

# Esquemas para Dados XML

- Definição da
  - Organização **hierárquica** do documento XML
  - **Conteúdo** de elementos simples e atributos
  - **Cardinalidade** de sub-elementos
- Documento XML **válido**
  - Respeita as regras de um esquema XML
  - Validado por um programa (*parser* de validação)
- Recomendações da W3C
  - **DTD** (*Document Type Definition*) - 1998
  - **XML Schema** - 2001

# DTD (*Document Type Definition*)

- Gramática Regular Proprietária
  - Definição **recursiva** do modelo de conteúdo de elementos XML  
 $E1(E2+, \dots) \dots E2(E3, \dots) \dots E3(\textit{conteúdoTexto})$
  - **Modelo de conteúdo** de um elemento XML pode ser formado por
    - **Seqüência** ordenada de sub-elementos
    - Alternativas de sub-elementos e/ou conteúdo (**Escolha**)
    - Algumas **restrições de valor** para elementos simples e atributos

# DTD - Definição

- Sintaxe de Declaração
  - `<! ... >`
- Declaração de Elemento
  - `<!ELEMENT ... >`
- Declaração de Atributos de um Elemento
  - `<!ATTLIST ... >`

# DTD – Definição de Elementos

- Elemento complexo

- Modelo de conteúdo **Seqüência** (*sequence*)

- `<!ELEMENT autor (nome, email)>`

- Modelo de conteúdo **Escolha** (*choice*)

- `<!ELEMENT sexo (M|F)>`

- Seqüência e escolha podem ser **combinados**

- `<!ELEMENT pessoa (nome, (empresa|universidade))>`

- Elemento simples

- `<!ELEMENT titulo (#PCDATA)>`

*Parsable Character DATA*



*conteúdo textual validado pelo Parser*

*(caracteres válidos: pertencem ao Encoding definido; não inclui caracteres reservados)*

# DTD – Definição de Elementos

- Restrições de Cardinalidade no Modelo de Conteúdo de Elementos Complexos
  - 0:1 (?)
    - `<!ELEMENT endereco(..., complemento?)>`
  - 1:1 (*default*)
    - `<!ELEMENT livro(titulo, ...)>`
  - 1:N (+)
    - `<!ELEMENT livro(..., capitulo+)>`
  - 0:N (\*)
    - `<!ELEMENT autor(..., email*)>`

# DTD – Elementos Especiais

- Elemento Vazio

- `<!ELEMENT figura EMPTY>`

- Elemento Misto

- `<!ELEMENT capitulo (#PCDATA | secao | figura) *>`

- Regra de definição deste elemento

- #PCDATA deve ser obrigatoriamente o primeiro *token* no modelo
    - o modelo de conteúdo deve ser uma escolha, opcional e com repetição

- Elemento do tipo ANY

- `<!ELEMENT observacao ANY>`

- Admite conteúdo livre (com ou sem *tags*)

- Pouco utilizado, devido à ausência de restrições

# DTD – Definição de Atributos

- Definição: `<!ATTLIST nome_elem def_at(s)>`
- Principais tipos de dados
  - **CDATA**
    - conteúdo textual não validado pelo *parser*
  - **ID**
    - identificador do elemento no documento XML
  - **IDREF (S)**
    - referência(s) a atributo(s) do tipo ID
- Exemplo
  - `<!ATTLIST livro ISBN ID ano CDATA>`

# DTD – Definição de Atributos

- Restrições de Integridade e Conteúdo
  - Conteúdo obrigatório/opcional
    - **#REQUIRED/#IMPLIED**
  - Enumeração
    - (valor 1, ..., valor n) "valor\_default"
  - Valor fixo
    - **#FIXED**
  - Valor *default*
    - Define uma *string* como valor *default*
- Exemplo
  - ```
<!ATTLIST autor codigo ID sexo CDATA (M,F) "M"
#IMPLIED nacionalidade CDATA #FIXED "brasileira">
```



# DTD – Definição de Entidades

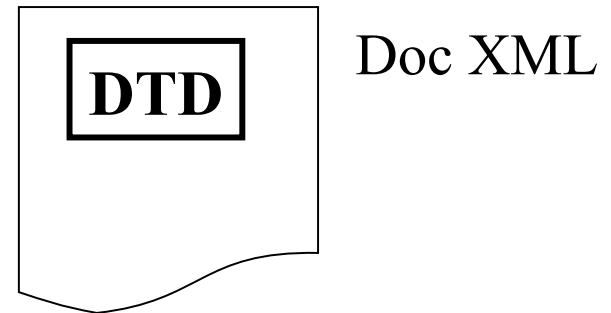
- `<!ENTITY fln "Florianópolis">`
- Exemplo de Utilização no Documento XML

```
...  
<Endereço>rua X, 111 &fln CEP 88000-000</Endereço>  
...  
<Cidade>&fln</Cidade>  
...
```

# Vinculação DTD – Documento XML

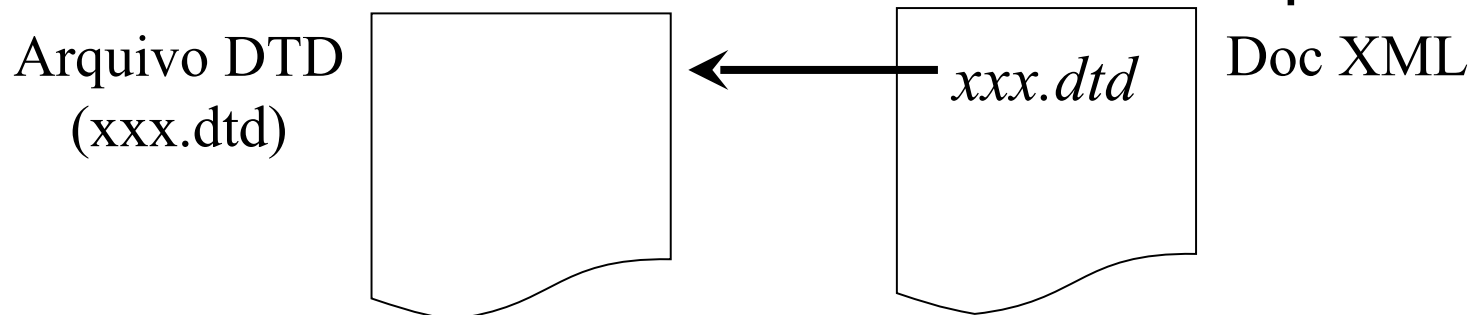
- Definição Interna

- DTD definida dentro do documento XML



- Definição Externa

- DTD armazenada em um arquivo próprio (**.dtd**)
- Documento XML faz referência a este arquivo



# Vinculação DTD – Documento XML

## Exemplo de Definição Interna

```
<?xml version="1.0"?>
  <!DOCTYPE artigo [
    <!ELEMENT artigo (titulo, autor)>
    <!ELEMENT titulo (#PCDATA)>
    <!ELEMENT autor (nome)>
    <!ELEMENT nome (#PCDATA)>
  ]>
<artigo>
  <titulo> ... .. </titulo>
  <autor>
    <nome> ... .. </nome>
  </autor>
</artigo>
```

## Exemplo de Definição Externa

```
<?xml version="1.0"?>
  <!DOCTYPE artigo
    SYSTEM "artigo.dtd">
<artigo>
  ...
</artigo>
```

**artigo.dtd**

```
<!ELEMENT artigo (titulo, autor)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT autor (nome)>
<!ELEMENT nome (#PCDATA)>
```

# Exercício

Definir uma DTD  
para este doc  
XML

```
<?xml version="1.0" encoding="UTF-8"?>
<listaLivros>
<livro ISBN="112">
  <titulo>Tecnologia XML</titulo>
  <preco>79.00</preco>
  <autor>
    <nome>Joao da Silva</nome>
    <mail>jjs@hotmail.com</mail>
    <mail>jsilva@xxx.com</mail>
  </autor>
  <autor>
    <nome>Maria Souza</nome>
  </autor>
  <editora>Campus</editora>
</livro>
<livro ISBN="72">
  <titulo>Banco de Dados</titulo>
  <preco>129.00</preco>
  <autor>
    <nome>Pedro Santos</nome>
  </autor>
  <editora>Makron Books</editora>
</livro>
<livro ISBN="239">
  <titulo>Sistemas de Informacao</titulo>
  <preco>208.50</preco>
  <autor>
    <nome>Ana Pereira</nome>
    <mail>anap@zzz.com</mail>
  </autor>
  <editora>Campus</editora>
</livro>
<livro ISBN="243">
  <titulo>ERBD 2010</titulo>
  <preco>38.50</preco>
  <evento>
    <local>
      <cidade>Joinville</cidade>
      <estado>SC</estado>
    </local>
    <dataInicio>09-04-2007</dataInicio>
    <dataTermino>11-04-2007</dataTermino>
  </evento>
  <editora>Editora da UCS</editora>
</livro>
...
</listaLivros>
```

# XML Schema

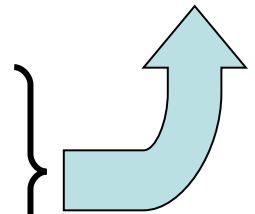
- **Sintaxe XML** para definição de esquemas
  - Esquema pode ser validado por um *parser*
- Maior número de recursos que a DTD

- Definição de **tipos de dados**
  - simples (*integer, boolean, string, ...*)
  - especiais (*list, union*)

Vantagem:  
reusabilidade!

Conceitos  
da OO

- Definição de **tipos abstratos**
  - simples e complexos
- **Especialização** de tipos abstratos e elementos
- Outras formas de definição de **restrições de integridade**
  - intervalos de valores permitidos, expressões regulares, ...
- Suporte a **namespaces**
  - vocabulários específicos para certos domínios



# Declaração de Tipos Abstratos

- Definem a **estrutura de elementos e atributos**
  - Podem ser **reutilizados**, ou seja, podem servir para a definição de outros tipos e de vários elementos ou atributos
- **Classificam-se em**
  - **Complexos**
    - definem a estrutura de elementos complexos
  - **Simples**
    - definem a estrutura de elementos simples e atributos

# Definição de Tipo Complexo

- Sintaxe: `<complexType>...</complexType>`
  - elementos complexos só podem ser definidos através de tipos complexos!
- Exemplo de Sintaxe1
  - definição de tipo e sua vinculação c/ elementos

```
<xs:complexType name="tPessoa">
  <xs:sequence>
    <xs:element name="nome" type="xs:string"/>
    <xs:element name="fone" type="xs:integer"/>
    <xs:element name="email" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
...
<xs:element name="funcionario" type="tPessoa"/>
<xs:element name="cliente" type="tPessoa"/>
```

# Definição de Tipo Complexo

- Exemplo de Sintaxe2
  - Definição direta de elemento complexo

```
<xs:element name="funcionario">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="fone" type="xs:integer"/>
      <xs:element name="email" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- Obs.: Principais tipos de dados simples pré-definidos da *XML Schema*

<code>xs:string</code>	<code>xs:byte</code>	<code>xs:integer</code>	<code>xs:boolean</code>
<code>xs:float</code>	<code>xs:double</code>	<code>xs:dateTime</code>	<code>xs:time</code>
<code>xs:date</code>	<code>xs:anyURI</code>	<code>xs:hexBinary</code>	



# Definição de Tipo Complexo

- Modelo de conteúdo de um `complexType`
  - Construtores de grupo `sequence`, `choice` e `all`
  - Exemplo de uso do construtor `choice`

```
<xs:complexType name="tPublic">
  <xs:sequence>
    <xs:element name="nome" type="xs:string"/>
    <xs:choice>
      <xs:element name="ISBN" type="xs:integer"/>
      <xs:element name="volume" type="xs:integer"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>

<xs:element name="publicacao" type="tPublic"/>
```

# Definição de Tipo Complexo

- Restrições de Cardinalidade
  - Cláusulas **minOccurs** e **maxOccurs**
- **minOccurs**
  - Número mínimo de ocorrências de um sub-elemento
  - Mínimo: 0
  - *Default*: 1
- **maxOccurs**
  - Número máximo de ocorrências de um sub-elemento
  - Mínimo: 1
  - Máximo: **unbounded** (“N”)
  - *Default*: 1

# Definição de Tipo Complexo

- Restrições de Cardinalidade
  - Exemplo

```
<xs:complexType name="tPessoa">
  <xs:sequence>
    <xs:element name="nome" type="xs:string"/>
    <xs:element name="fone" type="xs:integer"
      maxOccurs="Unbounded"/>
    <xs:element name="email" type="xs:string"
      minOccurs="0" maxOccurs="Unbounded"/>
  </xs:sequence>
</xs:complexType>
```

# Definição de Tipo Complexo

- Definição de **Atributo(s)** de um tipo complexo
  - Sintaxe: `<attribute>...</attribute>`
  - Especificado(s) após o construtor (`sequence`, `choice` ou `all`) de mais alto nível
  - Exemplo

```
<xs:complexType name="tPublic">
  <xs:sequence>
    <xs:element name="nome" type="xs:string"/>
    <xs:choice>
      <xs:element name="ISBN" type="xs:integer"/>
      <xs:element name="volume" type="xs:integer"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="ano" type="xs:integer"/>
  <xs:attribute name="edicao" type="xs:integer"/>
</xs:complexType>
```

# Definição de Tipo Complexo

- Restrições de Cardinalidade p/ Atributos
  - Definido no atributo **use**
  - Valores: **required** e **optional**
  - *Default*: **optional**
- Exemplo

```
<xs:complexType name="tPublic">
  ...
  <xs:attribute name="ano" type="xs:integer" use="required"/>
  <xs:attribute name="edicao" type="xs:integer" use="optional"/>
</xs:complexType>
```

# Definição de Tipo Complexo

- Restrições de Integridade p/ Atributos
  - Atributo **default**
    - define um *default* se nenhum valor for informado
  - Atributo **fixed**
    - define um valor constante para o atributo
    - se valor do atributo for diferente do valor fixo estipulado ou sem valor então o doc. XML é inválido
    - **default** e **fixed** não podem ser declarados juntos
- Exemplo

```
<xs:complexType name="tPublic">
  ...
  <xs:attribute name="pais" type="xs:string" fixed="Brasil"/>
  <xs:attribute name="lingua" type="xs:string" use="optional"
  default="Portugues"/>
</xs:complexType>
```

# Definição de Tipo Complexo

- Atributo identificador e de referência
  - **ID** e **IDREF**
  - Funcionalidades semelhantes às apresentadas para uma DTD
  - Exemplo

```
<xs:complexType name="tFuncionario">  
  ...  
  <xs:attribute name="codigo" type="xs:ID" use="required"/>  
  <xs:attribute name="chefe" type="xs:IDREF"/>  
</xs:complexType>
```

# Definição de Tipo Complexo

- Elemento Misto

- Atributo **mixed** definido como *true*

- Exemplo

```
<xs:complexType name="tAnuncio" mixed="true">
  <xs:sequence>
    <xs:element name="transacao" type="xs:string"/>
    <xs:element name="produto" type="xs:string"/>
    <xs:element name="foneContato" type="xs:integer"
maxOccurs="Unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="anuncio" type="tAnuncio"/>
```



# Definição de Tipo Complexo

- Elemento Vazio
  - Conteúdo do `complexType` é `vazio` ou `define apenas atributos`
  - Exemplo

```
<xs:complexType name="tImagem">  
  <xs:attribute name="arquivo" type="xs:string"  
    use="required"/>  
  <xs:attribute name="tamanho" type="xs:integer"/>  
</xs:complexType>  
  
<xs:element name="imagem" type="tImagem"/>
```

# Definição de Tipo Complexo

- Elemento **ANYTYPE**

- Funcionalidade semelhante ao elemento do tipo **ANY** apresentado para uma DTD

- Exemplo

```
<xs:complexType name="tComentario">
  <xs:sequence>
    <xs:element name="autor" type="xs:string"/>
    <xs:element name="data" type="xs:data"/>
    <xs:element name="texto" type="xs:anytype"/>
  </xs:sequence>
</xs:complexType>
```

# Exercício 2

- Defina em *XML Schema* o esquema abaixo definido em DTD para dados de livros:

```
<!ELEMENT listaLivros (livro+)>
<!ELEMENT livro (título, preço, autor+,
                 capítulo+)>
<!ATTLIST livro ISBN ID #REQUIRED
              edicaoAnterior IDREF #IMPLIED>
<!ELEMENT título (#PCDATA)>
<!ELEMENT autor (nome, eMail?)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT preço (#PCDATA)>
<!ELEMENT eMail (#PCDATA)>
<!ELEMENT capítulo (#PCDATA | seção)*>
<!ATTLIST capítulo nome CDATA #REQUIRED>
<!ELEMENT seção (nome, conteúdo)>
<!ELEMENT conteúdo (#PCDATA)>
```

# Derivação

- Conceito similar à **Especialização** em OO
  - “herança” de propriedades, com possibilidade de redefinição e definição de novas propriedades
- Tipos de derivação na XML Schema
  - **Extensão** (para tipos complexos)
    - Herança com definição de **novas propriedades**
  - **Restrição** (para tipos simples)
    - Herança com **redefinição mais restritiva de propriedades**

# Derivação por Extensão - Exemplo

```
<xsd:complexType name="Tlivro">
  <xsd:sequence>
    <xsd:element name="titulo" type="xsd:string"/>
    <xsd:element name="autor" type="Tautor"
      maxOccurs="unbounded"/>
    <xsd:element name="preco" type="xsd:float"/>
    ...
  </xsd:sequence>
  <xsd:attribute name="ISBN" type="Tisbn"/>
</xsd:complexType>
...

<xsd:complexType name="TlivroTécnico"
  base="Tlivro" derivedBy="extension">
  <xsd:element name="area" type="xsd:string"/>
</xsd:complexType>
```

# Definição de Tipo Simples

- Sintaxe: `<simpleType>...</simpleType>`
- Definido através de derivação por restrição
  - aplicação de uma restrição de integridade (**faceta**)
    - exemplos:
      - enumeração, intervalo de valores permitidos, padrão
- Definido através de um tipo de dado especial
  - Tipo lista (*list*)
    - define um conteúdo multivalorado
    - todos os valores devem ser de um mesmo tipo
  - Tipo união (*union*)
    - define alternativas (uma união de) de tipos de dados para o tipo sendo criado

# Definição de Tipo Simples - Facetas

- Maioria se aplica a qq tipo de dado simples
  - exemplos: `length`, `enumeration`, `pattern`
- Algumas se aplicam apenas a tipos de dados que possuem noção de ordem
  - exemplos: `minInclusive`, `maxInclusive`

```
<xs:simpleType name="Nota">  
  <xs:restriction base="xs:float">  
    <xs:minInclusive value="0"/>  
    <xs:maxInclusive value="10"/>  
  </xs:restriction>  
</xs:simpleType>
```

# Definição de Tipo Simples - Facetas

- Faceta **length**

- Define o tamanho (nro. caracteres) do dado

- Exemplos

```
<xs:simpleType name="TipoUF">
  <xs:restriction base="xs:string">
    <xs:length value = "2"/>
  </xs:restriction>
</xs:simpleType>
...
<xs:element name="UF" type="TipoUF"/>
```

```
<xs:simpleType name="TipoIdade">
  <xs:restriction base="xs:integer">
    <xs:minlength value = "1"/>
    <xs:maxlength value = "3"/>
  </xs:restriction>
</xs:simpleType>
...
<xs:element name="idade" type="TipoIdade"/>
```



# Definição de Tipo Simples - Facetas

- Faceta **enumeration**
  - Define um conjunto finito de valores permitidos
  - Exemplo

```
<xs:simpleType name="CategoriaAlunoPG">  
  <xs:restriction base="xs:string">  
    <xs:enumeration value = "especializacao"/>  
    <xs:enumeration value = "mestrado"/>  
    <xs:enumeration value = "doutorado"/>  
    <xs:enumeration value = "especial"/>  
  </xs:restriction>  
</xs:simpleType>  
...  
<xs:attribute name="categoria" type="CategoriaAlunoPG"/>
```

# Definição de Tipo Simples - Facetas

- Faceta **pattern**
  - Define expressões regulares
  - Exemplos

```
<xs:simpleType name="TnomePessoa">  
  <xs:restriction base="xs:string">  
    <xs:pattern value="[A-Z]{1}[a-zA-Z]{2,}" />  
  </xs:restriction>  
</xs:simpleType>
```

```
<xs:simpleType name="Tisbn">  
  <xs:restriction base="xs:string">  
    <xs:pattern value="[0-9]{2}-[0-9]{3}-[0-9]{4}-[0-9]" />  
  </xs:restriction>  
</xs:simpleType>
```

# Definição de Tipo Simples

- Definição de um tipo multivalorado

```
<xs:simpleType name="data">  
  <xs:union memberTypes="xs:date xs:string" />  
</xs:simpleType>  
  
<xs:simpleType name="ISBNs">  
  <xs:list itemType="ISBN" />  
</xs:simpleType>
```

*List e union só definem tipos simples, admitindo apenas componentes de tipo simples*