XML Schema

Carlos A. Heuser
UFRGS

XML Schema

- Como DTD:
  - Utilizado para descrever a **estrutura** de um documento XML
- Utiliza **sintaxe XML**
- Sintaxe simples: fácil compreensão humana (?)
- **Aumenta o poder de expressão** da DTD

XML Schema

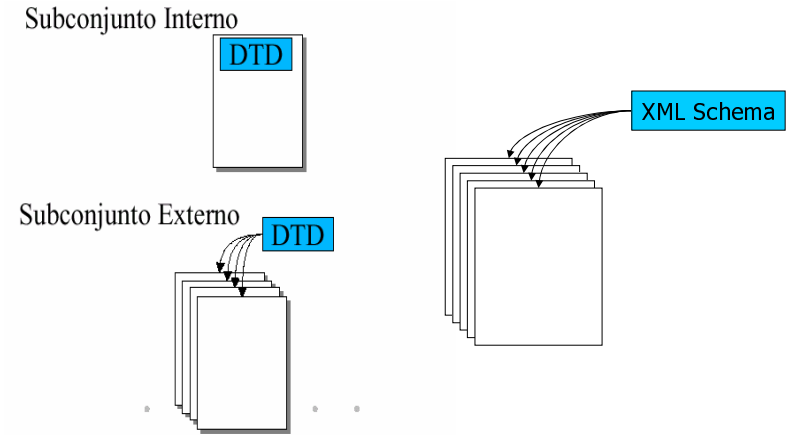
- **Como em DTD**, define:
  - Elementos e atributos que aparecem em um documento
  - Aninhamento de elementos
  - Ordem dos elementos
  - Número de elementos ("cardinalidade")
  - Se um elemento é vazio ou pode incluir texto
- **Novidades:**
  - Define **tipos de dados** (*data types*) para elementos e atributos
  - Define valores ***default*** e **fixos** para atributos ***e elementos***
  - Suporta ***namespaces***
  - Sintaxe é **XML**
  - **Tipos são extensíveis**

Declaração

- Ao contrário de DTD, em XMLSchema:
  - Todas as declarações (elementos, atributos, etc.) são feitas **externamente** ao documento XML
  - Não existe declaração interna (ou subconjunto interno)
  - Sufixo usual do arquivo que contém XMLSchema é .**xsd**

Declaração

- DTD
- XML Schema



Declaração de esquema no documento XML

Referência a DTD através da declaração **DOCTYPE**

```
<?xml version="1.0"?>
<!DOCTYPE artigo SYSTEM "artigo.dtd">
<artigo versao="1.3">
   <titulo> ... ... </titulo>
   <autor>
      <nome> ... ... </nome>
   </autor>
</artigo>
```


Declaração de esquema no documento XML

Chamada da DTD

```
<?xml version="1.0"?>
<!DOCTYPE artigo SYSTEM "artigo.dtd">
<artigo versao="1.3">
   <titulo> ... ... </titulo>
   <autor>
      <nome> ... ... </nome>
```

Referência a XML Schema (documento **sem *namespace***)

```
<?xml version="1.0"?>
<artigo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="artigo.xsd"
        versao="1.3">
 <titulo> ... ... </titulo>
   <autor>
      <nome> ... ... </nome>
   </autor>
<artigo>
```


Declaração de esquema no documento XML

Chamada da DTD

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

Referência a XML Schema (documento **com *namespace***)

```
<?xml version="1.0"?>
<artigo xmlns="http://www.my.com"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.my.com c:\artigo.xsd"
        versao="1.3">
 <titulo> ... ... </titulo>
   <autor>
      <nome> ... ... </n
   </autor>
<artigo>
```

Observar que o *namespace* aparece seguido da localização do arquivo que contém o XMLSchema

Estrutura de um arquivo XML Schema

- Elemento raiz de todo esquema:

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

- Extensão do arquivo: **.xsd**

Alguns atributos de <schema>

```
<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.domain.com"
            elementFormDefault="qualified">

<!- definições de elementos

</xs:schema>
```

Define o *namespace* do XML Schema

Alguns atributos de <schema>

```
<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.domain.com"
            elementFormDefault="qualified">

<!- definições de elem                tributos ->

</xs:schema>
```

Especifica o *namespace* dos nomes dados aos elementos e atributos do esquema sendo definido

Alguns atributos de <schema>

```
<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.domain.com"
            elementFormDefault="qualified">

<!- definições de elem                tributos ->

</xs:schema>
```

Quaisquer elementos usados em instâncias XML (docs.xml) devem ser qualificados pelo *namespace*, ou seja, devem possuir o prefixo do *namespace*

Definição de Elemento

- Uso da tag **xs:element**
  - Cria um elemento
    Através do atributo **name**
  - Associa o elemento a um tipo
    Através do atributo **type**

- Exemplo:
  - Definição de um elemento atômico

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


Definição de Elemento

- Na **DTD**:

```
<!ELEMENT rua (#PCDATA)>
<!ELEMENT nr (#PCDATA)>
```

- No **XMLSchema**:

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


Valores *default* e fixo para elementos

- ***Default***:
  - Uso do atributo *default*

```
<xs:element name="cor" type="xs:string" default="azul"/>
```

- **Fixo**:
  - Uso do atributo *fixed*

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


Tipos básicos

- Elementos atômicos podem ser:
  - xs:string
  - xs:decimal
  - xs:integer
  - xs:boolean
  - xs:date
  - xs:time

Declarações de tipos

- Há duas formas de declarar tipos:
  - **<xs:complexType>**
    Serve para declarar **elementos compostos** por outros elementos.
  - **<xs:simpleType>**
    Serve para declarar tipos de **elementos atômicos**, isto é, elementos que não são compostos por outros elementos.

Declaração de tipo de elemento complexo

- Uso de **<xs:complexType>**
- Declara tipo de elemento que pode **conter outros elementos e/ou atributos** (elemento **complexo**)

Elemento complexo – declaração de tipo com definição de elemento

```
<xs:element name="endereco">
  <xs:complexType>
     <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>
```


Comparação com DTD

- **XMLSchema**:

```
<xs:element name="endereco">
  <xs:complexType>
     <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>
```

- **DTD**:

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

Estilos de declaração de tipos

- Declaração pode ser feita de várias formas
  - **Separadamente** da definição de elementos do tipo, ou
  - **Conjuntamente** com a definição de elementos do tipo
- Os tipos pode ter nomes (podem ser usados para definir vários elementos) ou não.

Elemento complexo – declaração de tipo separada

- **Declaração** do tipo (**tipo com nome**)

```
<xs:complexType name="tEndereco">
  <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>
```

- Definição de **elemento do tipo**

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


Declarações locais vs. globais

- Pode-se declarar tipos, elementos e atributos globais ou locais
- Declaração **global**:
  - Declarada como filha do elemento **<schema>**
  - Tipos declarados são acessíveis em todo esquema
- Declaração **local**:
  - Dentro da declaração de um elemento
  - Tipo declarado acessível somente no escopo do elemento

Exemplo de declaração local

```
<xs:schema>
  <xs:element name="endereco">
    <xs:complexType>
      <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>
</xs:schema>
```

Tipo e elementos podem ser usados apenas para o elemento endereço

Exemplo de declaração global

```
<xs:schema>
  <xs:complexType name="tEndereco">
      <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="tEndereco"/>
</xs:schema>
```

Tipo pode ser usado como tipo de outros elementos

Referências a elementos

- Pode-se referenciar um elemento que tenha sido declarado anteriormente
- Elemento referenciado deve ser **global**
  - declarado no nível logo abaixo de **<schema>**

```
<!-- Declarações dos elementos -->
<xs:element name="rua" type="xs:string"/>
<xs:element name="numero" type="xs:integer"/>
<xs:element name="cidade" type="xs:string"/>
<!-- Declaração do elemento -
                contém referências a outros elementos -->
<xs:element name="endereco">
  <xs:complexType>
     <xs:sequence>
       <xs:element ref="rua"/>
        <xs:element ref="numero"/>
       <xs:element ref="cidade"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>
```


Restrições no *complexType*

- Especificação de **cardinalidade**
  - xs:minOccurs
  - xs:maxOccurs
- Especificação da **forma de combinação** dos elementos
  - <xs:sequence>
  - <xs:choice>
  - <xs: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

```
<xs:complexType name="tArtigo">
  <xs:sequence>
       <xs:element name="titulo" type="xs:string"/>
       <xs:element name="autor" type="TAutor"/>
       <xs:element name="ano" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="tAutor">
  <xs:sequence>
       <xs:element name="nome" type="xs:string"
                maxOccurs="unbounded"/>
       <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 JUNTOS ou nenhuma vez e podem aparecer em qualquer ordem

Choice - exemplo

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


Choice - exemplo

```
<xs:element name="publicacao">
  <xs:complexType>
    <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>
```

- Possíveis instâncias:

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

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

Choice – comparando com a DTD

- No XML Schema:

```
<xs:element name="publicacao">
 <xs:complexType>
   <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>
```

- Na DTD:

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


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 instância XML**:

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

Todos juntos ou nada
Sem restrição de ordem

Elementos de conteúdo misto

- Especificado através do uso do atributo **mixed**

```
<xs:element name="paragrafo">
  <xs:complexType  mixed="true">
    <xs:sequence>
       <xs:element name="ingles" type="xs:string"/>
       <xs:element name="enfase" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```


Elementos de conteúdo misto

- Especificado através do uso do atributo **mixed**

```
<xs:element name="paragrafo">
  <xs:complexType  mixed="true">
    <xs:sequence>
       <xs:element name="ingles" type="xs:string"/>
       <xs:element name="enfase" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- Exemplo de instância:

```
<paragrafo>é parágrafo em formato
  <ingles>default</ingles> com uma palavra
  <enfase>enfatizada</enfase></paragrafo>
```

Exercício 7

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

```
<notaFiscal>
   <cliente>
      <razao_social>JOAQUIM</razao_social>
      <cgc>00.000.000/0001-00</cgc>
        <obs>esta observação é sobre <sobre>clientes
especiais</sobre>
        </obs>
   </cliente>
   <item>
      <produto>caneta azul</produto>
      <quantidade>100</quantidade>
      <preco_unit>2</preco_unit>
   </item>
</notaFiscal>
```


Exercício 8

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

```
<itens_nota>
  <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_nota>
```


Exercício 9

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

```
<cliente>
  <razao_social>JOAQUIM</razao_social>
  <cgc>00.000.000/0001-00</cgc>
</cliente>
```

ou

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


Atributos

- Atributos são 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

Atributos

- Uso do elemento <xs:attribute>

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

- Exemplo de um atributo local

```
<xs:schema>
  <xs:complexType name="tEndereco">
       <xs:sequence>
              <xs:element name="rua" type="xs:string" />
       </xs:sequence>
       <xs:attribute name="numero" type="xs:integer"/>
  </xs:complexType>

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


Atributos

- Obrigatoriedade e opcionalidade
- Uso do atributo **<xs:use>**

```
<xs:attribute name="cor" type="xs:string"
  use="required"/>
```

- Use: optional | required
  - required: obrigatório
  - optional: opcional – default!!

Valores default e fixo para atributos

- Default:
  - Uso do atributo default

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

- Fixo:
  - Uso do atributo fixed

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


Exercício 10

- Crie um esquema completo para o documento abaixo

```
<pedido numero="1001">
  <cliente>
    <razao_social>JOAQUIM</razao_social>
    <cgc>00.000.000/0001-00</cgc>
  </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>
  <pago/>
</pedido>
```

Elementos vazios

- Por exemplo, declaração de elemento somente com atributo:

```
<xs:element name="figura">
  <xs:complexType>
      <xs:attribute name="nome">
  </xs:complexType>
<xs:element>
```

- Instancia XML:

```
<figura nome="foto"/>
```


Tipos de átomos - *simpleType*

- **<xs:list>**

- **<xs:union>**

list

- Declarados através do elemento **<xs:simpleType>**

- Atributo **name** define o **nome do tipo**

- Usado para declarar uma **lista** de elementos de um dos tipos básicos

Exemplo de *list*

- **Declaração** do tipo:

```
<xs:simpleType name="tListInt">
   <xs:list itemType="xs:integer"/>
</xs:simpleType>
```

Exemplo de *list*

- **Declaração** do tipo:

```
<xs:simpleType name="tListInt">
   <xs:list itemType="xs:integer"/>
</xs:simpleType>
```

- Definição de um **elemento de tipo** TListInt:

```
<xs:element name="lista"   type="tListInt"/>
```


Exemplo de *list*

- **Declaração** do tipo:

```
<xs:simpleType name="tListInt">
   <xs:list itemType="xs:integer"/>
</xs:simpleType>
```

- Definição de um **elemento de tipo** TListInt:

```
<xs:element name="lista"   type="tListInt"/>
```

- **Instância** do tipo:

```
<lista>20003 15037 95977 95945</lista>
```


union

- Declarados através do elemento `<xs:simpleType>`
- Atributo `name` define o **nome** do tipo
- Usado para declarar um elemento atômico cujo conteúdo pode ser um elemento de um tipo atômico **ou** de outro tipo atômico
  - **União** dos tipos atômicos forma um novo tipo atômico

Exemplo de *union*

- **Declaração** do tipo:

```
<xs:simpleType name="tUniao">
  <xs:union memberTypes="tCep tListaInteiro"/>
</xs:simpleType>
```

Exemplo de *union*

- **Declaração** do tipo:

```
<xs:simpleType name="tUniao">
  <xs:union memberTypes="tCep tListaInteiro"/>
</xs:simpleType>
```

- Definição de um **elemento do tipo**:

```
<xs:element name="CEPS" type="tUniao"/>
```


Exemplo de *union*

- **Declaração** do tipo:

```
<xs:simpleType name="tUniao">
  <xs:union memberTypes="tCep tListaInteiro"/>
</xs:simpleType>
```

- Definição de um **elemento do tipo**:

```
<xs:element name="CEPS" type="tUniao"/>
```

- Algumas **instâncias** XML válidas (valor = escolha de um dos tipos do *union*)

```
<CEPS>93320-005</CEPS>
<CEPS>95630 95977 95945</CEPS>
<CEPS>90000-240</CEPS>
```


Derivação de Tipos Simples

- Tipos simples
  - Derivados de tipos simples através de **restrição**.
- Processo:
  - Um tipo simples é usado com base sobre ele são aplicadas
    - **facetas** ou
    - **expressões regulares**.

Facetas

- Faceta **"exclusive"**
  - Limita um tipo simples a valores mínimo e máximo

```
<xs:simpleType name="tNumero">
  <xs:restriction base="xs:integer">
    <xs:minExclusive value="0"/>
    <xs:maxExclusive value="99999"/>
  </xs:restriction>
</xs:simpleType>
```

Facetas

- A faceta **"enumeration"**
  - Limita um tipo simples a um conjunto de valores distintos

```
<xs:simpleType name="TipoFigura">
  <xs:restriction base="xs:string">
    <xs:enumeration value = "jpeg"/>
    <xs:enumeration value = "gif"/>
    <xs:enumeration value = "bmp"/>
    <xs:enumeration value = "tiff"/>
    <xs:enumeration value = "wmf"/>
  </xs:restriction>
</xs:simpleType>
<xs:attribute name="tipo" type="TipoFigura">
```


Facetas

- Comparando o enumeration no XMLSchema com a DTD:

** XMLSchema

```
<xs:simpleType name="TipoFigura">
  <xs:restriction base="xs:string">
    <xs:enumeration value = "jpeg"/>
    <xs:enumeration value = "gif"/>
    <xs:enumeration value = "bmp"/>
    <xs:enumeration value = "tiff"/>
    <xs:enumeration value = "wmf"/>
  </xs:restriction>
</xs:simpleType>
<xs:attribute name="tipo" type="TipoFigura">
```

** DTD (só é possível delimitar valores para atributos)

```
<!ELEMENT figura ANY>
<!ATTLIST figura
    tipo NOTATION (jpeg|gif|bmp|tiff|wmf)>
```


Expressões regulares

- Faceta "pattern"
  - Possibilita a criação de máscaras para os valores

```
<xs:simpleType name="tCep">
  <xs:restriction base="xs:string">
    <xs:pattern value="\d{5}-\d{3}"/>
  </xs:restriction>
</xs:simpleType>

....

<xs:simpleType>
  <xs:restriction base="xs:string">
      <xs:pattern value="male|female"/>
  </xs:restriction>
</xs:simpleType>
```


Exercício 11 e 12

- 11. Crie um tipo tCGC para aceitar somente CGCs como o seguinte formato:

```
<cgc>00.000.000/0001-00</cgc>
```

- 12. Crie um tipo tCPF para aceitar somente CPFs com o seguinte formato:

```
<cpf>000.000.000-00</cpf>
```

Derivação de Tipos Complexos

- Tipos complexos podem ser derivados por restrição ou por extensão:
  - **Restrição**
    semelhante a restrição de tipos simples, mas ao invés de restringir valores, ela restringe elementos (por exemplo, cardinalidade)
  - **Extensão**
    utilizada para "aumentar" um tipo
    o novo tipo derivado possuirá tudo que o tipo base possuía, mais outros elementos e atributos definidos na extensão

Exemplo - Derivação por Extensão

- Tipo complexo a ser estendido:

```
<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:sequence>
</xs:complexType>
```


Exemplo - Derivação por Restrição

- Tipo complexo a ser estendido:

```
<xs:complexType name="tEndereco">
  <xs:
    <x
    <x
  </xs
</xs:c
```

Derivação por **restrição**:

```
<xs:complexType name="tEnderecoObrigatorio">
  <xs:complexContent>
    <xs:restriction base="tEndereco">
      <xs:sequence>
        <xs:element name="rua" type="xs:string"
                    minOccurs="1" maxOccurs="1"/>
        <xs:element name="numero" type="xs:integer"
                    minOccurs="1" maxOccurs="1"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```


Exemplo - Derivação por Extensão

- Tipo complexo a ser estendido:

```
<xs:complexType name="tEndereco">
  <xs:
    <x
    <x
  </xs
</xs:c
```

Derivação por **extensão**:

```
<xs:complexType name="tEnderecoCompleto">
  <xs:complexContent>
    <xs:extension base="tEndereco">
      <xs:sequence>
          <xs:element name="cep" type="tCep"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Exercício 13

- Crie um tipo complexo `tNotaFiscal` por extensão de pedido.
  - O tipo `tNotaFiscal` deve ter:

    o valor total da nota,

    a data de emissão

    valor dos impostos

Grupos de Substituição

- Mecanismo que permite que elementos sejam substituídos por outros elementos

```
<xs:element name="comment" type="TComent">
<xs:element name="comentário"
            type="TComent"
            substitutionGroup="comment"/>
```

  - Neste exemplo pode-se usar `comentário` em qualquer lugar onde é permitido utilizar `comment`
- O elemento exemplar deve ser um elemento global
- Elementos do grupo de substituição devem ser do mesmo tipo do exemplar, ou de um tipo derivado dele.

Elementos e Tipos Abstratos

- Elemento ou tipo declarado como **`abstract`**
  - Não podem ser utilizados em uma instância de um documento
- Neste caso, utiliza-se um elemento de seu grupo de substituição

```
<xs:element name="comment" type="TComent"
  abstract="true">
<xs:element name="comentário"
            type="TComent"
            substitutionGroup="comment"/>
```

- Apenas o elemento comentário poderá ser usado

Evitando a criação e uso de Tipos Derivados

- Para controlar a derivação de tipos, existe o atributo `final`:
  - `restriction`

    Proíbe a derivação por restrição.
  - `extension`

    Proíbe derivação por extensão.
  - `#all`

    Proíbe qualquer tipo de derivação.

Exercício 14 - Estudo de Caso

- Um banco deseja armazenar informações sobre contas corrente em documentos XML
- Existem 2 tipos de conta:
  - conta corrente - possui número e saldo
  - conta poupança - possui número, saldo e data de aniversário
- O correntista pode ser pessoa física ou jurídica
- Crie 4 instâncias XML exemplo
  - pessoa jurídica com conta poupança
  - pessoa física com conta poupança
  - pessoa jurídica com conta corrente
  - pessoa física com conta corrente
- Crie um esquema para as instâncias
- UTILIZE derivações!!!!

Exercício 15 e 16

- 15. O número da conta-corrente do exercício anterior deve ter o seguinte formato:
  - a000-XX
- 16. O número da conta-poupança do estudo de caso deve ter o seguinte formato:
  - 0000-aa-0

Faça as alterações necessárias no esquema

Os tipos de dados primitivos

- `xs:string`
- `xs:boolean`
- `xs:decimal`
- `xs:float`
- `xs:double`
- `xs:duration`
- `xs:dateTime`
- `xs:time`
- `xs:date`
- `xs:gYearMonth`
- `xs:gYear`
- `xs:gMonthDay`
- `xs:gDay`
- `xs:gMonth`
- `xs:hexBinary`
- `xs:base64Binary`
- `xs:anyURI`
- `xs:QName`
- `xs:NOTATION`

Any

- **<xs:any/>**
  - Estende a instância XML para uso de elementos não definidos em um esquema

```
<xs:element name="pessoa">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="nome" type="xs:string"/>
         <xs:element name="sobrenome" type="xs:string"/>
         <xs:any minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

```
<pessoa>
      <nome>Hege</nome>
      <sobrenome>Refsnes</sobrenome>
      <filho>
            <nome>Cecilie</nome>
      </filho>
</pessoa>
```

Any

- **<xs:anyAttribute/>**
  - Estende a instância XML para uso de atributos não definidos em um esquema

```
<xs:element name="pessoa">
   <xs:complexType>
        <xs:sequence>
             <xs:element name="nome" type="xs:string"/>
             <xs:element name="sobrenome" type="xs:string"/>
             <xs:anyAttribute/>
        </xs:sequence>
   </xs:complexType>
</xs:element>
```

```
<pessoa cpf="9999999999" RG="000000000">
        <nome>Hege</nome>
        <sobrenome>Refsnes</sobrenome>
</pessoa>
```