XML Schema

Vanessa Braganholo

Curso baseado em mini-cursos apresentados no SBBD. Autores: Carlos Heuser, Carina Dorneles e Vanessa Braganholo

XML Schema

- Utilizado para descrever a estrutura de um documento XML, assim como a DTD
- Utiliza sintaxe XML
- Sintaxe simples: fácil compreensão humana
- Introduz tipos de dados
  - data, string, números, etc.
- Estrutura

```
<xs:schema>
   <!-- declaração de tipos, elementos e atributos -->
</xs:schema>
```

Basicamente...

- **Todos** os elementos devem ser associados a tipos

- Os elementos atômicos (folhas da arvore XML) e atributos
  - Tipos Básicos
    - Definição de tipos primitivos - data, número, string, etc
  - Tipos Simples
    - Uso de simpleType
    - Definição de estruturas simples a partir dos tipos básicos

- Os elementos compostos
  - Tipos Complexos
    - Uso de complexType
    - Definição de estruturas complexas

Uso de *namespace*

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">
    <!-- declaração de tipos, elementos e atributos -->
</xs:schema>
```

- Não é necessário colocar os atributos **elementFormDefault** e **attributeFormDefault**
  - Quando não especificados, assume-se o valor default para ambos (“unqualified”)
  - O software XML Exchanger Lite exige que estes atributos sejam declarados explicitamente quando houver criação de namespace

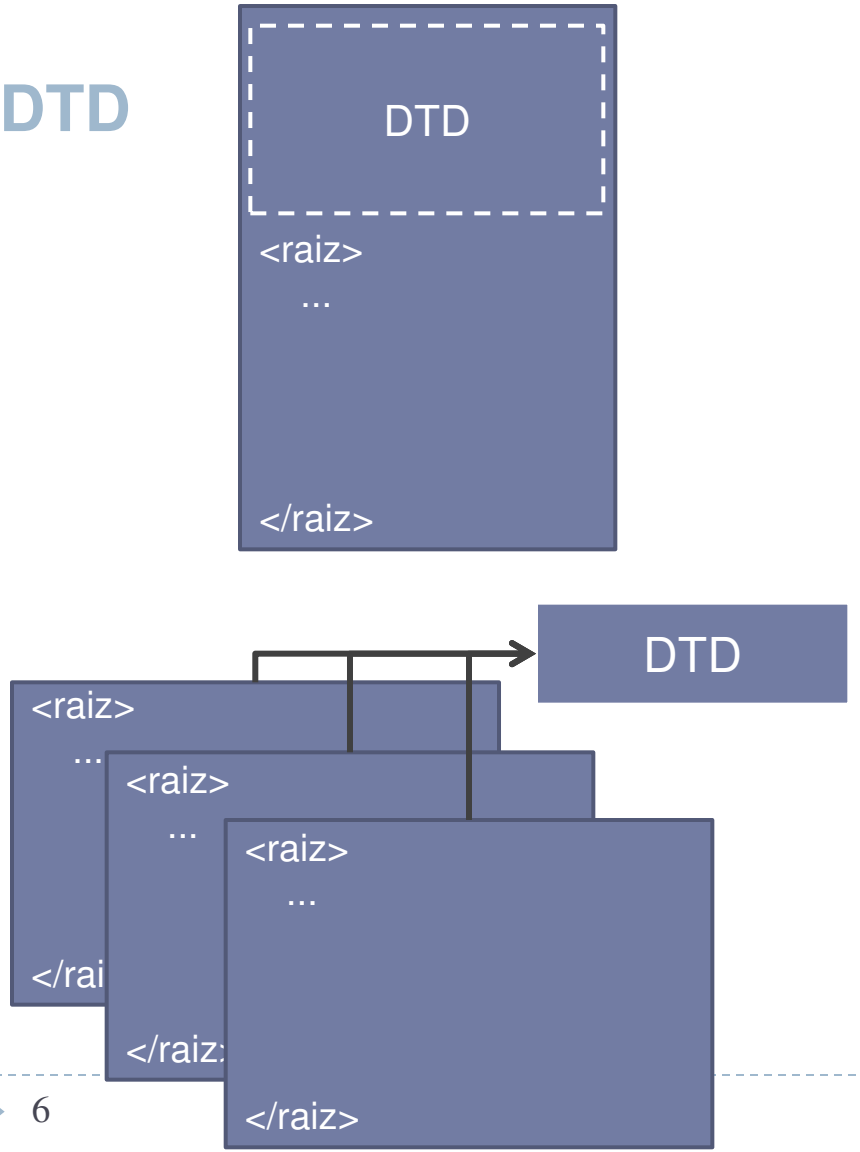
Uso de *namespace*

- Na instância XML, é necessário declarar o *namespace* do XML Schema
- Isto é feito no elemento raiz do documento:

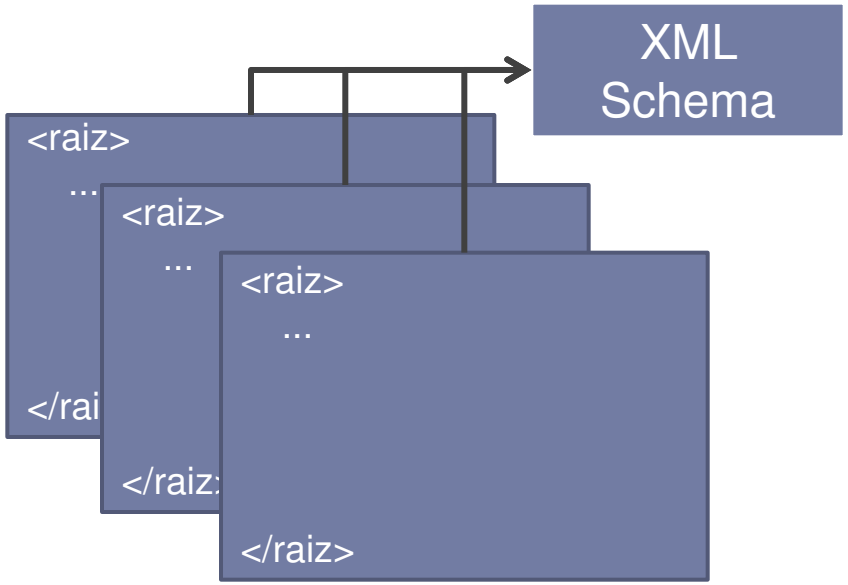
```
<bibliografia xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance”>
...
</bibliografia>
```

Declaração

DTD



XML Schema



Como ligar um XML a um XML Schema?

- No elemento raiz do documento XML, adicionar os atributos
  - **noNamespaceSchemaLocation** – quando não usamos namespace – valor do atributo é o caminho para o arquivo XSD

  OU

  - **schemaLocation** – necessário quando estamos usando um namespace associado ao nosso esquema – valor do atributo é o nome do namespace, um espaço em branco e o caminho para o arquivo XSD
    - Neste caso, é necessário também declarar o namespace

Como ligar um XML a um XML Schema? (Exemplos)

- Usando noNamespaceSchemaLocation
- No doc. XML:

```
<endereco xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance" xsi:noNamespaceSchemaLocation="endereco.xsd">
...
</endereco>
```

- No esquema:

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

Como ligar um XML a um XML Schema? (Exemplos)

- Usando schemaLocation
- No doc. XML:

```
<report xmlns="http://www.example.com/Report"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.example.com/Report
                        http://www.example.com/Report.xsd“>
...
</report>
```

- No esquema:

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

Definições

- Um documento XML Schema é um documento XML!!
- **Um documento XML Schema é um documento XML** onde são definidos os elementos, atributos e outras características de **outros documentos XML**

Definições

- Definições de elementos
  - **element** define um elemento e o associa a um tipo

- Exemplos:
  - Elemento atômico:
    - Define o elemento "rua" e o associa ao tipo "string"

```
<xs:element name="rua" type="xs:string"/>
```

  - Elemento composto
    - Define o elemento "endereco" e o associa ao tipo "tEndereco"

```
<xs:element name="endereco" type="tEndereco"/>
```

Os tipos...

- tEndereco

```
<xs:complexType name="tEndereco">
  <xs:sequence>
    <xs:element name="rua" type="xs:string"/>
    <xs:element name="numero" type="xs:integer"/>
    <xs:element name="bairro" type="xs:string"/>
    <xs:element name="cidade" type="xs:string"/>
    <xs:element name="estado" type="xs:string"/>
    <xs:element name="CEP" type="xs:string"/>
    <xs:element name="pais" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

- xs:string
  - Um dos tipos built in do XML Schema (xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, xs:time, etc)

Tipos complexos - **complexType**

- Define restrições para o modelo de conteúdo de um determinado elemento

- Feito através de atributos para especificação de:
  - Cardinalidade
    - minOccurs e maxOccurs
  - Delimitadores de grupos de elementos
    - sequence, choice e all

Cardinalidade

- xs:minOccurs
  - número mínimo de vezes que um subelemento pode aparecer.
    - Default = 1
- xs:maxOccurs
  - número máximo de vezes que um subelemento pode aparecer.
    - Default = 1
    - Max = unbounded

Cardinalidade - exemplo

```
<xs:complexType name="tEndereco">
  <xs:sequence>
    <xs:element name="rua" type="xs:string"
          minOccurs="0" maxOccurs="1"/>
    <xs:element name="numero" type="xs:integer"
          minOccurs="0" maxOccurs="1"/>
    <xs:element name="cidade" type="xs:string"
          minOccurs="0" maxOccurs="1"/>
    <xs:element name="estado" type="xs:string"
          minOccurs="0" maxOccurs="1"/>
    <xs:element name="CEP" type="tCep"
          minOccurs="0" maxOccurs="1"/>
    <xs:element name="email" type="xs:string"
          minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Delimitadores de grupo

- <xs:sequence>
  - subelementos devem aparecer na instância XML na mesma ordem em que foram declarados no esquema
- <xs:choice>
  - somente um dos elementos declarados no grupo pode aparecer na instância
- <xs:all>
  - os elementos do grupo podem aparecer **uma vez** em qualquer ordem

Sequence - exemplo

- No XML Schema:

```
<xs:complexType name="tEnder">
    <xs:sequence>
        <xs:element name="rua" type="xs:string"/>
        <xs:element name="numero" type="xs:integer"/>
        <xs:element name="cidade" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
<xs:element name=“endereco" type="tEnder“/>
```

- Na instância XML:

```
<endereco>
  <rua>Osvaldo Aranha</rua>
  <numero>1212121</numero>
  <cidade>Poa</cidade>
</endereco>
```

Sequence – comparando com DTD

- No XML Schema:

```
<xs:complexType name="tEnder">
  <xs:sequence>
    <xs:element name="rua" type="xs:string"/>
    <xs:element name="numero" type="xs:integer“/>
    <xs:element name="cidade" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="Endereco" type="tEnder“/>
```

- Na DTD:

```
<!ELEMENT Endereco (rua, numero, cidade)>
<!ELEMENT rua (#PCDATA)>
<!ELEMENT numero (#PCDATA)>
<!ELEMENT cidade (#PCDATA)>
```

Choice – exemplo

- No XML Schema:

```
<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“/>
```

- Na instancia XML:

```
<publicacao>
   <nome>Projeto de Banco de dados</nome>
   <ISBN>989898989</ISBN>
</publicacao>
```

```
<publicacao>
    <nome>SQL Magazine</nome>
    <volume>9</volume>
</publicacao>
```

Choice – comparando com a DTD

- No XML Schema:

```
<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“/>
```

- Na DTD:

```
<!ELEMENT publicacao (nome, (ISBN | volume))>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT volume (#PCDATA)>
```

All - Restrições

- Na instância XML
  - Todos os elementos no grupo podem aparecer uma única vez
  - Podem aparecer em qualquer ordem
- No XML Schema
  - Só pode aparecer como grupo mais externo de qualquer modelo de conteúdo
  - Os filhos de **all** devem ser todos elementos (não podem ser grupos)
  - Nenhum elemento pode ter cardinalidade maior que 1 (valores permitidos para **minOccurs** e **maxOccurs** são 0 e 1)
    - no caso de minOccurs = 0, o elemento é opcional

All - exemplo

- No XML Schema:

```
<xs:complexType name="tAut">
  <xs:all>
    <xs:element name="nome" type="xs:string"/>
    <xs:element name="email" type="xs:integer"/>
    <xs:element name="instituicao" type="xs:string"/>
  </xs:all>
</xs:complexType>
<xs:element name="autor" type="tAut“/>
```

- Na instancia XML:

```
<autor>
  <nome>Ana Clara</nome>
  <instituicao>Universidade XYZ</instituicao>
  <email>ana@server.domain</email>
</autor>
```

Todos juntos
Sem restrição de ordem

All – comparando com a DTD

- No XML Schema:

```
<xs:complexType name="tAut">
  <xs:all>
     <xs:element name="nome" type="xs:string"/>
     <xs:element name="email" type="xs:integer"/>
     <xs:element name="instituicao" type="xs:string"/>
  </xs:all>
</xs:complexType>
<xs:element name="autor" type="tAut“/>
```

All – comparando com a DTD

- Na DTD:

```
<!ELEMENT autor (
  (nome, email, instituicao) |
  (nome, instituicao, email) |
  (email, nome, instituicao) |
  (email, instituicao, nome) |
  (instituicao, nome, email) |
  (instituicao, email, nome)
)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT instituicao (#PCDATA)>
```

Exercício 1

- Construa um XMLSchema de tal forma que o seguinte documento XML possa ser validado:

```
<itens_pedido>
  <item>
    <produto>caneta azul</produto>
    <quantidade>100</quantidade>
    <preco_unit>2</preco_unit>
  </item>
  <item>
    <produto>caneta preta</produto>
    <quantidade>200</quantidade>
    <preco_unit>3</preco_unit>
  </item>
</itens_pedido>
```

Exercício 2

- Construa um XMLSchema para o elemento cliente de modo que ele possa ser ou pessoa física, ou pessoa jurídica

```
<cliente>
  <razao_social>JOAQUIM S.A.</razao_social>
  <cnpj>00.000.000/0001-00</cnpj>
</cliente>
```

ou

```
<cliente>
  <nome>JOSÉ</nome>
  <cpf>000.000.000-00</cpf>
</cliente>
```

Atenção: Sempre que aparecer nome, tem que aparecer CPF. Sempre que aparecer razão social, tem que aparecer CNPJ.

Atributos

- Atributos podem ser definidos com attribute
- Um atributo pode ser declarado dentro do escopo de um complexType
  - diferentes atributos podem ser declarados com o mesmo nome, mas com significados diferentes
- Quando declarados fora do escopo de um complexType
  - diferentes tipos complexos podem compartilhar atributos sem precisar redeclará-los
- Na declaração, não é necessário dizer a quem o atributo pertence

```
<xs:attribute name="data" type="xs:date"/>
```

Atributos

- use
  - required: obrigatório
  - optional: opcional
  - prohibited: atributo não pode ser usado no doc. XML
- default
  - Indica o valor *default*, caso ele seja omitido

```
<xs:attribute name="pais" type="xs:string"
        use="optional" default="Brasil"/>
```

Exemplo

```
<xs:complexType name="tEnder">
  <xs:sequence>
    <xs:element name="rua" type="xs:string"/>
    <xs:element name="numero" type="xs:integer">
    <xs:element name="cidade" type="xs:string"/>
  </xs:sequence>
  <xs:attribute name=“tipo” type=“xs:string”/>
</xs:complexType>
```

Referência

- Pode-se referenciar um elemento que tenha sido declarado anteriormente

```
<xs:element name="comentario" type="xs:string"/>

<!--Outras declarações -->
<xs:complexType name="tendereco">
  <xs:sequence>
     <xs:element ref="comentario"/>
     <--!etc-->
  </xs:sequence>
</xs:complexType>
```

Exercício 3

- Crie um esquema completo para o documento abaixo

```
<pedido numero="1001">
  <cliente>
  <razao_social>JOAQUIM</razao_social>
  <cnpj>00.000.000/0001-00</cnpj>
 </cliente>
 <itens_pedido>
  <item>
    <produto>caneta azul</produto>
    <quantidade>100</quantidade>
    <preco_unit>2</preco_unit>
  </item>
  <item>
    <produto>caneta preta</produto>
    <quantidade>200</quantidade>
    <preco_unit>3</preco_unit>
  </item>
 </itens_pedido>
</pedido>
```