

---

# Integração de Dados

2021/22

## Validação do Modelo XML XSD

### XSD: Características

---

- **Um Schema de XML define:**
  - os elementos
  - os atributos
  - qual o elemento raiz
  - que elementos são elementos filhos
  - a ordem de elementos dos filhos
  - o número de elementos filhos
  - se um elemento está vazio ou pode incluir o texto
  - tipos de dados para elementos e atributos
  - valores por defeito e e valores fixos para elementos e atributos.

## XSD: ligação ao XML

---

- **Validação de um documento XML com XSD**
  - Instância (ficheiro .XML)
  - Esquemas de validação escritas em sintaxe XSD
    - Num ficheiro .XSD separado
- **O ficheiro .xsd define as regras às quais a instância tem de obedecer;**

## XSD: ligação ao XML

---

- **Ligar a instância XML ao ficheiro XSD**

```
<?xml version="1.0" encoding="UTF-8"?>
<catalogo
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="catalogo.xsd">

  <cd pais="UK">
    <titulo>Dark Side of the Moon</titulo>
    <artista>Pink Floyd</artista>
    <preco>10.90</preco>
  </cd>
  ...

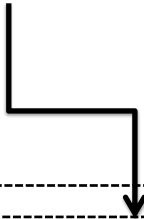
</catalogo>
```

Indica que a validação vai ser feita usando XSD

Localização do ficheiro XSD

## XSD: ligação ao XML

### ○ Um exemplo simples



```
<?xml version="1.0" encoding="UTF-8"?>
<catalogo xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="catalogo.xsd">
  <cd pais="UK">
    <titulo>Dark Side of the Moon</titulo>
    <artista>Pink Floyd</artista>
    <preco>10.90</preco>
  </cd>
  ...
</catalogo>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  ....
</xsd:schema>
```

## XSD: elemento raiz <xsd:schema>

### ○ O elemento <schema>

- É o elemento raiz de qualquer ficheiro xsd

```
<?xml version="1.0"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" ... >
  .....
</xsd:schema>
```

## XSD: tipos de elementos

---

- **Elementos**
  - **Simples**
    - Vazio (sem atributos)
    - Textual (folha sem atributos)
      - cardinalidade: nº de vezes que pode aparecer
      - tipos de dados (int, double, string, ...)
      - restrições (expressão regular, tamanho, enumeração, ...)
  - **Complexos**
    - Elemento vazio com atributos
    - Elemento textual (folha) com atributos
    - Elemento agregador com/sem atributos
    - Elemento de conteúdo misto com/sem atributos

## XSD: tipos de atributos

---

- **Atributos**
  - Simples
  - Com restrições
    - **expressão regular**
    - **enumeração**
    - **tamanho**
    - ...

## XSD: Elementos simples - VAZIO

- Definem-se no ficheiro **xsd** usando a seguinte sintaxe:

```
<xsd:element name="x" param="..." />
```

nome do elemento

param:

opcional  
pode ser

minOccurs = "..."  
maxOccurs = "..."

### Exemplo:

```
<produto .../>
```

```
<xsd:element name="produto"/>
```

## XSD: Elementos simples – TEXTUAL (folha)

- Definem-se no ficheiro **xsd** usando a seguinte sintaxe:

```
<xsd:element name="x" type="y" param="..." />
```

nome do elemento

tipo de dados

Tipos de dados  
mais comuns:

xsd:string  
xsd:decimal  
xsd:integer  
xsd:boolean  
xsd:date  
xsd:time  
xsd:ID

param:

opcional  
pode ser

default = "..."  
fixed = "..."  
minOccurs = "..."  
maxOccurs = "..."

## XSD: Elementos simples – TEXTUAL (folha)

- Elementos simples, exemplo

XML

```
...
<apelo>Simões</apelo>
<idade>36</idade>
<data>1970-03-27</data>
...
```

XSD

```
...
<xsd:element name="apelo" type="xsd:string"/>
<xsd:element name="idade" type="xsd:integer"/>
<xsd:element name="data" type="xsd:date"/>
...
```

## XSD: Elementos simples

- **parâmetro opcional:**

- **default:** indicado o valor por defeito se nada for indicado
- **fixed:** indica o valor fixo

- Exemplos:

```
<xsd:element name="nome_cliente" type="xsd:string" default="desconhecido">
<xsd:element name="local_cliente" type="xsd:string" fixed="Portugal"/>
```



- XML:

```
<nome_cliente>Maria</nome_cliente>
<local_cliente>Portugal</local_cliente>
```

## XSD: Elementos simples

---

- **cardinalidade:**
  - Especifica quantas vezes um elemento pode aparecer
  - Usam-se os atributos:
    - **minOccurs:**
      - valores inteiros  $\geq 0$
      - por defeito tem o valor 1
    - **maxOccurs:**
      - valores inteiros  $\geq 0$
      - "unbounded" (indica que não tem limite máximo)
      - por defeito tem o valor 1

## XSD: Elementos simples

---

### XSD

```
<xsd:element name="encomendas_cliente" type="xsd:integer" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="hobbies_cliente" type="xsd:string" minOccurs="2"
maxOccurs="10"/>
```

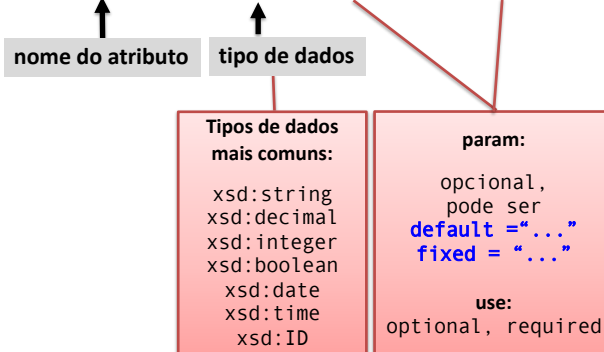
### XML

```
<raiz>
  <listaenc>
    <encomendas_cliente>10</encomendas_cliente>
    <encomendas_cliente>12</encomendas_cliente>
    <encomendas_cliente>5</encomendas_cliente>
    <encomendas_cliente>6</encomendas_cliente>
  </listaenc>
  <lista_hobbies>
    <hobbies_cliente>fotografia</hobbies_cliente>
    <hobbies_cliente>cinema</hobbies_cliente>
  </lista_hobbies>
</raiz>
```

## XSD: Atributos simples

- Atributos – declarados como os elementos simples:

```
<xsd:attribute name="xxx" type="yyy" param="..." use="..."/>
```



Não são atribuídos ao elemento a que pertencem!

## XSD: Atributos simples

Exemplos:

```
<xsd:attribute name="lingua" type="xsd:string"/>
<xsd:attribute name="casado" type="xsd:boolean"/>
<xsd:attribute name="codigo" type="xsd:ID"/>
```

```
<xsd:attribute name="lingua" type="xsd:string" default="EN"/>
<xsd:attribute name="lingua" type="xsd:string" fixed="EN"/>
```



## XSD: Atributos simples

---

- **Atributos** – Podem ser opcionais ou obrigatórios
  - Define-se com o atributo **use**
  - **optional**: indica o atributo pode não ser usado
  - **required**: indica que o atributo tem de aparecer

### Exemplos:

```
<xsd:attribute name="lingua" type="xsd:string" use="optional"/>  
<xsd:attribute name="lingua" type="xsd:string" use="required"/>
```

## XSD: Elementos/atributos simples - **restriction**

---

- Como restringir a gama de valores de um **elemento** ou **atributo**?
  - Por exemplo, permitir que a idade de uma pessoa (do tipo `xsd:integer`) não seja menor do que 0 nem superior a 120
- Com o XSD **restriction** é possível restringir:
  - os limites de valores
  - o conteúdo dos valores (enumeração de valores possíveis)
  - os caracteres/dígitos válidos (expressão regular)
  - o tamanho
  - ...

## XSD: Elementos/atributos simples - **restriction**

### ○ Sintaxe:

```

<xsd:element name="nome_elemento">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      ...
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

```

Elemento (ou atributo) a restringir

Indica que é um elemento simples

qual o tipo de dados a restringir

## XSD: Elementos/atributos simples - **restriction**

### ○ restrições nos limites de valores (para xsd:integer)

- **<xsd:minInclusive value="10"/>** → ≥ 10
- **<xsd:minExclusive value="10"/>** → > 10
- **<xsd:maxInclusive value="10"/>** → ≤ 10
- **<xsd:maxExclusive value="10"/>** → < 10

Restringir os valores do elemento idade ≥ 0 e ≤ 120

```

<xsd:element name="idade">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="120"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

```

## XSD: Elementos/atributos simples - **restriction**

- restrições nos valores possíveis (enumerações)

**<xsd:enumeration** value = "..."/>

Restringir os valores do elemento **carro** a 3 possibilidades:

```
<xsd:element name="carro">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Audi"/>
      <xsd:enumeration value="Golf"/>
      <xsd:enumeration value="Honda"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

## XSD: Elementos/atributos simples - **restriction**

- restrições caracteres possíveis

**<xsd:pattern** value="expressão regular"/>

Restringir os valores do elemento **palavra** aos caracteres alfabéticos:

```
<xsd:element name="palavra">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[a-zA-Z]+"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

## XSD: Elementos/atributos simples - **restriction**

Mais exemplos com `<xsd:pattern = "expressão regular"/>`

Restringir os valores do elemento **genero** a duas possibilidades (masc ou fem):

```
<xsd:element name="genero">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="masc|fem"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Válidos:

```
<genero...>masc</genero>
<genero...>fem</genero>
```

Inválidos:

```
<genero...>M</genero>
<genero...>Fem</genero>
```

## XSD: Elementos/atributos simples - **restriction**

○ **restrições no tamanho dos elementos (xsd:string):**

○ `<xsd:length="value"/>`

○ Número exacto de caracteres que o elemento deve ter

○ `<xsd:maxLength="value"/>`

○ Número máximo de caracteres que o elemento deve ter

○ `<xsd:minLength="value"/>`

○ Número mínimo de caracteres que o elemento deve ter

## XSD: Elementos/atributos simples - **restriction**

- restrições no tamanho dos elementos (xsd:string)
  - length

O elemento **password** deve ter exactamente 8 caracteres (xsd:string):

```
<xsd:element name="password">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

## XSD: Elementos/atributos simples - **restriction**

- restrições no tamanho dos elementos (xsd:string)
  - minLength
  - maxLength

O elemento **password** deve ter entre 6 e 8 caracteres (xsd:string):

```
<xsd:element name="password">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength value="6"/>
      <xsd:maxLength value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

## XSD: Elementos/atributos simples - **restriction**

Restrição	Descrição
enumeration	Define uma lista de valores aceitáveis
maxExclusive	Especifica o limite máximo de um elemento numérico (o valor deve ser menor do que o valor especificado). <b>Para xsd:integer</b>
maxInclusive	Especifica o limite máximo de um elemento numérico (o valor deve ser menor ou igual do que o valor especificado). <b>Para xsd:integer</b>
minExclusive	Especifica o limite mínimo de um elemento numérico (o valor deve ser maior do que o valor especificado). <b>Para xsd:integer</b>
minInclusive	Especifica o limite mínimo de um elemento numérico (o valor deve ser maior ou igual do que o valor especificado). <b>Para xsd:integer</b>
pattern	Define a sequência exacta de caracteres aceitáveis <b>Para xsd:string</b>
length	Especifica o número exacto de caracteres. Deve ser maior ou igual a zero. <b>Para xsd:string</b>
maxLength	Especifica o número máximo de caracteres. Deve ser maior ou igual a zero. <b>Para xsd:string</b>
minLength	Especifica o número mínimo de caracteres. Deve ser maior ou igual a zero. <b>Para xsd:string</b>
fractionDigits	Especifica o número máximo de casas decimais permitidas. Deve ser maior ou igual a zero. <b>Para xsd:double ou xsd:float</b>
totalDigits	Especifica o exacto nº de dígitos permitidos. Deve ser maior ou igual a zero. <b>Para tipos de dados numéricos</b>

## XSD: Exercício 1

```

<produtos>
  <prod id="x1" tipo="software">
    <nome>Anti virus</nome>
    <localizacao>Piso 001</localizacao>
    <localizacao>Sala 002</localizacao>
    <localizacao>Bloco 012</localizacao>
    <preco>100.90</preco>
    <preco_desc valor="0.1"/>
    <quant registo="2014-03-
10">102</quant>
  </prod>
  <prod id="x2" tipo="hardware">
    <nome>Macbook pro</nome>
    <localizacao>Piso 021</localizacao>
    <localizacao>Sala 012</localizacao>
    <preco>1030.90</preco>
    <quant registo="2014-03-
10">102</quant>
  </prod>
</produtos>

```

**id:** identificador único, obrigatório

**tipo:** enumeração (hardware, software), obrigatório

**localização:** padrão formado por palavra + 3 dígitos

**nome:** string

**preço:** número real

## XSD: Elementos complexos

---

- Tipos de dados **complexos**
  - Elemento vazio com atributos
  - Elemento textual (folha) com atributos
  - Elemento agregador com/sem atributos
  - Elemento de conteúdo misto com/sem atributos

## XSD: Elementos complexos

---

Exemplos de um elementos complexos:

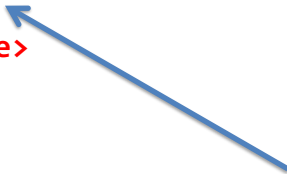
- Vazio com atributos  
`<produto id="003"/>`
- Textual com atributos:  
`<menu tipo="sobremesa">Pudim</menu>`
- Agregador:  
`<peessoa>`  
     `<nome>Anabela Simões</nome>`  
     `<email>abs@isec.pt</email>`  
`</peessoa>`
- Conteúdo Misto:  
`<desc>A docente chama-se <nome>Anabela Simões</nome> e ...`  
`</desc>`

## XSD: Elementos complexos

---

### Definição de elementos complexos:

```
<xsd:element name="...">
  <xsd:complexType>
    ...
  </xsd:complexType>
</xsd:element>
```



- dentro deste elemento define-se a **ordem** por que podem aparecer os **elementos filho** e quais os **atributos** associados

## XSD: Elementos complexos

---

```
<xsd:element name="NOME_ELEMENTO">
  <xsd:complexType>
    <xsd:attribute name="..." type="..." />
    ...
  </xsd:complexType>
</xsd:element>
```

VAZIO COM ATRIBUTOS

```
<xsd:element name="NOME_ELEMENTO">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="..." type="..." />
        ...
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

FOLHA COM ATRIBUTOS



## XSD: Elementos complexos

```
<xsd:element name="NOME_ELEMENTO">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="..." type="..." />
      <xsd:element name="..." type="..." />
      ...
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

AGREGADOR SEM  
ATRIBUTOS

```
<xsd:element name="NOME_ELEMENTO">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="..." type="..." />
      <xsd:element name="..." type="..." />
      ...
    </xsd:sequence>
    <xsd:attribute name="..." type="..." />
  </xsd:complexType>
</xsd:element>
```

AGREGADOR COM  
ATRIBUTOS

## XSD: Elementos complexos

### ○ Conteúdo MISTO (texto + outros elementos + atributos)

```
<xsd:element name="texto">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="nome" type="xsd:string"/>
      <xsd:element name="idade" type="xsd:positiveInteger"/>
      <xsd:element name="data" type="xsd:date"/>
    </xsd:sequence>
    <xsd:attribute name="ano" type="xsd:integer" />
  </xsd:complexType>
</xsd:element>
```

<nome>, <idade> e <data> têm  
de aparecer por esta ordem e  
apenas uma vez

```
<texto ano="1990">0 irmão do <nome>João</nome> tem
<idade>12</idade> anos e viajou pela primeira vez a
<data>2011-02-20</data></texto>
```

## XSD: Elementos complexos

### ○ Compositores de ordem

- Determinam como os elementos filho vão surgir no documento XML
- Três tipos de compositores
  - `<xsd:sequence>`
  - `<xsd:choice>`
  - `<xsd:all>`

`<xsd:sequence>` e `<xsd:choice>`

podem ser aninhados dentro de outros compositores

- Todos podem ter valores nas propriedades **minOccurs** e **maxOccurs**

## XSD: Elementos complexos

### ○ Ordem: sequência (sequence)

Indica quais os elementos filho de um elemento raiz e a sua ordem:

```
<xsd:element name="cliente">
  <xsd:complexType>
    <xsd:sequence maxOccurs="4">
      <xsd:element name="NIF" type="xsd:string" />
      <xsd:element name="idade" type="xsd:integer" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<cliente>
  <NIF>...</NIF>
  <idade>...</idade>
</cliente>
```

```
<cliente>
  <NIF>...</NIF>
  <idade>...</idade>
  <NIF>...</NIF>
  <idade>...</idade>
</cliente>
```

```
<cliente>
  <idade>...</idade>
</cliente>
```

## XSD: Elementos complexos

### ○ Ordem: sequência (sequence)

```
<xsd:element name="cliente">
  <xsd:complexType>
    <xsd:sequence maxOccurs="10">
      <xsd:element name="NIF" type="xsd:string"
        minOccurs="0" maxOccurs="4"/>

      <xsd:element name="idade" type="xsd:integer"
        minOccurs="0" maxOccurs="4"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<cliente>
  <NIF>...</NIF>
  <idade>...</idade>
</cliente>
```

```
<cliente>
  <idade>...</idade>
  <NIF>...</NIF>
</cliente>
```

```
<cliente>
  <idade>...</idade>
</cliente>
```

```
<cliente>
  <idade>...</idade>
  <NIF>...</NIF>
  <idade>...</idade>
  <NIF>...</NIF>
</cliente>
```

## XSD: Elementos complexos

### ○ Compositores: escolha/alternativa (choice)

- Apenas um dos elementos filho indicados pode fazer parte do elemento agregador.

```
<xsd:element name="pessoa">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="singular" type="xsd:string" />
      <xsd:element name="empresa" type="xsd:string" />
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

```
<pessoa>
  <singular>.....</singular>
</pessoa>
```

```
<pessoa>
  <empresa>.....</empresa>
</pessoa>
```

```
<pessoa>
  <singular>.....</singular>
  <empresa>.....</empresa>
</pessoa>
```

## XSD: Elementos complexos

### ○ Compositores: escolha/alternativa (choice)

#### ○ QUANTIFICADORES

```
<xsd:element name="pessoa">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="singular" type="xsd:string" maxOccurs="5"/>
      <xsd:element name="empresa" type="xsd:string" />
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

```
<pessoa>
  <singular>.....</singular>
  <singular>.....</singular>
</pessoa>
```

```
<pessoa>
  <empresa>.....</empresa>
</pessoa>
```

```
<pessoa>
  <singular>.....</singular>
  <empresa>.....</empresa>
</pessoa>
```

## XSD: Elementos complexos

### ○ Compositores: escolha/alternativa (choice)

Usando QUANTIFICADORES no choice, todos os elementos podem aparecer em qualquer ordem até ao valor quantificado

```
<xsd:element name="pessoa">
  <xsd:complexType>
    <xsd:choice maxOccurs="10">
      <xsd:element name="singular" type="xsd:string" />
      <xsd:element name="empresa" type="xsd:string" />
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

```
<pessoa>
  <singular>.....</singular>
</pessoa>
```

```
<pessoa>
  <empresa>.....</empresa>
</pessoa>
```

```
<pessoa>
  <singular>.....</singular>
  <empresa>.....</empresa>
</pessoa>
```

```
<pessoa>
  <singular>.....</singular>
  <empresa>.....</empresa>
  <singular>.....</singular>
  <empresa>.....</empresa>
  <empresa>.....</empresa>
</pessoa>
```

## XSD: Elementos complexos

### ○ Compositores de ordem: todos (all)

Elementos têm de aparecer todos, por qualquer ordem e com um número de ocorrências **igual a um**

```
<xsd:element name="pessoa">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="nome" type="xsd:string"/>
      <xsd:element name="apelido" type="xsd:string"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

```
<pessoa>
  <nome>...</nome>
  <apelido>...</apelido>
</pessoa>
```

```
<pessoa>
  <apelido>...</apelido>
  <nome>...</nome>
</pessoa>
```

```
<pessoa>
  <apelido>...</apelido>
</pessoa>
```

## XSD: Elementos complexos

### ○ Compositores de ordem: todos (all)

- Quantificadores: aceita apenas com valores 0 ou 1

```
<xsd:element name="pessoa">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="nome" type="xsd:string" minOccurs="0"/>
      <xsd:element name="apelido" type="xsd:string" minOccurs="0"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

```
<pessoa>
</pessoa>
```

```
<pessoa>
  <apelido>...</apelido>
</pessoa>
```

```
<pessoa>
  <nome>...</nome>
</pessoa>
```

```
<pessoa>
  <apelido>...</apelido>
  <nome>...</nome>
</pessoa>
```

```
<pessoa>
  <nome>...</nome>
  <apelido>...</apelido>
</pessoa>
```

## XSD: Elementos complexos

- Compositores de ordem: todos (all)

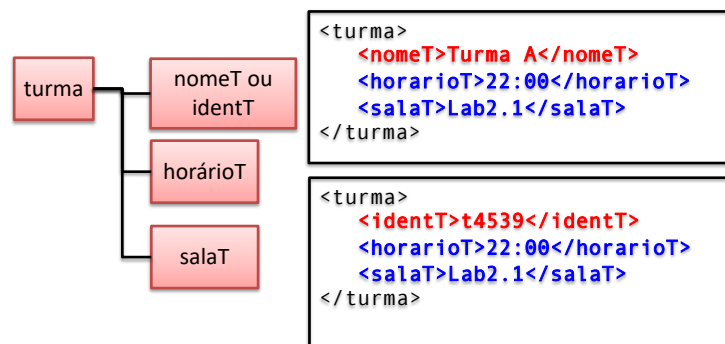
- Quantificadores: aceita apenas com valores 0 ou 1

```
<xsd:element name="pessoa">
  <xsd:complexType>
    <xsd:all minOccurs="20">
      <xsd:element name="nome" type="xsd:string" />
      <xsd:element name="apelido" type="xsd:string" minOccurs="10"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

**XSD ERRADO!**

## XSD: Elementos complexos

- Misturar compositores: <xsd:sequence> e <xsd:choice>



## XSD: Elementos complexos

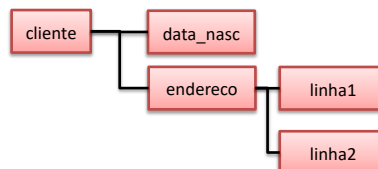
- Misturar compositores: `<xsd:sequence>` e `<xsd:choice>`

```
<xsd:element name="turma">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:choice>
        <xsd:element name="nomeT" type="xsd:string" />
        <xsd:element name="identT" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="horarioT" type="xsd:string" />
      <xsd:element name="salaT" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

## XSD: Elementos complexos

- Misturar compositores: `<xsd:sequence>`

```
<xsd:element name="cliente">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="data_nasc" type="xsd:date" />
      <xsd:element name="endereco">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="linha1" type="xsd:string" />
            <xsd:element name="linha2" type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```



## XSD: Elementos complexos

### ○ ref

- permite dividir o XSD de forma a que este possa ser reutilizado de forma mais versátil
- Separar os elementos/atributos simples dos complexos
- Referenciar os elementos/atributos simples sempre que necessário

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

  <xsd:element name="nome" type="xsd:string"/>
  <xsd:attribute name="tipo" type="xsd:string"/>

  <xsd:element name="nomes">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="nome" maxOccurs="5"/>
      </xsd:sequence>
      <xsd:attribute ref="tipo" use="required"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

## Exemplo – catálogo de cds

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalogo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="catalogo.xsd">
  <cd pais="UK">
    <titulo>Dark Side of the Moon</titulo>
    <artista>Pink Floyd</artista>
    <preco>10.90</preco>
  </cd>
  <cd pais="UK">
    <titulo>Space Oddity</titulo>
    <artista>David Bowie</artista>
    <preco>9.90</preco>
  </cd>
  <cd pais="UK">
    <titulo>Aretha: Lady Soul</titulo>
    <artista>Aretha Franklin</artista>
    <preco>9.90</preco>
  </cd>
</catalogo>
```



## Exemplo: XSD SEM usar **ref**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="catalogo">
    <xsd:complexType>
      <xsd:sequence maxOccurs="10">
        <xsd:element name="cd">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="titulo" type="xsd:string"/>
              <xsd:element name="artista" type="xsd:string"/>
              <xsd:element name="preco" type="xsd:double"/>
            </xsd:sequence>
            <xsd:attribute name="pais" type="xsd:string" use="required"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

## Exemplo: XSD usando **ref**

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="titulo" type="xsd:string"/>
  <xsd:element name="artista" type="xsd:string"/>
  <xsd:element name="preco" type="xsd:double"/>
  <xsd:attribute name="pais" type="xsd:string"/>

  <xsd:element name="cd">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="titulo"/>
        <xsd:element ref="artista"/>
        <xsd:element ref="preco" />
      </xsd:sequence>
      <xsd:attribute ref="pais" use="required"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="catalogo">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="cd" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

## XSD: Exercício 1 (complete o XSD para os elementos complexos)

```
<produtos>
  <prod id="x1" tipo="software">
    <nome>Anti virus</nome>
    <localizacao>Piso 001</localizacao>
    <localizacao>Sala 002</localizacao>
    <localizacao>Bloco 012</localizacao>
    <preco>100.90</preco>
    <preco_desc valor="0.1"/>
    <quant registo="2014-03-10">102</quant>
  </prod>
  <prod id="x2" tipo="hardware">
    <nome>Macbook pro</nome>
    <localizacao>Piso 021</localizacao>
    <localizacao>Sala 012</localizacao>
    <preco>1030.90</preco>
    <quant registo="2014-03-10">102</quant>
  </prod>
</produtos>
```

**id:** identificador único, obrigatório

**tipo:** enumeração (hardware, software), valor por defeito "software" obrigatório

**localização:** padrão formado por palavra + 3 dígitos

**nome:** string

**preço:** número real

## Exercício 2 – construa o XSD

```
<raiz>
  <misto>Elemento de nome <nome>qualquer</nome> que aparece
    <quant>1</quant> vez</misto>
  <nome>www</nome>
  <agrega1 at="xxx">
    <nome>zzz</nome>
    <nome>zzz</nome>
    <nome>zzz</nome>
    <agrega2 at="yyy">
      <nome>zzz</nome>
      <quant>1</quant>
      <quant uni="stock">2</quant>
    </agrega2>
    <agrega2 at="zzz">
      <nome>kkk</nome>
      <quant>10</quant>
      <quant uni="stock">20</quant>
      <quant>20</quant>
    </agrega2>
  </agrega1>
</raiz>
```

**<nome>**  
 1 vez em <raiz>  
 1 vez em <agrega2>  
 1-10 vezes em <agrega1>  
**<quant>**  
 1-10 vezes em <agrega2>

## Exercício 3 – ficheiro animais.xml

```
<animais xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="animais.xsd">
  <carnivoros cat="pag10">
    <animal id="a001">
      <nome>lobo</nome>
      <tipo>mamifero</tipo>
    </animal>
    <animal id="a002">
      <nome>águia</nome>
      <tipo>ave</tipo>
    </animal>
    ...
  </carnivoros>
  <herbivoros cat="pag2">
    <animal id="a004">
      <nome>coelho</nome>
      <tipo>mamifero</tipo>
    </animal>
    ...
  </herbivoros>
</animais>
```

**atributo tipo:**  
enumeração  
mamifero  
ave  
reptil

## Exercício 3 – animais.xsd

```
<xsd:schema ...>
  <xsd:element name="nome" type="xsd:string"/>
  <xsd:attribute name="id" type="xsd:ID"/>
  <xsd:attribute name="cat" type="xsd:string"/>

  <xsd:element name="tipo">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="mamifero"/>
        <xsd:enumeration value="reptil"/>
        <xsd:enumeration value="ave"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>

  <xsd:element name="animal">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="nome"/>
        <xsd:element ref="tipo"/>
      </xsd:sequence>
      <xsd:attribute ref="id" use="required"/>
    </xsd:complexType>
  </xsd:element>
```

## Exercício 3 – animais.xsd (cont)

---

```
<xsd:element name="carnivoros">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="animal" maxOccurs="4"/></xsd:element>
    </xsd:sequence>
    <xsd:attribute ref="cat" use="required"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="herbivoros">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="animal" maxOccurs="unbounded"/></xsd:element>
    </xsd:sequence>
    <xsd:attribute ref="cat" use="required"/>
  </xsd:complexType>
</xsd:element>
```

## Exercício 3 – animais.xsd (cont)

---

```
<xsd:element name="animais">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="carnivoros" maxOccurs="unbounded"/></xsd:element>
      <xsd:element ref="herbivoros" maxOccurs="unbounded"/></xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

</xsd:schema>
```

## XSD: Tipos de dados

---

- Além dos Tipos de dados fornecidos (string, double, integer, ....), o XSD permite criar novos tipos de dados:
  - Simples

```
<xsd:simpleType name = "nomeTipo">
...
</xsd:simpleType>
```
  - Complexos

```
<xsd:complexType name = "nomeTipo">
...
</xsd:complexType>
```

## XSD: Tipos de dados

---

- Porquê criar novos tipos de dados?
  - reaproveitar código para elementos similares
  - criar restrições/extensões de tipos de dados

## XSD: Tipos de dados SIMPLES

---

- Definir um elemento simples através de um tipo de dados.

Duas etapas

- 1) Definir o tipo de dados com:

```
<xsd:simpleType name="tipoN">
    ...
</xsd:simpleType>
```

- 2) Usar o tipo definido para criar elementos

```
<xsd:element name="nome_elemento1" type="tipoN"/>
<xsd:element name="nome_elemento2" type="tipoN"/>
...
```

## XSD: Tipos de dados COMPLEXOS

---

- Definir um elemento complexo através de um tipo de dados.

Duas etapas

- 1) Definir o tipo de dados com:

```
<xsd:complexType name="tipoN">
    ...
</xsd:complexType>
```

- 2) Usar o tipo definido para criar vários elementos:

```
<xsd:element name="nome_elemento1" type="tipoN"/>
<xsd:element name="nome_elemento2" type="tipoN"/>
...
```

## XSD: Tipos de dados SIMPLES

- **Exemplo:** existem **quatro** elementos simples que são uma restrição de número de telemóvel. Números iniciados por 91, 92, 93 ou 96 e com 9 dígitos

```
<contacto>918191888</contacto>

<telemovel>968885553</telemovel>

<telefone>934449992</telefone>

<mobile>928882229</mobile>
```

## XSD: Tipos de dados SIMPLES

### SOLUÇÃO SEM USAR TIPOS DE DADOS

```
<xsd:element name="contacto">
  <xsd:simpleType>
    <xsd:restriction base = "xsd:string">
      <xsd:pattern value = "9[1236][0-9]{7}" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="telemovel">
  <xsd:simpleType>
    <xsd:restriction base = "xsd:string">
      <xsd:pattern value = "9[1236][0-9]{7}" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

## XSD: Tipos de dados SIMPLES

### SOLUÇÃO SEM USAR TIPOS DE DADOS

```
<xsd:element name="telefone">
  <xsd:simpleType>
    <xsd:restriction base = "xsd:string">
      <xsd:pattern value = "9[1236][0-9]{7}" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="mobile">
  <xsd:simpleType>
    <xsd:restriction base = "xsd:string">
      <xsd:pattern value = "9[1236][0-9]{7}" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

## XSD: Tipos de dados SIMPLES

### SOLUÇÃO COM TIPOS DE DADOS

```
<xsd:simpleType name = "t1mTIPO">
  <xsd:restriction base = "xsd:string">
    <xsd:pattern value = "9[1236][0-9]{7}" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:element name="contacto" type = "t1mTIPO"/>
<xsd:element name="telemovel" type = "t1mTIPO"/>
<xsd:element name="telefone" type = "t1mTIPO"/>
<xsd:element name="mobile" type = "t1mTIPO"/>
```

criar tipo de dados

criar elementos



## XSD: Tipos de dados COMPLEXOS

- **Exemplo:** existem **quatro** elementos complexos que são uma sequência de **nome**, **telefone** e **morada**

<pre>&lt;cliente&gt;   &lt;nome&gt;...&lt;/nome&gt;   &lt;morada&gt;...&lt;/morada&gt;   &lt;telefone&gt;...&lt;/telefone&gt; &lt;/cliente&gt;</pre>	<pre>&lt;funcionario&gt;   &lt;nome&gt;...&lt;/nome&gt;   &lt;morada&gt;...&lt;/morada&gt; &lt;/funcionario&gt;</pre>
<pre>&lt;fornecedor&gt;   &lt;nome&gt;...&lt;/nome&gt;   &lt;morada&gt;...&lt;/morada&gt;   &lt;telefone&gt;...&lt;/telefone&gt; &lt;/fornecedor&gt;</pre>	<pre>&lt;empresa&gt;   &lt;nome&gt;...&lt;/nome&gt;   &lt;morada&gt;...&lt;/morada&gt;   &lt;telefone&gt;...&lt;/telefone&gt;   &lt;telefone&gt;...&lt;/telefone&gt;   &lt;telefone&gt;...&lt;/telefone&gt; &lt;/empresa&gt;</pre>

## XSD: Tipos de dados COMPLEXOS

### SOLUÇÃO SEM USAR TIPOS DE DADOS

```
<xsd:element name="cliente">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="nome" type="xsd:string" />
      <xsd:element name="morada" type="xsd:string" />
      <xsd:element name="telefone" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="fornecedor">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="nome" type="xsd:string" />
      <xsd:element name="morada" type="xsd:string" />
      <xsd:element name="telefone" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

## XSD: Tipos de dados COMPLEXOS

### SOLUÇÃO SEM USAR TIPOS DE DADOS

```
<xsd:element name="funcionario">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="nome" type="xsd:string" />
      <xsd:element name="morada" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="empresa">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="nome" type="xsd:string" />
      <xsd:element name="morada" type="xsd:string" />
      <xsd:element name="telefone" type="xsd:string" maxOccurs="10"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

## XSD: Tipos de dados COMPLEXOS

### SOLUÇÃO COM TIPOS DE DADOS

```
<xsd:complexType name="TipoPessoa">
  <xsd:sequence>
    <xsd:element name="nome" type="xsd:string" />
    <xsd:element name="morada" type="xsd:string" />
    <xsd:element name="telefone" type="xsd:string"
      minOccurs="0" maxOccurs="10"/>
  </xsd:sequence>
</xsd:complexType>
```

criar tipo de dados

```
<xsd:element name="cliente" type="TipoPessoa"/>
<xsd:element name="fornecedor" type="TipoPessoa"/>
<xsd:element name="funcionario" type="TipoPessoa"/>
<xsd:element name="empresa" type="TipoPessoa"/>
```

criar elementos

## XSD: Estender Tipos de dados

### ○ Estender tipos de dados

#### Exemplo 1 (simples):

```
<contacto1>919233339</contacto1>
<contacto2 tipo="t1m">919233339</contacto2>
```

#### Exemplo 2 (complexo):

<pre>&lt;EUAEndereco&gt;   &lt;linha1&gt;234 Lancaster Av&lt;/linha1&gt;   &lt;linha2&gt;Smallville&lt;/linha2&gt;   &lt;state&gt;Florida&lt;/state&gt;   &lt;zipcode&gt;34543&lt;/zipcode&gt; &lt;/EUAEndereco&gt;</pre>	<pre>&lt;UKEndereco&gt;   &lt;linha1&gt;34 thingy street&lt;/linha1&gt;   &lt;linha2&gt;someplace&lt;/linha2&gt;   &lt;county&gt;Somerset&lt;/county&gt;   &lt;postcode&gt;w1w8uu&lt;/postcode&gt; &lt;/UKEndereco&gt;</pre>
---	--

## XSD: Estender Tipos de dados SIMPLES

```
<xsd:simpleType name = "t1mTIPO">
  <xsd:restriction base = "xsd:string">
    <xsd:pattern value = "9[1236][0-9]{7}" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:element name="contacto1" type = "t1mTIPO">

<xsd:element name="contacto2">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base = "t1mTIPO">
        <xsd:attribute ref="tipo"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

criar tipo de dados

criar elementos  
contacto1 é uma  
restrição

criar elementos  
contacto2 é uma  
extensão de uma  
restrição

## XSD: Estender Tipos de dados COMPLEXOS

- Criar o tipo de dados base
  - o que é comum aos dois endereços?

```
<xsd:complexType name="TipoEndereco">
  <xsd:sequence>
    <xsd:element name="linha1" type="xsd:string"/>
    <xsd:element name="linha2" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

## XSD: Estender Tipos de dados COMPLEXOS

- Criar ELEMENTO **UKEndereco** como Extensão

```
<xsd:element name="UKEndereco">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="TipoEndereco">
        <xsd:sequence>
          <xsd:element name="county" type="xsd:string"/>
          <xsd:element name="postcode" type="xsd:string"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

Tipo existente que vai ser estendido

Necessário para indicar que se vai alterar um tipo de dados Complexo

Elementos adicionais

## XSD: Estender Tipos de dados COMPLEXOS

- Criar ELEMENTO **EUAEndereco** como Extensão

```

<xsd:element name="EUAEndereco">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="TipoEndereco">
        <xsd:sequence>
          <xsd:element name="state" type="xsd:string"/>
          <xsd:element name="zipcode" type="xsd:string"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

```

Necessário para indicar que se vai alterar um tipo de dados Complexo

Tipo existente que vai ser estendido

Elementos adicionais

## XSD: Estender Tipos de dados COMPLEXOS

- Em vez de criar elementos como extensões podem criar-se **NOVOS** tipos de dados como extensões

```

<xsd:complexType name="EUAEnderecoType">
  <xsd:complexContent>
    <xsd:extension base="TipoEndereco">
      <xsd:sequence>
        <xsd:element name="state" type="xsd:string"/>
        <xsd:element name="zipcode" type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```

Criar os elementos usando os novos tipos de dados:

```

<xsd:element name="UKEndereco" type="UKEnderecoType"/>
<xsd:element name="EUAEndereco" type="EUAEnderecoType"/>

```

### Exercício: criar dois novos Tipos de Dados -> divisaoT e andarT

```
<casa>
  <primeiro_andar id="d1" >
    <divisao>
      <nome>Sala</nome>
      <area>60</area>
      <movel>Sofa</movel>
      <movel>Mesa</movel>
      <movel>Cadeira</movel>
      <movel>Tapete</movel>
    </divisao>
    <divisao>
      <nome>Cozinha</nome>
      <area>40</area>
      <eletrodomestico>Forno</eletrodomestico>
      <eletrodomestico>Frigorifico</eletrodomestico>
      <eletrodomestico>Microondas</eletrodomestico>
    </divisao>
    <divisao>
      ...
    </divisao>
  </primeiro_andar>
  <segundo_andar id="d2" >
    <desc>Segundo andar renovado em 2011</desc>
    <espaco>
      <nome>Quarto</nome>
      <area>30</area>
      <movel>Cama</movel>
      <movel>C moda</movel>
      <movel>Tapete</movel>
    </espaco>
    <espaco>
      <nome>Quarto</nome>
      <area>35</area>
      <movel>Cama</movel>
      <movel>C moda</movel>
      <movel>Tapete</movel>
    </espaco>
    ...
  </segundo_andar>
</casa>
```

### XSD: Solu  o

```
....
<xsd:complexType name="divisaoT">
  <xsd:sequence>
    <xsd:element ref="nome"/>
    <xsd:element ref="area"/>
    <xsd:element ref="movel" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="eletrodomestico" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="andarT">
  <xsd:sequence>
    <xsd:element name="desc" type="xsd:string" minOccurs="0"/>
    <xsd:choice maxOccurs="unbounded">
      <xsd:element ref="divisao"/>
      <xsd:element ref="espaco"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute ref="id" use="required"/>
</xsd:complexType>

<xsd:element name="divisao" type="divisaoT"/>
<xsd:element name="espaco" type="divisaoT"/>
<xsd:element name="primeiro_andar" type="andarT"/>
<xsd:element name="segundo_andar" type="andarT"/>
```