Universidade Federal de Santa Catarina

Esquemas XML

Saulo Popov Zambiasi

Roteiro

- Avaliação de Documentos XML
- Esquemas XML
- DTD
  - Sintaxe
  - Referência a um DTD
  - Exemplos
- *Namespaces*
- XML *Schema*
  - Estrutura
  - Exemplos

Avaliação de Documentos XML

Avaliação de Documentos XML

- **Documento XML <u>Bem-formado</u>:** está de acordo com as regras sintáticas da linguagem XML.

- **Documento XML <u>Válido</u>:**
  - Se for *bem-formado*, e ;
  - Estiver de acordo com a estrutura definida no ***esquema*** correspondente.

- Em XML, um ***esquema*** define formalmente a estrutura que os documentos devem ter, ou seja, a ordem e aninhamento dos *tags*. Um esquema pode ser:
  - **DTD (*Document Type Definition*)**
  - **XML *Schema***

Esquemas XML

Esquemas XML

- Um *esquema* (**DTD** ou **XML *Schema***) define a estrutura de documentos XML.
- Documentos XML podem ser validados automaticamente a partir de seu esquema correspondente.
- A definição de um esquema em comum permite o intercâmbio de dados entre aplicações heterogêneas.

DTD - *Document Type Definition*

DTD - *Document Type Definition*

- Possui sintaxe diferente da XML.
- Não permite a definição de tipos de dados, com exceção de alguns poucos tipos para atributos. Todos os campos são do tipo texto (*string*).
- Por não definir completamente os tipos de dados, os valores dos elementos devem ser validados pela aplicação (por exemplo, valores numéricos, datas).
- Pode ser declarado dentro de um documento XML ou como uma entidade externa.

DTD Externo

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE mensagem SYSTEM "msg.dtd">
<mensagem>
  <de>Leandro</de>
  <para>Fabiano</para>
  <assunto>Festa!!</assunto>
  <corpo>Não esqueça da festa na sexta!!</corpo>
</mensagem>
```

Referência ao DTD é feita na Document Type Declaration

Arquivo "msg.dtd":

```
<!ELEMENT mensagem (de,para,assunto,corpo)>
 <!ELEMENT de (#PCDATA)>
 <!ELEMENT para (#PCDATA)>
 <!ELEMENT assunto (#PCDATA)>
 <!ELEMENT corpo (#PCDATA)>
```

DTD Interno

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE mensagem [
  <!ELEMENT mensagem (de,para,assunto,corpo)>
  <!ELEMENT de (#PCDATA)>
  <!ELEMENT para (#PCDATA)>
  <!ELEMENT assunto (#PCDATA)>
  <!ELEMENT corpo (#PCDATA)>
]>
<mensagem>
  <de>Leandro</de>
  <para>Fabiano</para>
  <assunto>Festa!!</assunto>
  <corpo>Não esqueça da festa na sexta!!</corpo>
</mensagem>
```

Exemplos (DTD)

Exemplo 1: Mail Box

```
<!ELEMENT mail_box (mensagem*)>
 <!ELEMENT mensagem (de, para, cc*, anexo*, assunto, corpo)>
 <!ATTLIST mensagem data CDATA #REQUIRED>
  <!ELEMENT de (#PCDATA)>
  <!ELEMENT para (#PCDATA)>
  <!ELEMENT cc (#PCDATA)>
  <!ELEMENT anexo EMPTY>
  <!ATTLIST anexo nome CDATA #REQUIRED>
  <!ELEMENT assunto (#PCDATA)>
  <!ELEMENT corpo (#PCDATA)>
```

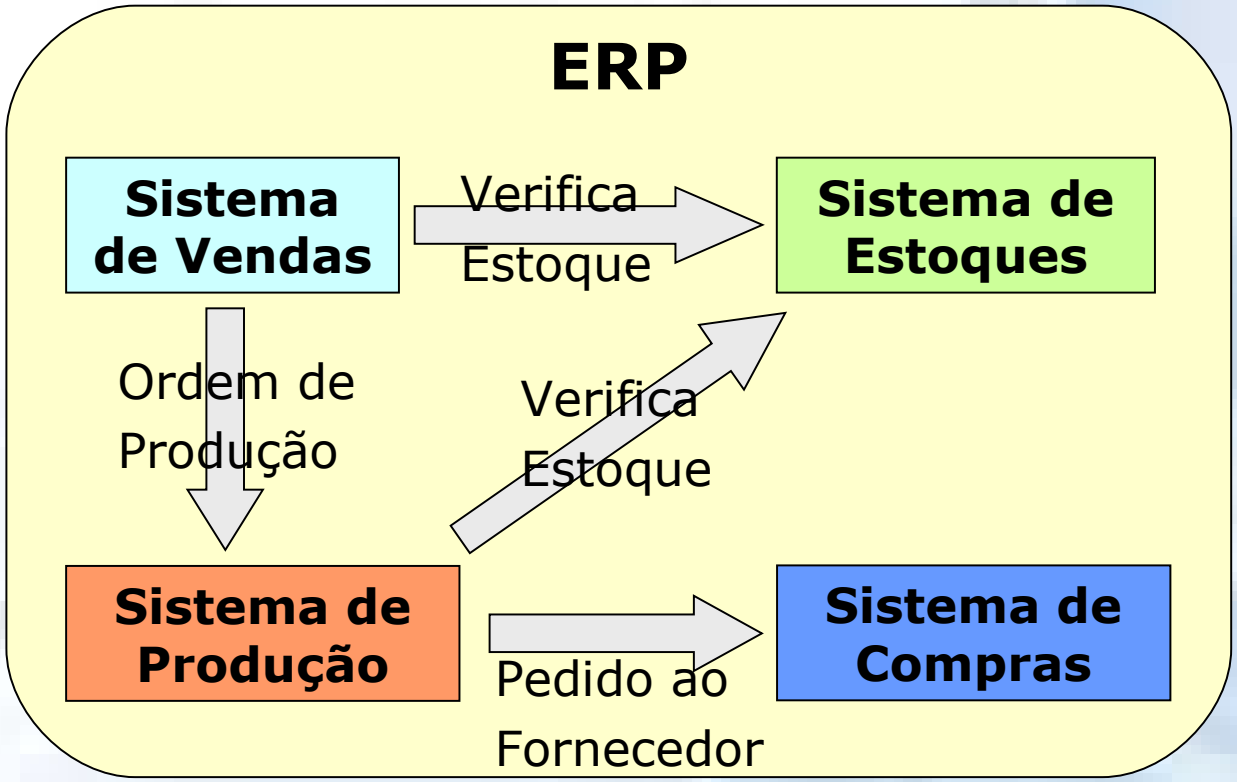
Exemplo 2: Livros

```
<?xml version="1.0"?>
<!DOCTYPE livros SYSTEM "livros.dtd">
<livros>
  <livro>
    <titulo>Professional XML</titulo>
    <autor>Nicola Ozu et al.</autor>
    <editora>Wrox UK</editora>
    <ano>2000</ano>
  </livro>
  <livro>
    <titulo>Java &amp;amp; XML</titulo>
    <autor>Brett MacLauglin</autor>
    <editora>O'Reilly</editora>
    <ano>2001</ano>
  </livro>
  <livro>
    <titulo>XML Handbook</titulo>
    <autor>Charles F. Goldfarb, Paul Prescod</autor>
    <editora>Prentice Hall</editora>
    <ano>1998</ano>
  </livro>
</livros>
```

Exemplo 2: Livros

```
<!ELEMENT livros (livro+)>
 <!ELEMENT livro (titulo, autor, editora, ano)>
  <!ELEMENT titulo (#PCDATA)>
  <!ELEMENT autor (#PCDATA)>
  <!ELEMENT editora (#PCDATA)>
  <!ELEMENT ano EMPTY>
```

Exemplo “real”



XML e DTD: Verifica Estoque

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE verificaEstoque SYSTEM
"verifica_estoque.dtd">
<verificaEstoque>
    <produto código="XPTO"/>
    <quantidade>20</quantidade>
</verificaEstoque>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT verificaEstoque (produto, quantidade)>
  <!ELEMENT produto EMPTY>
  <!ATTLIST produto código CDATA #REQUIRED>
  <!ELEMENT quantidade (#PCDATA)>
```

Doc. XML: Ordem de Produção

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE ordemProdução SYSTEM "ordem_prod.dtd">

<ordemProdução>
    <número>OP_2532</número>
    <data>2001-09-22</data>
    <dataEntrega>2001-09-25</dataEntrega>
    <item id="1">
      <quantidade>20</quantidade>
      <produto código="XPTO"/>
    </item>
</ordemProdução>
```

DTD: Ordem de Produção

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!ELEMENT ordemProdução (número, data, dataEntrega,
item+)>
  <!ELEMENT número (#PCDATA)>
  <!ELEMENT data (#PCDATA)>
  <!ELEMENT dataEntrega (#PCDATA)>
  <!ELEMENT item (quantidade, produto)>
  <!ATTLIST item id CDATA #REQUIRED>
    <!ELEMENT quantidade (#PCDATA)>
    <!ELEMENT produto EMPTY>
    <!ATTLIST produto código CDATA #REQUIRED>
```

Doc. XML: Pedido de Material

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE pedidoMaterial SYSTEM "pedido_material.dtd">
<pedidoMaterial>
    <número>P_763</número>
    <códigoFornecedor>P_763</códigoFornecedor>
    <data>2001-09-22</data>
    <dataEntrega>2001-09-23</dataEntrega>
    <item id="1">
      <quantidade>30</quantidade>
      <produto código="XYZ"/>
    </item>
    <item id="2">
      <quantidade>10</quantidade>
      <produto código="ABCD"/>
    </item>
</pedidoMaterial>
```

DTD: Pedido de Material

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!ELEMENT pedidoMaterial (número, códigoFornecedor,
data, dataEntrega, item+)>
  <!ELEMENT número (#PCDATA)>
  <!ELEMENT códigoFornecedor (#PCDATA)>
  <!ELEMENT data (#PCDATA)>
  <!ELEMENT dataEntrega (#PCDATA)>
  <!ELEMENT item (quantidade, produto)>
  <!ATTLIST item id CDATA #REQUIRED>
    <!ELEMENT quantidade (#PCDATA)>
    <!ELEMENT produto EMPTY>
    <!ATTLIST produto código CDATA #REQUIRED>
```

Namespaces

Namespaces

- Java usa *packages* para evitar conflitos de nomes. Assim, diferentes classes podem ter o mesmo nome desde não estejam na mesmo *package*.
- XML tem um mecanismo similar, chamado *namespace* para nomes de elementos e atributos.
- *Namespaces* servem para definir espaços de nomes, ou seja, qualificam os nomes dos elementos e atributos para que não ocorram conflitos de nomes repetidos;
- Este mecanismo permite que um documento XML utilize elementos definidos por diferentes esquemas XML.

Namespaces

- Um *namespace* consiste em um **prefixo** (que qualifica o nome do elemento) e de um **URL** (que serve como um identificador único para o *namespace*);
- É declarado usando o atributo especial **xsmlns**:

```
<prefixo:elemento xmlns:prefixo="URLdoNamespace">
   filhos
</prefixo:elemento>
```

- O elemento e seus filhos fazem parte do *namespace*.
- É preciso salientar que o URL é apenas usado como um identificador, ele não precisa ser resolvido.

Namespaces

- **Exemplo:**

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element ... />
...
</xs:schema>
```

- Neste exemplo, o prefixo é a string "xs". Assim, o elemento "xs:schema" significa "elemento *schema* no *namespace http://www.w3.org/2001/XMLSchema*".

Namespaces

- **Exemplo:**

```
<?xml version="1.0"?>
<pedido
  xmlns:bibl="http://www.gsigma.ufsc.br/livros"
  xmlns:cli="http://www.gsigma.ufsc.br/clientes"
>
  <bibl:titulo>XML em 5 minutos</bibl:titulo>
  <bibl:valor>75,00</bibl:valor>
  <cli:titulo>Dr.</cli:titulo>
  <cli:nome>Giovani</cli:nome>
  <cli:sobrenome>da Silva</cli:sobrenome>
  <cli:fone>123-4567</cli:fone>
</pedido>
```

XML *Schema*

Introdução

- XML *Schema* é uma linguagem que foi criada com o intuito de suprir as limitações do DTD, principalmente a falta de tipos de dados.

- XML *Schemas* vem gradativamente subtituindo os DTDs, pois:
  - São extensíveis a adições futuras;
  - São mais expressivos que DTDs ;
  - São escritos em XML;
  - Suportam tipos de dados;
  - Suportam *namespaces*.

Introdução

- Por ser definida em XML, XML *Schema* é mais "volumosa" que o DTD, o que pode diminuir a legibilidade.
- Embora venha gradativamente substituindo o DTD, em casos mais simples o uso de DTD pode ser mais adequado.
- XML *Schema* foi originalmente proposto pela Microsoft, mas se tornou uma recomendação oficial da W3C em maio de 2001.
- Existem diversas linguagens de *esquema* para XML (W3C, Relax, etc.).

Características

- Suporte a Tipos de Dados:
  - Tipos podem ser validados;
  - É mais fácil trabalhar com dados de banco de dados;
  - É possível definir tipos de dados customizados;

- Usam sintaxe XML:
  - Não é preciso aprender uma nova linguagem;
  - Pode-se usar um *parser* XML (DOM) para processar e manipular um XML Schema;
  - É possível tranformar um XML *Schema* usando XSLT.

Características

- São extensíveis. Assim, é possível:
  - Reusar um *Schema* em outros *Schemas*;
  - Criar um tipo personalizado de dados (derivado);
  - Referenciar múltiplos Schemas a partir de um mesmo documento XML.

- XML *Schema* evita ambigüidades na interpretação dos dados. Por exemplo:

```
<data type="date">2004-03-11</data>
```

  - A correta interpretação do conteúdo é garantida pois o tipo "date" do XML *Schema* exige o formato "YYYY-MM-DD".

Estrutura do XML *Schema*

XML *Schema*: Estrutura

- O elemento raiz de um XML *Schema* (da W3C) é o elemento "schema", definido no *namespace* "http://www.w3.org/2001/XMLSchema".

- Um XML *Schema* permite declarar:
  - Elementos e atributos (vinculados a tipos de dados);
  - Tipos de dados (simples ou complexos);

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- Declarações do XML Schema -->

</xs:schema>
```

Declaração de Elementos e Atributos

Declaração de Elementos

- **Sintaxe Geral:**

```
<xs:element name="nome" type="tipo"/>
```

- ***Exemplos:***

```
<xs:element name="nome" type="xs:string"/>
<xs:element name="idade" type="xs:integer"/>
<xs:element name="data" type="xs:date"/>
```

```
<nome>Marcos</nome>
<idade>34</idade>
<data>1968-03-27</data>
```

Declaração de Atributos

- Atributos devem ser declarados ou referenciados dentro de tipos complexos.

- Sintaxe: `<xs:attribute name="nome" type="tipo"/>`

Exemplos:

```
<xs:attribute name="fileName" type="xs:string"/>
```

```
<attachment fileName="trabalho.doc"/>
```

```
<xs:attribute name="cor" type="xs:string" default="vermelha"/>
```

```
<xs:attribute name="cor" type="xs:string" fixed="azul"/>
```

Tipos Simples

Tipos Simples

- Um tipo simples contém apenas valores simples (string, inteiro, booleano, etc.) e não contém outros elementos ou atributos.
- XML Schema possui mais de 40 tipos simples predefinidos (*string*, *decimal*, *integer*, *boolean*, *date*, *time*, etc.)
- Tipos simples podem ser referenciados pelo seu **nome**, ou podem ser **anônimos**, quando definidos dentro da declaração de um elemento:

```
<xs:element name="nome" type="tipo"/>

<xs:simpleType name="tipo">
        ...
</xs:simpleType>
```

```
<xs:element name="nome">
  <xs:simpleType>
        ...
  </xs:simpleType>
</xs:element>
```

Restrições / Facetas

- Restrições (facetas) são usadas para controlar valores aceitáveis para tipos simples.

Restrição	Descrição
enumeration	Define um lista de valores aceitáveis.
fractionDigits	Especifica o número máximo de casas decimais.
length	Especifica o número exato de caracteres ou itens de lista permitidos.
maxExclusive	Especifica o limite superior para valores numéricos.
maxInclusive	Especifica o limite superior para valores numéricos.
maxLength	Especifica o número máximo de caracteres ou itens de lista permitidos.
minExclusive	Especifica o limite inferior para valores numéricos.
minInclusive	Especifica o limite inferior para valores numéricos.
minLength	Especifica o número mínimo de caracteres ou itens de lista permitidos.
pattern	Define a seqüência exata de caracteres aceitáveis.
totalDigits	Especifica o número exato de dígitos aceito.

Restrições / Facetas (intervalo)

- Este exemplo define um elemento chamado "idade" com a restrição de que o seu valor deve estar entre 0 e 100.

```
<xs:element name="idade">

  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="100"/>
    </xs:restriction>
  </xs:simpleType>

</xs:element>
```

Restrições / Facetas (conjunto)

- Elemento "carro" deve conter um dos valores especificados.
- O "tipoCarro" pode ser associado a outros elementos.

```
<xs:element name="carro" type="tipoCarro"/>

<xs:simpleType name="tipoCarro">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Audi"/>
    <xs:enumeration value="Golf"/>
    <xs:enumeration value="BMW"/>
  </xs:restriction>
</xs:simpleType>
```

Restrições / Facetas (tamanho)

- Elemento "senha" deve ter no mínimo 5 e no máximo 8 caracteres.

```
<xs:element name="senha">

  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="5"/>
      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>

</xs:element>
```

Tipos Complexos

Tipos Complexos

- Um tipo complexo é aquele que:
  - Contém outros elementos (podendo ter também atributos);
  - É vazio (podendo ter também atributos);
  - Contém valor de tipo simples, mas que também tem atributos.

- Podem ser anônimos, ou referenciados por nome:

```
<xs:element name="mensagem">
  <xs:complexType>
        ...
  </xs:complexType>
</xs:element>
```

```
<xs:element name="mensagem" type="tipoMsg"/>

<xs:complexType name="tipoMsg">
      ...
</xs:complexType>
```

Tipos Complexos

- O elemento "complexType" é usado para definir o conteúdo de tipos complexos através dos chamados *elementos de agrupamento*:
  - *sequence;*
  - *choice;*
  - *all;*
  - *attribute.*

- Se "complexType" não possuir nenhum destes elementos ou apenas "attribute", o elemento terá conteúdo vazio (equivale ao `EMPTY` do DTD).

Tipos Complexos

Sequence

- Cada elemento (**element**) declarado neste grupo deve aparecer na ordem especificada.

Exemplo:

```
<xs:complexType name="tipoMensagem">
  <xs:sequence>
    <xs:element name="de" type="xs:string"/>
    <xs:element name="para" type="xs:string"/>
    <xs:element name="assunto" type="xs:string"/>
    <xs:element name="corpo" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Tipos Complexos

Sequence

Exemplo (Doc. XML):

```
<mensagem>
    <de>Leandro</de>
    <para>Fabiano</para>
    <assunto>Festa!!</assunto>
    <corpo>Não esqueça da festa na sexta!!</corpo>
</mensagem>
```

Tipos Complexos

Choice

- Apenas um dos elementos declarados neste grupo deve ocorrer (análogo ao "|" do DTD).

Exemplo:

```
<xs:complexType name="connection">
  <xs:choice>
    <xs:element name="http" type="tipoVazio"/>
    <xs:element name="ftp" type="tipoVazio"/>
    <xs:element name="smtp" type="tipoVazio"/>
  </xs:choice>
</xs:complexType>
```

Tipos Complexos

Choice

Exemplo (Doc. XML):

```
<connection>
    <http/>
</connection>
```

Tipos Complexos

Choice

- Elemento "choice" pode ser utilizado dentro de um "sequence".

Exemplo:

```
<xs:complexType name="connection">
  <xs:sequence>
    <xs:element name="soap" type="tipoSoap"/>
    <xs:choice>
      <xs:element name="http" type="tipoVazio"/>
      <xs:element name="ftp" type="tipoVazio"/>
      <xs:element name="smtp" type="tipoVazio"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

Tipos Complexos

Choice

Exemplo (Doc. XML):

```
<connection>
    <soap>...</soap>
    <http/>
</connection>
```

Tipos Complexos

All

- Cada elemento declarado neste grupo deve ocorrer no máximo uma vez, mas a ordem não é importante.

Exemplo:

```
<xs:complexType name="tipoMensagem">
  <xs:all>
    <xs:element name="de" type="xs:string"/>
    <xs:element name="para" type="xs:string"/>
    <xs:element name="assunto" type="xs:string"/>
    <xs:element name="corpo" type="xs:string"/>
  </xs:all>
</xs:complexType>
```

Tipos Complexos

All

Exemplo (Doc. XML):

```
<mensagem>
    <assunto>Festa!!</assunto>
    <de>Leandro</de>
    <corpo>Não esqueça da festa na sexta!!</corpo>
    <para>Fabiano</para>
</mensagem>
```

Tipos Complexos

Attribute

- Deve ser o último grupo declarado no tipo complexo.

Exemplo:

```
<xs:complexType name="tipoMensagem">
  <xs:sequence>
    <xs:element name="de" type="xs:string"/>
    <xs:element name="para" type="xs:string"/>
    <xs:element name="assunto" type="xs:string"/>
    <xs:element name="corpo" type="xs:string"/>
  </xs:sequence>
  <xs:attribute name="data" type="xs:date" use="required"/>
</xs:complexType>
```

Tipos Complexos

Attribute

Exemplo (Doc. XML):

```
<mensagem data="2005-06-12">
    <de>Leandro</de>
    <para>Fabiano</para>
    <assunto>Festa!!</assunto>
    <corpo>Não esqueça da festa na sexta!!</corpo>
</mensagem>
```

Tipos Complexos

Atributos de Cardinalidade

- Os elementos “element”, “sequence”, “choice”, e “all” possuem dois atributos opcionais, que indicam o número de ocorrências:
  - **minOccurs**: número mínimo de ocorrências;
  - **maxOccurs**: número máximo de ocorrências.
- Quando omitidos, assumem valor “1”.
- “maxOccurs” pode conter o valor “unbounded”, que não impõe restrições ao número de ocorrências.

Tipos Complexos

Atributos de Cardinalidade

- Exemplo:

```
<xs:element name="mensagem" type="tipoMensagem"/>
<xs:complexType name="tipoMensagem">
 <xs:sequence>
  <xs:element name="de" type="xs:string"/>
  <xs:element name="para" type="xs:string"/>
  <xs:element name="cc" type="xs:string" minOccurs="0" maxOccurs="10"/>
  <xs:element name="assunto" type="xs:string"/>
  <xs:element name="corpo" type="xs:string"/>
 </xs:sequence>
</xs:complexType>
```

Tipos Complexos

Atributos de Cardinalidade

- Exemplo:

```
<xs:complexType name="connection">
  <xs:sequence>
    <xs:element name="soap" type="tipoSoap"/>
    <xs:choice maxOccurs="unbounded">
      <xs:element name="http" type="tipoVazio"/>
      <xs:element name="ftp" type="tipoVazio"/>
      <xs:element name="smtp" type="tipoVazio"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

Tipos Complexos

- Um tipo complexo pode conter também valores simples (texto e atributos).
- Exemplo:

```
<xs:element name="altura" type="tipoAltura"/>

<xs:complexType name="tipoAltura">
  <xs:simpleContent>
    <xs:extension base="xs:integer">
      <xs:attribute name="unidade" type="xs:string" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

```
<altura unidade="metro">273</altura>
```

Fazendo um Doc. XML referenciar um XML *Schema*

Referência ao XML *Schema*

- O elemento raiz deve conter dois atributos especiais, além dos seus próprios:
  - ***xmlns***, que referencia o namespace *"http://www.w3.org/2001/XMLSchema-instance"*
  - ***noNamespaceSchemaLocation***, que referencia diretamente o XML Schema.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<mensagem xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="msg.xsd" data="2003-12-11">
...
</mensagem>
```

Exemplos (XML *Schema*)

Doc. XML: Mail Box

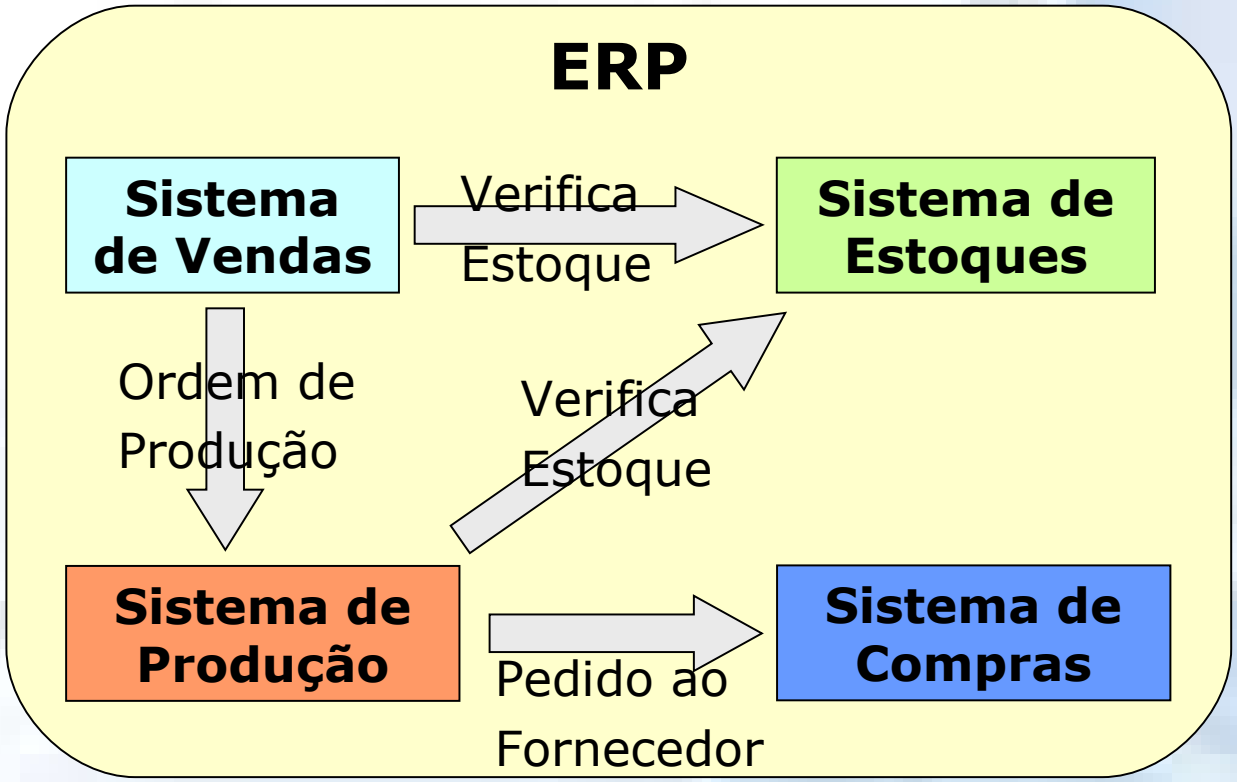
```
<?xml version="1.0" encoding="ISO-8859-1"?>

<mensagem xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="msg.xsd" data="2003-12-11">
  <de>Leandro</de>
  <para>Fabiano</para>
  <cc>Carlos</cc>
  <anexo nome="mapa.jpg"/>
  <assunto>Festa!!</assunto>
  <corpo>Não esqueça da festa na sexta!!</corpo>
</mensagem>
```

XML *Schema*: Mail Box

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="mensagem" type="tipoMensagem"/>
<xs:complexType name="tipoMensagem">
 <xs:sequence>
  <xs:element name="de" type="xs:string"/>
  <xs:element name="para" type="xs:string"/>
  <xs:element name="cc" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
  <xs:element name="anexo" minOccurs="0">
   <xs:complexType>
    <xs:attribute name="nome" type="xs:string" use="required"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="assunto" type="xs:string"/>
  <xs:element name="corpo" type="xs:string"/>
 </xs:sequence>
 <xs:attribute name="data" type="xs:date" use="required"/>
</xs:complexType>
</xs:schema>
```

Exemplo “real”



Doc. XML: Verifica Estoque

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<verificaEstoque
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="verifica_estoque.xsd">
    <produto código="XPTO"/>
    <quantidade>20</quantidade>
</verificaEstoque>
```

XML *Schema*: Verifica Estoque

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="verificaEstoque" type="tipoEstoque"/>
 <xs:complexType name="tipoEstoque">
  <xs:sequence>
    <xs:element name="produto">
      <xs:complexType>
       <xs:attribute name="código" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="quantidade" type="xs:integer"/>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

Doc. XML: Ordem de Produção

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<ordemProdução
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="ordem_prod.xsd">
    <número>OP_2532</número>
    <data>2001-09-22</data>
    <dataEntrega>2001-09-25</dataEntrega>
    <item id="1">
      <quantidade>10</quantidade>
      <produto código="XPTO"/>
    </item>
</ordemProdução>
```

XML *Schema*: Ordem de Produção

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="ordemProdução">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="número" type="xs:string"/>
    <xs:element name="data" type="xs:date"/>
    <xs:element name="dataEntrega" type="xs:date"/>
    <xs:element name="item" minOccurs="1" maxOccurs="20">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="quantidade" type="xs:int"/>
          <xs:element name="produto">
            <xs:complexType>
              <xs:attribute name="código" type="xs:string" use="required"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="id" type="xs:int" use="required"/>
      </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Doc. XML: Pedido de Material

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<pedidoMaterial xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="pedido_material.xsd">
    <número>P_763</número>
    <códigoFornecedor>P_763</códigoFornecedor>
    <data>2001-09-22</data>
    <dataEntrega>2001-09-23</dataEntrega>
    <item id="1">
      <quantidade>30</quantidade>
      <produto código="XYZ"/>
    </item>
    <item id="2">
      <quantidade>10</quantidade>
      <produto código="ABCD"/>
    </item>
</pedidoMaterial>
```

XML *Schema*: Pedido de Material

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="pedidoMaterial" type="tipoPedido"/>
 <xs:complexType name="tipoPedido">
  <xs:sequence>
    <xs:element name="número" type="xs:string"/>
    <xs:element name="códigoFornecedor" type="xs:string"/>
    <xs:element name="data" type="xs:date"/>
    <xs:element name="dataEntrega" type="xs:date"/>
    <xs:element name="item" minOccurs="1" maxOccurs="10">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="quantidade" type="xs:int"/>
          <xs:element name="produto">
            <xs:complexType>
              <xs:attribute name="código" type="xs:string" use="required"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="id" type="xs:int" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

Exercício

- A partir do modelo de dados definido, fazer um XML Schema que representa uma **empresa** com os seus respectivos produtos em **estoque**. Fazer também um documento XML válido.

- Empresa
  - Código da empresa
  - Nome da empresa
  - Rua
  - Número
  - CEP
  - Bairro
  - Cidade
  - Estado
  - *Produtos em estoque*

- Estoque
  - Código do produto
  - Nome do Produto
  - Preço
  - Quantidade

Exercício

- A partir do modelo de dados definido, fazer um XML Schema que representa uma **ordem de venda** com os seus respectivos **produtos**. Fazer também um documento XML válido.

- Ordem de Venda
  - Código da ordem
  - Nome consumidor
  - Data
  - Valor da ordem
  - Foi entregue ao cliente?
  - *Código dos produtos*
  - *Nome dos Produtos*
  - *Preços*
  - *Quantidade de cada produto*

Referências

- **W3C – www.w3.org/XML**
- **W3 Schools – www.w3schools.com**
- **EditiX (Editor XML) - www.editix.com**
- **XML Software – www.xmlsoftware.com**
- **The XML Industry Portal – www.xml.org**
- **Microsoft – msdn.microsoft.com/xml**
- **Tabelas de codificação de caracteres – en.wikipedia.org/wiki/Character_encoding.**