identifier.
- URI = URL ou URN



## URI

- URL (*Uniform Resource Locator*): identifica recursos por meio de sua localização física na Internet.  
Ex.: <http://www.paleo.org>  
<ftp://ftp.unicamp.br>  
<mailto:horacio@paleo.org>
- URN (*Uniform Resource Names*): identificador é relacionado indiretamente com sua localização física na rede (exige um resolver).  
Ex.: <urn:ogc:def:uom:celsius>  
<urn:mpeg:mpeg21:dli:iswc:T-041.220.506-1>

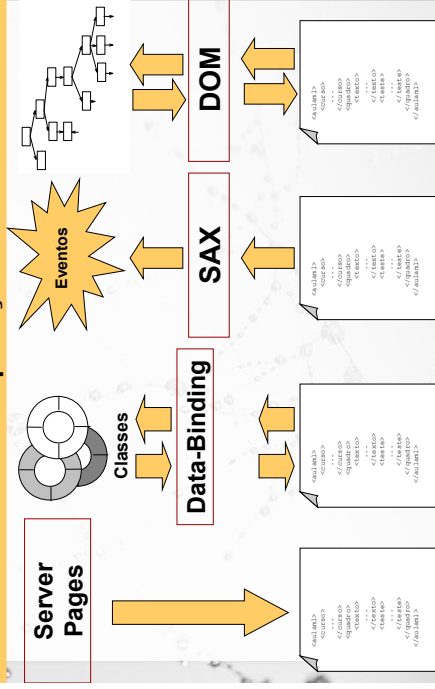
## Parser XML



## Introdução

Diversas tecnologias têm sido criadas para o processamento de documentos XML.

### Aplicação



## Introdução

Dentre estas tecnologias duas se destacaram e se tornaram referência:

- SAX - Simple API for XML
- DOM - Document Object Model



## SAX

- API baseada em eventos.
- Se tornou a mais estável API XML largamente utilizada [DOD01].
- Iniciou como uma solução para acesso a documentos XML por programas Java.
- Hoje tem sido portada para outras linguagens de programação, tal como: C++, Pascal, Perl, Python, etc.

## Eventos de conteúdo

```
public class SAXBasico extends  
    org.xml.sax.helpers.DefaultHandler  
{  
  
    public void startDocument() ...  
  
    public void startElement(...) ...  
  
    public void characters (...) ...  
  
    public void endElement(...) ...  
  
    public void endDocument() ...  
}
```

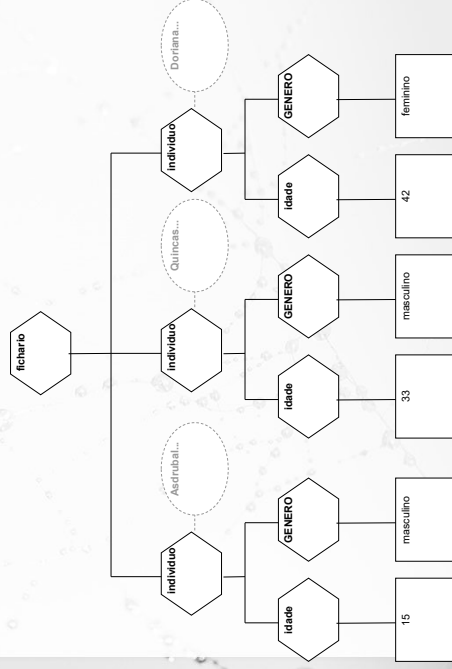
## Eventos de conteúdo

Método	Ação quando o parser encontra
<code>startDocument</code>	início do documento
<code>startElement</code>	início de um elemento
<code>characters</code>	conteúdo texto
<code>endElement</code>	final de um elemento
<code>endDocument</code>	final do documento

## DOM

- DOM define uma API para documentos XML e HTML.
- Ele acrescenta ao padrão destas linguagens toda a funcionalidade e flexibilidade que um programa precisa para acessar e manipular documentos.
- Definido em IDL, ECMAScript e Java.

## Document Object Model



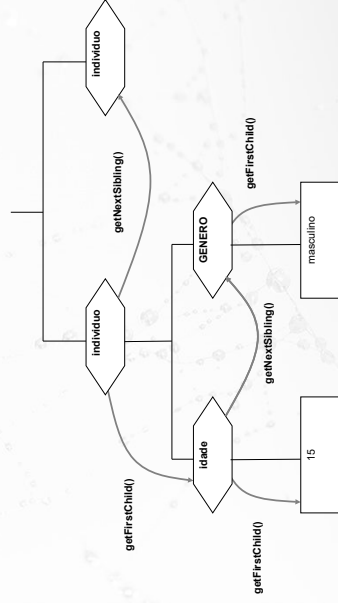
## DOM - Estudo de Caso

```
<FICHARIO>  
<indivuido nome="Asdrubal da Silva">  
  <idade>15</idade>  
  <genero>masculino</genero>  
</indivuido>  
<indivuido nome="Quincas Borba">  
  <idade>33</idade>  
  <genero>masculino</genero>  
</indivuido>  
<indivuido nome="Doriana Margarina">  
  <idade>42</idade>  
  <genero>feminino</genero>  
</indivuido>  
</FICHARIO>
```

## Processo



## Navegar pelo Documento



## JSON x XML

```

{"employees": [
  { "firstName": "John", "lastName": "Doe" },
  { "firstName": "Anna", "lastName": "Smith" },
  { "firstName": "Peter", "lastName": "Jones" }
]}
  
```

```

<employees>
<employee>
  <firstName>John</firstName>
</lastName>Doe</lastName>
</employee>
<employee>
  <firstName>Anna</firstName>
</lastName>Smith</lastName>
</employee>
</employees>
  
```

## Interfaces

- **Node** - esta interface representa genericamente qualquer nó da árvore.
- **Element** - acrescenta propriedades e métodos específicos de um nó do tipo elemento.
- **Document** - interface do nó raiz da árvore que representa o documento completo.
- **NodeList** - representa uma lista de nós. Pode representar, por exemplo, a lista de filhos de um nó.

## JSON

- Modelo de representação de dados semiestruturados
- Baseado em JavaScript com foco em aplicações Web
- Mais simples e mais leve que XML (mas também mais limitado)
- Não define linguagens (em definição)
- XML Nutella :)

## Exercícios

- Dê exemplos de dados para cada tipo de modelo (relacional, hierárquico, grafos)
- Quais as vantagens de se usar um modelo padronizado como o XML?

## Referências e Agradecimentos

- Diversos slides baseados no curso de BD do Prof. André Santanchè (UNICAMP)
  - Site: <http://www.ic.unicamp.br/~santanche>
  - Canal Youtube:  
<https://www.youtube.com/santanche>