Engenharia de Ontologias (Ontology Engineering)

Universidade Federal de Uberlândia

Faculdade de Computação

Programa de Pós-Graduação em Ciência da Computação

Prof. Fabiano Azevedo Dorça

XML Schema

XML Schema

- Tanto a DTD quanto o XSD ou XML-Schema definem:

  - elementos que podem aparecer em um documento,
  - atributos que podem aparecer em um documento,
  - que elementos são elementos filhos,
  - a ordem dos elementos filhos,
  - o número de elementos filhos,
  - se um elemento é vazio ou pode incluir texto,
  - tipos de dados para elementos e atributos,
  - valores padrão e fixos para elementos e atributos.

XML Schema

- As vantagens do XSD ou XML-Schema sobre a DTD são:
  - XML-Schemas são extensíveis para adições futuras, com isto é possível:
    - reutilizar seu Schema em outros Schemas,
    - criar seus próprios tipos de dados derivados dos tipos padrões,
    - referenciar múltiplos esquemas em um mesmo documento;

XML Schema

- XML-Schemas são mais ricos e úteis que DTDs
- XML-Schemas são escritos em XML, desta forma:
    - não é preciso utilizar outra linguagem
    - pode-se usar um editor XML para editar seus arquivos XML-Schemas
    - pode usar umparser XML para verificar arquivos XML-Schemas
    - pode usar XML-Schema com XSLT

XML Schema

- XML-Schemas suportam tipos de dados, com isso é possível:

  - validar os dados
  - trabalhar com dados de um banco de dados
  - definir restrições aos dados
  - definir padrões/formatos de dados
  - converter dados entre diferentes tipos

- XML-Schemas suportam namespaces

XML Schema

- XML Schema é comumente conhecido como XML Schema Definition (XSD).
- Ele é usado para descrever e validar a estrutura e o conteúdo de dados XML.
- O esquema XML define os elementos, atributos e tipos de dados.
- É semelhante a um esquema de banco de dados que descreve os dados em um banco de dados.
- Foi a primeira linguagem de esquema para XML a obter o status de recomendação por parte do W3C.

XML Schema

- Forma de uma definição de esquema XML
- Estrutura básica

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

  <!-- declaração de tipos, elementos e atributos -->

</xs:schema>
```

XML Schema

- Prefixos de *namespace* comumente usados: xs, xsd.
- Qualquer prefixo pode ser usado, inclusive nenhum
- xmlns - referência URI que especifica um ou mais namespaces para uso neste esquema.
- Se nenhum prefixo for atribuído, os componentes de esquema do namespace podem ser usados com referências não qualificadas.

XML Schema

- Exemplo:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="values" type="xs:string"/>
</xs:schema>
```

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
    <element name="values" type="string"/>
</schema>
```

XML Schema

- Exemplo: Utilizando 2 namespaces

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:wsc="https://www.w3schools.com/w3shoolsschema">

 <element name="fname" type="wsc:mystring"/>

</schema>
```

XML Schema

- Especificação do elemento
  - Os elementos são declarados usando um elemento chamado ***xs:element*** com um atributo que dá o nome do elemento a ser definido.
  - O tipo de conteúdo do novo elemento pode ser especificado por outro atributo ou pelo conteúdo da definição de ***xs:element***.
  - As declarações de elemento podem ser de dois tipos:

XML Schema

- Tipo Simples
- Exemplos

```
<xs:element name="item" type="xs:string"/>
<xs:element name="price" type="xs:decimal"/>
```

- Os valores xs:string e xs:decimal são dois dos 44 tipos simples predefinidos na linguagem XML Schema.

XML Schema

Numéricos	xsd:float	Números reais (32bits)
	xsd:double	Números reais (64bits)
	xsd:decimal	Número decimal
	xsd:integer	Número inteiro
	xsd:nonPositiveInteger	Número inteiro negativo (incluindo 0)
	xsd:nonNegativeInteger	Número inteiro positivo (incluindo 0)
	xsd:negativeInteger	Número inteiro negativo
	xsd:positiveInteger	Número inteiro positivo
	xsd:long	Números inteiros (64bits)
	xsd:int	Números inteiros (32bits)
	xsd:short	Números inteiros (16bits)
	xsd:byte	Números inteiros (8bits)
	xsd:unsignedLong	Números long positivos (incluindo 0)
	xsd:unsignedInt	Números int positivos (incluindo 0)
	xsd:unsignedShort	Números short positivos (incluindo 0)
	xsd:unsignedByte	Números byte positivos (incluindo 0)

XML Schema

Data/Hora	xsd:dateTime	YYYY-MM-DDtHH:MM:SS.000
	xsd:date	YYYY-MM-DD
	xsd:time	HH:MM:SS.000
	xsd:gDay	Número do dia (1-31)
	xsd:gMonth	Número do mês (1-12)
	xsd:gYear	Número do ano
	xsd:gYearMonth	Números do ano e do mês
	xsd:gMonthDay	Números do mês e do dia
	xsd:duration	Período de tempo
String	xsd:string	Caracteres Unicode
	xsd:normalizedString	Caracteres sem CRLF nem Tabs
	xsd:token	Sem espaços
Binários	xsd:hexBinary	Dígitos em HEX (hexadecimal)
	xsd:base64Binary	Binários em base64
Booleanos	xsd:boolean	1 | 0 | true | false

XML Schema

- Tipo Complexo
  - Um elemento complexo é um elemento XML que contém outros elementos e/ou atributos.
  - Existem quatro tipos de elementos complexos:
    - Elementos vazios,
    - Elementos que contêm apenas outros elementos,
    - Elementos que contêm apenas texto,
    - Elementos que contêm outros elementos e texto,
    - Cada um desses elementos também pode conter atributos.

XML Schema

- Exemplo

```
<xs:element name="location">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="city" type="xs:string"/>
      <xs:element name="state" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML Schema

Exemplo

```
<?xml version="1.0">
   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="recado">
             <xs:complexType>
                  <xs:sequence>
                       <xs:element name="de" type="xs:string"/>
                       <xs:element name="para" type="xs:string"/>
                       <xs:element name="mensagem" type="xs:string"/>
                  </xs:sequence>
             </xs:complexType>
         </xs:element>
</xs:schema>
```

XML Schema

- O elemento **xs:sequence** é uma das várias maneiras de combinar elementos no conteúdo.
- DTD correspondente: <!ELEMENT location (city, state)>
- Um elemento ***xs:element*** também pode ter atributos que especificam o número de ocorrências do elemento nesta posição na seqüência.
  - minOccurs="0" // *default* = 1
  - maxOccurs="5" // *default* = maximum(1, minOccurs)
  - maxOccurs="unbounded"

XML Schema

- Exemplo: Elemento **"child_name"** pode ocorrer um **mínimo de zero vezes** e um **máximo de dez vezes** no elemento "pessoa".

```
...
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
                                    maxOccurs="10" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- Para permitir que um elemento apareça um número ilimitado de vezes, use a instrução maxOccurs = "unbounded".

XML Schema

- Ligando um XML a um XML Schema

Arquivo XML

```
<?xml version="1.0" encoding="UTF-8"?>
<persons xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="family.xsd">
<person>
  <full_name>Hege Refsnes</full_name>
  <child_name>Cecilie</child_name>
</person>
<person>
  <full_name>Tove Refsnes</full_name>
  <child_name>Hege</child_name>
  <child_name>Stale</child_name>
  <child_name>Jim</child_name>
  <child_name>Borge</child_name>
</person>
<person>
  <full_name>Stale Refsnes</full_name>
</person>
</persons>
```

schema file "family.xsd":

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

<xs:element name="persons">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="person" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="full_name" type="xs:string"/>
            <xs:element name="child_name" type="xs:string"
            minOccurs="0" maxOccurs="10"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

XML Schema

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

XML Schema

- 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>
```

XML Schema

- Usando **schemaLocation**
- *No doc. XML:*

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

- *No esquema:*

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

XML Schema

- *No esquema:*

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

- ElementFormDefault - Indica que **todo elemento** usado por uma **instância** de documento XML **que foi declarado neste esquema** deve ser **qualificado** pelo namespace.

XML Schema

- SE elementFormDefault="unqualified"
- Então a seguinte instancia XML é válida...

```
<x:author
xmlns:x="http://example.org/publishing">
   <name>Aaron Skonnard</name>
   <phone>(801)390-4552</phone>
</x:author>
```

XML Schema

- SE elementFormDefault="qualified"
- Então a instância precisa ter elementos qualificados

```
<x:author
xmlns:x="http://example.org/publishing">
   <x:name>Aaron Skonnard</name>
   <x:phone>(801)390-4552</phone>
</x:author>
```

XML Schema

Exemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:target="http://www.levijackson.net/web340/ns"
        targetNamespace="http://www.levijackson.net/web340/ns"
        elementFormDefault="qualified">
  <element name="assignments">
    <complexType>
      <sequence>
        <element name="assignments"
type="target:TypeAssignments"
                 minOccurs="1" maxOccurs="unbounded"/>
      </sequence>
    </complexType>
  </element>
```

XML Schema

Exemplo Xml Schema

DTD

```
<!ELEMENT phoneNumbers (title, entries)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT entries (entry*)>
<!ELEMENT entry (name, phone, city?)>
<!ELEMENT name (first, middle?, last)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT middle (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT city (#PCDATA)>
```

XML Schema

- Seqüenciamento de elementos é tratado com *xs:sequence*.
- Os atributos *minOccurs* e *maxOccurs* cuidam do * e ? no DTD.
- Traduzindo...

File: phone.xsd

```
<?xml version="1.0"? encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="phoneNumbers">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="entries">
          <xs:complexType><xs:sequence>
            <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType><xs:sequence>

                <xs:element name="name">
                  <xs:complexType><xs:sequence>
                    <xs:element name="first" type="xs:string"/>
                    <xs:element name="middle" type="xs:string" minOccurs="0"/>
                     <xs:element name="last" type="xs:string"/>
                  </xs:sequence></xs:complexType>

                </xs:element>
                <xs:element name="phone" type="xs:string"/>
                <xs:element name="city" type="xs:string" MinOccurs="0"/>
              </xs:sequence></xs:complexType>
            </xs:element>
          </xs:sequence></xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

XML Schema

- Combinando Elementos em Tipos Complexos
  - Já vimos que uma seqüência de elementos pode ser especificada usando **xs:sequence** dentro de um tipo complexo.
    - <xs:sequence> subelementos devem aparecer na instância XML na mesma ordem em que foram declarados no esquema
  - Os elementos podem ser combinados de outras formas em um XML Schema.

XML Schema

- **Alternativa**: xs:choice

  - O elemento de escolha XML Schema permite que apenas um dos elementos contidos na declaração *<xs:choice>* esteja presente dentro do elemento que contém.

```
<xs:element name="color">
   <xs:complexType>
      <xs:choice>
         <xs:element name="rgb" type="xs:string"/>
         <xs:element name="hsv" type="xs:string"/>
         <xs:element name="cymk" type="xs:string"/>
       </xs:choice>
    </xs:complexType>
</xs:element>
```

XML Schema

- **Ignorar ordem**: xs:all

- O elemento all especifica que os **elementos filho podem aparecer em qualquer ordem**.

```
<xs:element name="rgbColor">
<xs:complexType>
  <xs:all>
      <xs:element name="red" type="xs:unsignedbyte"/>
      <xs:element name="green" type="xs:unsignedbyte"/>
      <xs:element name="blue" type="xs:unsignedbyte"/>
   </xs:all>
</xs:complexType>
</xs:element>
```

- O conteúdo do elemento **rgbColor** deverá conter uma ocorrência de cada um dos três elementos em qualquer ordem.

- Para elementos dentro de um xs:all os atributos minOccurs e maxOccurs **não podem ser maiores do que "1"**.

XML Schema

- **Conteúdo misto**
- Define um elemento com **texto e subelementos** em seu conteúdo.
- Devido aos subelementos, o elemento que está sendo definido deve ser um **tipo complexo**.
- Para permitir **conteúdo misto** em uma definição de elemento, basta adicionar um **atributo *mixed*** à tag de início *xs:complexType*:
- mixed="true"
- Esse atributo tem um valor padrão de "false".

Exemplo

DTD Specification

```
<!ELEMENT narrative (#PCDATA | bold | italics | underline)*>
<!ELEMENT bold (#PCDATA)>
<!ELEMENT italics (#PCDATA)>
<!ELEMENT underline (#PCDATA)>
```

XML Schema Specification

```
<xs:element name="narrative">
    <xs:complexType mixed="true">
        <xs:choice minOccurs="0" maxOccurrs="unbounded">
            <xs:element name="bold" type="xs:string"/>
            <xs:element name="italics" type="xs:string/>
            <xs:element name="underline" type="xs:string/>
        </xs:choice>
    </xs:complexType>
</xs:element>
```

XML Schema

- O seguinte elemento XML pode ser validado em relação ao fragmento XSD anterior.

```
<narrative>
   Higher beings from <italics>outer space</italics> may not want to
   tell us the <underline>secrets of life </underline>because we're
   not ready. But maybe they'll change their tune after a little
   <bold>torture</bold>.
   <italics>Jack Handey</italics>
</narrative>
```

XML Schema

- Exemplo:
- O seguinte elemento XSD...

```
<xs:element name="letter">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="orderid" type="xs:positiveInteger"/>
      <xs:element name="shipdate" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML Schema

- Valida o XML...

```
<letter>
  Dear Mr.<name>John Smith</name>.
  Your order <orderid>1032</orderid>
  will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

- Neste caso, o *sequence* obriga que os elementos sejam usados em sequencia.

XML Schema

- Especificações de **atributo**
  - Elementos simples não podem ter atributos.
  - Se um elemento tem atributos, ele é considerado do tipo complexo.
  - Atributos são declarados como tipos simples.
  - Isso significa que um elemento com atributos sempre tem uma definição do tipo complexo.

XML Schema

- Os atributos são definidos usando o elemento ***xs:attribute*** com seus próprios atributos, *name* e *type*.

- Os valores de atributo não podem conter elementos ou outros atributos.

XML Schema

- Exemplo
- Elemento XML com um atributo:
  - <lastname lang="EN">Smith</lastname>
- Definição de atributo correspondente:
  - <xs:attribute name="lang" type="xs:string"/>

XML Schema

- Exemplo:

```
<xs:element name="name">
  <xs:complexType>
    <xs:sequence>
        <xs:element name="first" type="xs:string"/>
        <xs:element name="middle" type="xs:string" minOccurs="0"/>
        <xs:element name="last" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="gender" type="xs:string"/>
  </xs:complexType>
</xs:element>
….
<name gender=”masculino”>
<first>Joao</first><last>Silva</last>
</name>
```

XML Schema

- Declarando **elementos complexos** que contenham apenas texto:

  Exemplo:
- **XML instance**
  - <produto prodid="1234">Feijão</produto>

  XML Schema

```
<xs:element name="produto">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attribute name="prodid" type="xs:positiveInteger"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

  ou...

XML Schema

- As especificações de **atributo** devem estar dentro de um **tipo complexo**, mas devem ficar **após** a definição da estrutura (xs:sequence, xs:all, or xs:choice) no tipo complexo.

XML Schema

- Esta tabela resume a maioria dos possíveis atributos permitidos no elemento **xs:attribute**.

Attribute	Possible Attribute Values
name	Name of the attribute
type	Type of the value of the attribute
use	"required" "optional" (the default) "prohibited" (rarely used)
default	Default value for the attribute, which must conform to the type of the attribute value (*use* must be "optional").
fixed	Fixed value for the attribute, which must conform to the type of the attribute value.
ref	Name of an attribute defined globally; used in place of *name*.

XML Schema

- Valores Padrão e Fixo para Atributos
- Um valor padrão é atribuído automaticamente ao atributo quando nenhum outro valor é especificado.
- No exemplo a seguir, o valor padrão é "EN":
  - <xs:attribute name="lang" type="xs:string" default="EN"/>
- Um valor fixo também é atribuído automaticamente ao atributo e não é possível especificar outro valor.
- No exemplo a seguir, o valor fixo é "EN":
  - <xs:attribute name="lang" type="xs:string" fixed="EN"/>

XML Schema

- Atributos opcionais e obrigatórios
  - Os atributos são opcionais por padrão.
  - Para especificar que o atributo é obrigatório:
- <xs:attribute name="lang" type="xs:string" use="required"/>

XML Schema

- Fim