Gerenciamento de Dados e Informação

Dados Semiestruturados & XML

Fernando Fonseca
Ana Carolina
Robson Fidalgo

Dados na Web

- O objetivo é **integrar** todos os tipos de informação, incluindo informação não estruturada
  - Informação **irregular** ou **ausente**
  - Informação com estrutura **não conhecida completamente**
  - Esquemas que **evoluem** dinamicamente

Representação de Dados para Web/BD

- Necessidades de modelar
  - A própria **Web**
  - A estrutura de **Web sites**
  - A estrutura **interna de páginas** da Web
  - O **conteúdo** do Web site em menor granularidade

Web X Banco de Dados

- Web: **enorme** banco de dados
- A maioria dos documentos é gerada para ser disponibilizada para **leitura**
- Alguns destes documentos foram gerados a partir de **consultas a BD**
- Dados podem ser extraídos das páginas Web para serem **utilizados** por outros programas

Web X Banco de Dados

Web

- Padrão simples e universal para troca de informações
- Informações decompostas como unidades que possam ter nome (URL) e ser transmitidas (HTTP)
- Estrutura da informação (HTML)

Banco de Dados

- Esquemas (relacional) e diagramas (E-R) para descrever a estrutura
- Linguagem de consulta, controle de concorrência, recuperação e integridade
- Separa a visão lógica da implementação física

Estrutura dos Dados

- Dados Estruturados (BD)
- Dados Semi-estruturados (XML)
- Dados Não Estruturados (HTML)

Dados Estruturados

- Os SGBD trabalham com dados bem estruturados
  - Esquema **pré-definido**
  - Todos os dados **de acordo** com o esquema
- SGBD **precisam** do esquema para
  - Armazenar e indexar **dados**
  - Processar **consultas** e **atualizações**
- Usuários precisam do esquema para **formular** consultas e atualizações

Dados Semi-estruturados

- Atualmente, muitas informações são semi-estruturadas
  - **Ausência** de uma estrutura regular, ou a estrutura é capaz de **evoluir** de forma imprevisível
  - Dados podem ser **incompletos**
  - SGBD e usuários não precisam **conhecer completamente** a estrutura

Dados Semi-estruturados

- Fontes de dados semi-estruturados
  - **Integração** de dados e ambientes de **troca** de informação
  - Dados **extraídos** da Web
  - **XML** (eXtensible Markup Language)
- Características dos Dados Semi-estruturados
  - Estrutura **irregular** (dados heterogêneos)
    - Modelar e consultar esta estrutura irregular é essencial

Dados Semi-estruturados

- Características dos Dados Semi-estruturados (Cont.)
  - A estrutura pode ser **implícita**
    - Alguma **computação** é necessária para obtê-la
    - A correspondência entre a estrutura e a representação lógica dos dados **nem sempre é imediata**

Dados Semi-estruturados

- A estrutura pode ser parcial
  - **Parte** dos dados pode não ter estrutura (ex: **bitmaps**)
  - Outros podem ter uma estrutura "**fraca**" (ex: **textos**)
- Tipos são apenas **indicativos**
  - Ao contrário do sistema de **tipos rígido** das aplicações de BD

XML

XML - Extensible Markup Language

- Desenvolvida em 1996 pelo **XML Working Group** formado sob a proteção do World Wide Web Consortium (**W3C**)
- Linguagem de marcadores
  - Para **descrever** informações
  - **Estrutural** e **semântica**, não uma linguagem de formatação
- Padrão XML
  - Para **representação** de dados
  - Para **troca** de informações

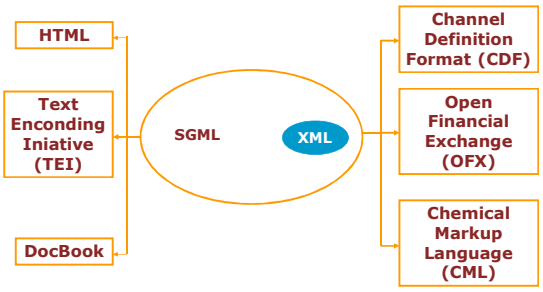
O que significa "markup language"?

- Markup
  - Informação extra que consiste de instruções para controlar o layout e a aparência das palavras
  - É qualquer forma de tornar explícita a interpretação de um texto
  - Uma linguagem de marcadores é uma coleção de convenções de marcadores utilizados em conjunto para a codificação de textos

Qual a origem de XML?

- XML é um subconjunto de SGML - ISO8879
- **SGML** (Standard Generalized Markup Language) Uma metalinguagem por meio da qual se pode definir linguagens de marcação para documentos
  - Um **padrão internacional** para a definição de métodos de representação de texto em formato eletrônico
  - Padrão muito poderoso e bastante geral, o que torna **complicada** a sua implementação

Relação entre SGML, XML e HTML

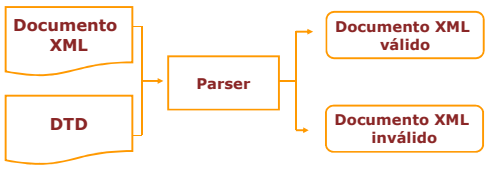


XML, outra linguagem de marcadores?

- Não!
  - A maioria das linguagens provê um conjunto fixo de marcadores, XML é **extensível**
    - XML permite a definição de novos marcadores
  - Descrição de documentos XML
    - **DTD** - Document Type Definition
    - XML **Schema**

DTD - Document Type Definition

- O conjunto dos tipos de elementos são usados para definir os tipos de documentos e são referenciados como Document Type Definition - **DTD**



XML Schema

- Proposta da W3C para descrever a estrutura de um documento XML
- XML Schema é um padrão mais abrangente que uma DTD
  - Dá suporte a um **conjunto maior** de tipos de dados primitivos
  - Permite **definir novos** tipos de dados
  - Dá suporte à **herança**

Qual a idéia central de XML?

- Tornar explícita a separação entre os principais componentes de um documento eletrônico

Conteúdo

Apresentação **Estrutura**

Exemplo de Documento XML

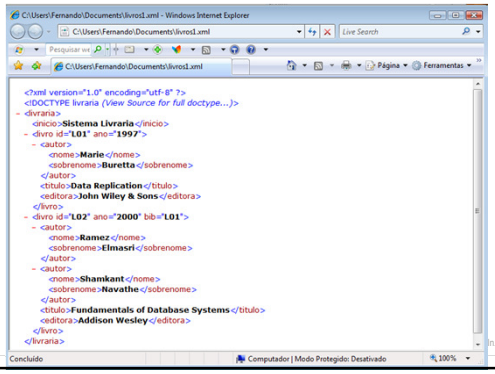
```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE livraria SYSTEM "livraria.dtd">
<livraria>
<livro id="L01" ano="1997">
        <autor>
          <nome>Marie</nome >
          <sobrenome>Buretta</sobrenome >
        </autor>
        <titulo>Data Replication</titulo>
        <editora>John Wiley &amp;amp; Sons </editora>
</livro>
...
```

Arquivo com extensão **.xml**

Exemplo de Documento XML

```
...
<livro id="L02" ano= " 2000" bib="L01">
    <autor>
      <nome>Ramez </nome>
      <sobrenome>Elmasri</sobrenome >
    </autor>
    <autor>
      <nome> Shamkant </nome>
      <sobrenome> Navathe </sobrenome >
    </autor>
   <titulo>Fundamentals of Database Systems</titulo>
   <editora>Addison Wesley</editora>
</livro>
</livraria>
```


Visão do Arquivo no Internet Explorer



Especificações de XML

- Extensible Markup Language (XML) 1.0
  - Define a sintaxe de XML
- XML Link Language (XLL)
  - Define como conectar documentos XML utilizando links de hipertexto
- Extensible Style Language (XSL)
  - Define como formatar documentos XML utilizando stylesheet

Objetivos de XML

- Ser possível usar XML **diretamente** por toda a Internet
- Dar suporte a uma **grande variedade** de aplicações
- Ser compatível com **SGML**
- Ser fácil **escrever programas** para processar documentos XML

Objetivos de XML

- O número de características adicionais a XML deverá ser o **mínimo** possível
- Documentos XML devem ser **legíveis** e razoavelmente claros
- O projeto de XML deve ser preparado **rapidamente**
- O projeto de XML deve ser **formal** e **conciso**
- Documentos XML devem ser **fáceis** de criar

Benefícios

- XML é um padrão completamente **aberto**
- Documentos XML podem ser usados e reusados de **diferentes** formas e em diferentes formatos
- Os autores de documentos XML podem concentrar-se no **conteúdo** e não na formatação
- Documentos XML são **auto-descritíveis**
- Documentos XML são como **BD** de informações
- O conteúdo dos documentos pode ser **manipulado** e **reorganizado** pelo browser

Classes de documentos XML

- Documento **bem formado**
  - Documento que está de acordo com o padrão XML
- Documento **válido**
  - Documento XML bem formado que está de acordo com a DTD (ou esquema) associada(o)

Produtos para XML

- Ferramentas para criação e modificação de documentos XML
  - Editores de XML
- Ferramentas para criação e modificação de DTD, XSL style sheets, etc.
  - DTD (editores, geradores)
  - Ferramentas para fazer conversão entre DTD e Esquemas
  - XSL (editores, geradores)

Produtos para XML

- Ferramentas para dar suporte ao gerenciamento e ao armazenamento de documentos XML
  - Sistemas que armazenam persistentemente documentos XML e oferecem acesso à estrutura dos documentos e a seus componentes
  - Utilitários para gerenciamento de documentos
  - Mecanismos de busca para XML
  - SGBD
  - Parsers
  - Browsers

Parsers para XML

- Toda aplicação (browsers, editores, ...) para XML possui um parser
  - Parsers que **fazem a validação** de acordo com a DTD
  - Parsers que **ignoram as restrições** de validade impostas pela DTD

Parsers para XML

- O parser divide o documento em "porções"
  - Geralmente correspondem a elementos e atributos
- A aplicação pode manipular as "porções" diretamente, como se fosse um BD
  - **Transformar** para outros formatos
  - **Reorganizar** a sequência dos elementos
  - **Aplicar** alguma formatação para apresentação

API para XML

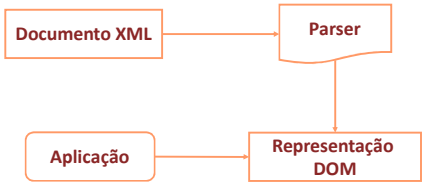
- DOM e SAX são API para XML
- Oferecem meios para acessar e manipular o conteúdo de um documento XML
- Oferecem diferentes visões do documento
  - **DOM** (Document Object Model): visão baseada em **árvore**
  - **SAX** (The Simple API for XML): visão baseada em **eventos**

Como manipular o conteúdo de um documento XML?

- As aplicações podem utilizar as operações disponíveis na **API** para acessar o conteúdo do documento XML
  - Um parser baseado em **DOM** produz como saída uma árvore que representa a **hierarquia dos elementos** em um documento XML
  - Um parser baseado em **SAX** produz como saída uma **sequência de eventos**

A API DOM

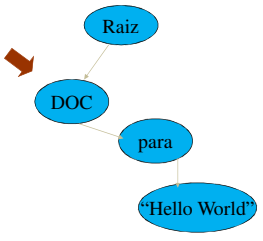
- Proposta pelo W3C
- API **independente** de linguagem e plataforma que permite programas e *scripts* acessarem e atualizarem o conteúdo e a estrutura de um documento dinamicamente



A API DOM

- Exemplo

```
<?xml version="1.0">
 <doc>
 <para>Hello, world! </para>
</doc>
```



A API SAX

- Interface que permite a interação com documentos XML
- Proposta por um grupo de participantes da lista **XML-DEV**
- Exemplo

```
<?xml version="1.0">
 <doc>
 <para>Hello, world!
</para> </doc>
```

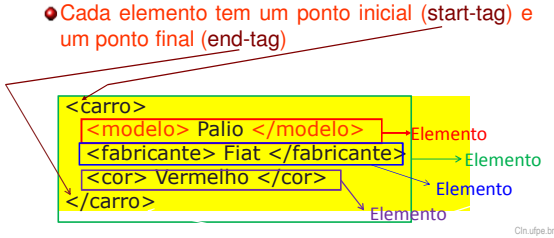
```
start document
start element: doc
start element: para
characters: Hello, world!
end element: para
end element: doc
end document
```


A API SAX

- **Não** permite acessos randômicos na manipulação do documento
- É preciso implementar um **modelo próprio** para a manipulação dos dados
- É mais adequada quando o **processamento** do documento é sequencial

Construindo Documentos XML

- Elementos
  - Os elementos são os blocos principais da estrutura hierárquica de XML
  - Cada elemento tem um ponto inicial (start-tag) e um ponto final (end-tag)



Elementos Aninhados

- Um elemento pode conter outros elementos

```
<livro id="L01" ano="1997">
  <autor>
    <nome>Marie</nome >
    <sobrenome>Buretta</sobrenome >
  </autor>
  <titulo>Data Replication</titulo>
  <editora>John Wiley &amp;amp; Sons </editora>
</livro>
```


Elemento Vazio

- Um elemento também pode ter um conteúdo vazio

```
<vazio/>
```

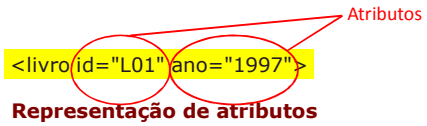
ou

```
<vazio></vazio>
```

Representações de um elemento vazio

Atributos

- Podem ser associados com um elemento em um start-tag ou um elemento vazio
- Valores de atributos podem ser delimitados por " ou '



Representação de atributos

Atributo ou Elemento?

- A informação possui alguma **estrutura**?
  - Atributos **não têm** hierarquia
  - Elementos **podem** ter hierarquia

```
<pessoa>
  <nome>...
  <endereco>...
  <telefone>...
</pessoa>
```


Atributo ou Elemento?

- A informação deve seguir alguma **ordem** pré-definida?
  - Múltiplos valores de atributos em um único start tag não têm uma ordem pré-definida
  - Os **subelementos** de um elemento devem ser definidos na ordem estabelecida na declaração do elemento
- Outras diferenças
  - Um atributo pode aparecer uma **única** vez dentro de um start tag
  - Sublementos com mesmo tag podem ser **repetidos** na definição do elemento

O elemento raiz (root)

- É o elemento que contém **todos** os outros elementos do documento
- Pode existir apenas **um** elemento raiz

Raiz

```
<?xml version="1.0"?>
<livraria>
....outros elementos
</livraria>
```


Escrevendo Comentários

- A string "--" não é permitida dentro de um comentário
- Não pode ser colocada dentro de outro marcador

```
<!-- Exemplo de comentário -->
```

Início

Término

Escrevendo símbolos especiais

- Seções **CDATA** são usadas quando um documento XML contém um grande número de caracteres especiais (ex: "<" e "&")
- São blocos de texto nos quais estes caracteres não são considerados especiais

```
<Documento>
<![CDATA [
se a<b e b<c então a<c ]]>
</Documento>
```


Instruções de processamento

- São utilizadas para enviar comandos e informações à aplicação que está processando o documento XML
- Um exemplo de instrução de processamento é a declaração de XML

```
<?xml version="1.0" encoding="utf-8"?>
```

Instrução

Regras para criar um documento XML bem-formado

- Todos os *tags* de **início** devem ter um *tag* de final correspondente
- *Tags* de elementos **vazios** terminam com />
- Existe um **único** elemento raiz
- Elementos **não podem** se sobrepor
- **Valores** de atributos devem ser colocados entre " ou '
- < e & são usados apenas em *start tags* e entidades. Caso necessários no texto, representar como em **HTML**
  - < → <
  - & → &

DTD

Uso de DTD

- Uma DTD **descreve** os elementos e atributos que podem aparecer em um documento
- A **validação** compara um documento em particular com a DTD correspondente
- É **necessário** que um documento seja bem-formado para ser validado
- Garante que os dados estão **corretos** antes de serem utilizados por outras aplicações
- Garante que o **formato** foi seguido

Uso de DTD

- Armazena
  - Declarações de tipos de elementos
    - <!ELEMENT...>
  - Declaração de lista de atributos
    - <!ATTLIST ...>
  - Declarações de entidade
    - <!ENTITY ...>
- Pode ter
  - Um componente interno (subconjunto interno) e/ou
  - Um componente externo (subconjunto externo)

Declarando uma DTD interna

Documento - liv.xml

```
<!-- DTD Interna -->
<!DOCTYPE livraria [
<!ELEMENT livro (titulo, autor)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT autor (#PCDATA)>]>
```

Nome do elemento raiz do documento

Declaração de um tipo de elemento

Declarando uma DTD interna

- Exemplo

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE livraria [
<!ELEMENT livro (titulo, autor)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT autor (nome, sobrenome)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT sobrenome (#PCDATA)>]>
<livraria>
<livro >
        <titulo>Data Replication</titulo>
      <autor>
          <nome>Marie</nome >
          <sobrenome>Buretta</sobrenome >
        </autor>
</livro>
...
</livraria>
```

DTD Interna

Declarando uma DTD externa

Documento - liv.xml

```
...
<!-- DTD Externa -->
<!DOCTYPE livraria SYSTEM "livraria.dtd">
<livraria>
<livro>
        <titulo>Data Replication</titulo>
        <autor>
          <nome>Marie</nome >
          <sobrenome>Buretta</sobrenome >
        </autor>
</livro>
...
</livraria>
```

DTD Externa

Referência a um arquivo externo

Declarando uma DTD externa

DTD - livraria.dtd

```
<!ELEMENT livraria(livro+)
<!ELEMENT livro (titulo, autor)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT autor (nome, sobrenome)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT sobrenome (#PCDATA)>
```


XML Schema

Uso de XML Schema

- Uso de **Namespaces**
  - Maneira simples e direta de **distinguir** nomes usados em documentos XML, sem levar em consideração sua origem
  - Tem como propósito oferecer aos programadores uma ajuda, permitindo que tags e **atributos** sejam processados somente quando forem **relevantes**

Namespaces - Exemplo

```
<h:html xmlns:xdc="http://www.xml.com/livros"
        xmlns:h="http://www.w3.org/HTML/1998/html4">
 <h:head><h:title>Livro</h:title></h:head>
 <h:body>
  <xdc:livro>
   <xdc:title>Fundamentals of database systems</xdc:title>
   <h:table>
    <h:tr align="center">
     <h:td>Autor</h:td><h:td>Preco</h:td>
```


O papel dos prefixos

- Os prefixos são apenas **atalhos** para os nomes completos
- Os prefixos são definidos como **atributos** do elemento raiz
- Exemplo

```
<h:html xmlns:xdc="http://www.xml.com/livros"
        xmlns:h="http://www.w3.org/HTML/1998/html4">
```

Namespace default

- É possível declarar um namespace *default* e ocultar alguns prefixos

```
<html xmlns="http://www.w3.org/HTML/1998/html4"
      xmlns:xdc="http://www.xml.com/books">
<head><title>Book Review</title></head>
<body>
 <xdc:bookreview>
  <xdc:title>XML: A Primer</xdc:title>
  <table>
   <tr align="center"> ....
```


Atributos podem ter Namespaces

- Tanto **atributos** quanto **elementos** podem ter namespaces

```
<h:body>
 <xdc:bookreview>
 <xdc:title h:style="font-family: sans-serif;">
   Fundamentals of database system </xdc:title>...
```


Namespaces – Nomes universais

- A **combinação** de um nome local com uma URL é chamada de "nome universal"
- O papel da **URL** em um nome universal é puramente permitir que as aplicações tenham como **identificar unicamente** os elementos ou atributos

Esquemas XML - Objetivos

- O propósito de uma linguagem de definição de esquemas é oferecer um **conjunto de construtores** para definições de esquemas XML
- Esquemas XML podem ser usados para **definir**, **descrever** e **catalogar** vocabulários para classes de documentos

Linguagens de Esquemas

- DCD [Document Content Description]
- XML-Data Reduced (XDR)
- DDML (Xschema)
- Schema for Object-Oriented XML (SOX)
- W3C XML Schema Definition Language (XSDL)

XML Schema – Sintaxe Básica

- Uma especificação em XML *Schema* sempre inicia com a tag *<schema>* e termina com a tag *</schema>*
- Todas as declarações de elementos, atributos e tipos devem ser inseridas entre estas duas tags

XML Schema – Sintaxe Básica

- Tipos podem ser
  - Simples (*simpleType*):são tipos básicos como string, date, float, double...
  - Complexos (*complexType*): definem a estrutura de elementos, ou seja definem características como:
    - Subelementos
    - Atributos
    - Cardinalidades dos subelementos
    - Obrigatoriedade dos atributos

XML Schema – Exemplo 1

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
<xsd:element name="livro">
 <xsd:complexType>
  <xsd:sequence>
    <xsd:element name="titulo" type="xsd:string"/>
    <xsd:element name="autor" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="isbn" type="xsd:string"/>
 </xsd:complexType>
</xsd:element>
</xsd:schema>
```


XML Schema – Exemplo 2

- Considerando a relação

Filme (mat: *number*, titulo: *varchar2(30)*, duracao: *varchar2(4)*, genero: *varchar2(7)*)

- Criar um XML Schema para um arquivo XML contendo a resposta para: SELECT * FROM Filme;

XML Schema – Exemplo 2

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
<xsd:element name="filme">
 <xsd:complexType>
  <xsd:sequence>
    <xsd:element name="mat" type="xsd:integer"/>
    <xsd:element name="titulo" type="xsd:string"/>
    <xsd:element name="duracao" type="xsd:string"/>
    <xsd:element name="genero" type="xsd:string"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
</xsd:schema>
```


XML Schema – Sintaxe Básica

- Os **tipos de dados** definidos em um esquema podem ser usados para a definição de elementos e atributos
  - Os tipos complexos (`complexType`) podem ser usados apenas para definição de **elementos**
  - Os tipos simples (`simpleType`) podem ser usados para definição tanto de **elementos** como de **atributos**

XML Schema – Sintaxe Básica

- XML Schema permite a definição de **cardinalidade** para um elemento
  - O atributo `minOccurs` determina o número **mínimo** de ocorrências de um elemento
  - O atributo `maxOccurs` determina o número **máximo** de ocorrências de um elemento

Declaração de Elementos

- Basicamente, existem **três formas** diferentes de declarar elementos
  - A declaração de um elemento tem como **subelemento** a definição de um **tipo complexo**
  - A declaração de um elemento tem como **subelemento** a definição de um **tipo simples**
  - A declaração de um elemento faz **referência** a um **tipo complexo já definido**

Declaração de Elementos (1)

- A declaração de um elemento tem como subelemento a definição de um tipo complexo

```
<xsd:element name="livro">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="titulo" type="xsd:string"/>
      <xsd:element name="editora" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```


Declaração de Elementos (2)

- A declaração de um elemento tem como subelemento a definição de um tipo simples

```
<xsd:element name="meuInteiro">
  <simpleType>
      <restriction base="integer">
        <minInclusive value = "1">
        <maxInclusive value = "10">
      </restriction>
  </simpleType>
</xsd:element>
```


Declaração de Elementos (3)

- A declaração de um elemento faz **referência** a um tipo complexo **já definido**

```
<xsd:complexType name="Tlivro">
    <xsd:sequence>
      <xsd:element name="titulo" type="xsd:string"/>
      <xsd:element name="editora" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:element name="livro" type="Tlivro"/>
```


Declaração de Elementos

- Declarações de elementos e tipos são ditas **globais** quando são filhas imediatas do elemento `<schema>`
- Declarações de elemento e definições de tipos são consideradas **locais** quando estão aninhadas dentro de outros elementos ou tipos
- Esta diferença é importante porque apenas elementos e tipos globais podem ser **reusados**

Derivação de Tipos

- XML Schema possui um mecanismo de **derivação de tipos**, permitindo a criação de novos tipos a partir de outros já existentes
- A derivação pode ser feita de duas maneiras
  - Por **restrição**
  - Por **extensão**

Derivação de Tipos

- Tipos simples só podem ser derivados por **restrição**, aplicando-se "facetas" a um tipo básico ou utilizando uma linguagem de expressões regulares

```
<simpleType name="meuInteiro">
  <restriction base="integer">
    <minInclusive value = "1">
    <maxInclusive value = "10">
  </restriction>
</simpleType>
```


Derivação de Tipos

- Tipos complexos podem ser **derivados** por restrição ou por extensão
  - Por **restrição**: permite restringir a cardinalidade de um subelemento
  - Por **extensão**: adiciona características a um tipo (semelhante à **herança**)

Derivação de Tipos

```
<xs:element name="employee" type="fullpersoninfo"/>
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
    <xs:sequence>
      <xs:element name="address" type="xs:string"/>
      <xs:element name="city" type="xs:string"/>
      <xs:element name="country" type="xs:string"/>
    </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```


XML Schema - Grupos

- Grupos especificam restrições sobre um conjunto fixo de subelementos, as quais podem ser de três tipos
  - *sequence*: todos os elementos pertencentes a ele devem aparecer na ordem em que foram definidos e nenhum pode ser omitido
  - *choice*: apenas um dos elementos pertencentes ao grupo deve aparecer em uma instância XML
  - *all*: os elementos podem aparecer em qualquer ordem e podem ser repetidos ou omitidos

XML Schema – Sintaxe Básica

- Os atributos de um *ComplexType* são declarados utilizando-se a tag *<attribute>* e devem ser do tipo ***simpleType***
- Um atributo pode ser declarado como opcional por meio da cláusula ***use***. Os valores permitidos para esta cláusula são
  - required (obrigatório)
  - optional (opcional)
  - fixed (fixo)
    - Neste caso deve-se dizer o **valor *default*** do atributo utilizando a cláusula ***value***

XML Schema - Exemplo

```
....
<xsd:complexType>
 <xsd:sequence>
  <xsd:element name="titulo" type="xsd:string"/>
  <xsd:element name="autor" type="xsd:string"/>
 </xsd:sequence>
 <xsd:attribute name="isbn" type="xsd:string"/>
</xsd:complexType>
.....
```

Formatação de Documentos

Formatando Documentos XML

- **Style sheets** descrevem a forma de apresentação de documentos na tela do computador / impressora
  - Extensible Style Language (XSL)
  - Cascading Style Sheets (CSS)
- Utilizando **CSS**
  - Criar arquivo .css para armazenar a formatação de acordo com a sintaxe **CSS**
    - http://www.w3.org/Style/CSS/
  - Introduzir informação sobre o CSS no arquivo XML

```
<?xml-stylesheet type="text/css" href="arquivo.css"?>
```


Exemplo de CSS

- Baseado no exemplo Livraria – arquivo livros1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE livraria SYSTEM "livraria.dtd">
<?xml-stylesheet type="text/css"  href="estilolivro.css"?>
<livraria>
<inicio>Sistema Livraria</inicio>
<livro id="L01" ano="1997">
        <autor>
          <nome>Marie</nome >
          <sobrenome>Buretta</sobrenome >
        </autor>
        <titulo>Data Replication</titulo>
        <editora>John Wiley &amp;amp; Sons </editora>
</livro>
```

Informação sobre arquivo CSS

Cabeçalho

Exemplo de CSS

```
<livro id="L02" ano= " 2000" bib="L01">
    <autor>
      <nome>Ramez </nome>
      <sobrenome>Elmasri</sobrenome >
    </autor>
    <autor>
      <nome> Shamkant </nome>
      <sobrenome> Navathe </sobrenome >
    </autor>
   <titulo>Fundamentals of Database Systems</titulo>
   <editora>Addison Wesley</editora>
</livro>
</livraria>
```


Exemplo de CSS

- Arquivo estilolivro.css

```
@media screen {
livraria {
    display: block;
    margin: 10px;
    width: 400px;
  }
```

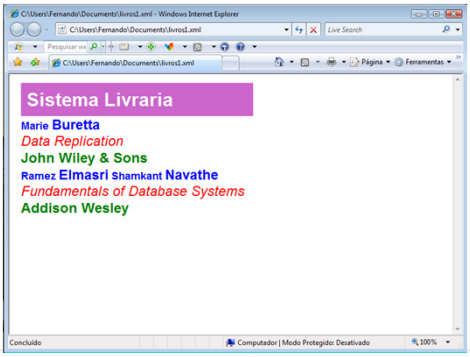

Exemplo de CSS

```
inicio {
    display: block;
    padding: 0.3em;
    font: bold x-large sans-serif;
    color: white;
    background-color: #C6C;
  }

livro {
    display: block;
    font: normal medium sans-serif;
  }
```

Exemplo de CSS

```
titulo {display: block;
    font-style: italic;
    font-size: large;
    color: red;
  }

   editora { display: block;
           font-weight: bold;
            font-size: large;
           color: green;
    }
}
```


Visão do Arquivo no Internet Explorer



Bancos de Dados e XML

- Sistema Gerenciador de Banco de Dados XML Nativo (Native XML Database - NXD)
  - Ex: Tamino, dbXML e X-Hive
- Sistema Gerenciador de Banco de Dados compatível com XML (XML Enabled Database - XEDB)
  - Ex: As soluções para XML propostas pela Oracle e Microsoft
- Sistema Gerenciador de Banco de Dados XML Híbrido (Hybrid XML Database - HXD)
  - Ex: Excelon e Ozone