Gerenciamento de Dados e Informação

Fernando Fonseca
Ana Carolina
Robson Fidalgo

Centro de Informática
Universidade Federal de Pernambuco
CIn.ufpe.br

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

- Por que desejamos tratar a Web como um BD?
  - Para **manter** integridade
  - Para fazer consultas de acordo **com alguma estrutura** (oposto a consultas baseadas em conteúdo)
  - Para introduzir **alguma organização**
- A Web não tem estrutura. O melhor que podemos afirmar é que a Web é um **enorme grafo**

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

13

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

14

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

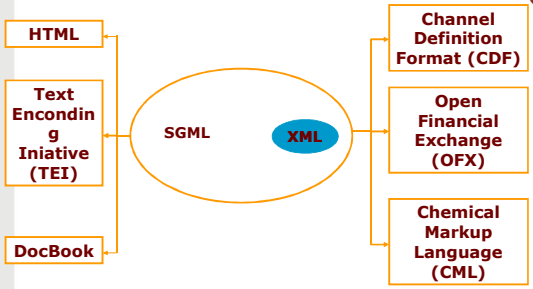
15

Qual a origem de XML?

- XML é um subconjunto de SGML - ISO8879
- **SGML** (Standard Generalized Markup Language) Uma metalinguagem através 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

16

Relação entre SGML, XML e HTML



17

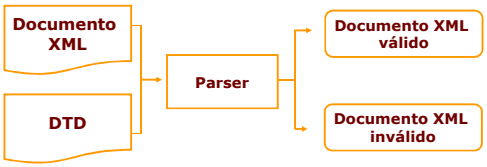
XML, outra linguagem de marcadores?

- 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**

18

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**



19

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**

20

Qual a idéia central de XML?

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

Conteúdo

Apresentação **Estrutura**

21

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**

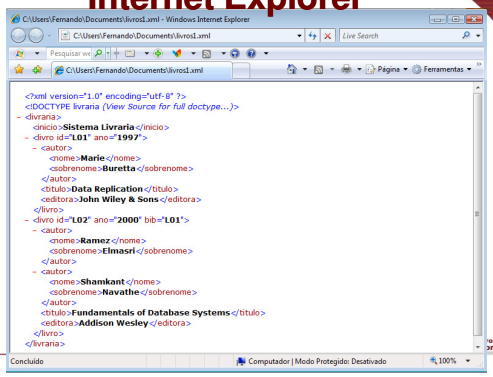
22

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>
```

23

Visão do Arquivo no Internet Explorer



24

Especificações de XML

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

25

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

26

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

27

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

28

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)

29

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)

30

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 seqüência dos elementos
  - **Aplicar** alguma formatação para apresentação

API para XML

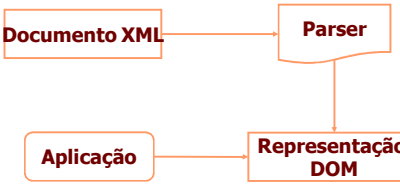
- DOM e SAX são API para XML
- Oferecem meios para acessar e manipular o conteúdo de um documento XML
- Oferecem diferentes visões do documento
  - **DOM** (Document Object Model): visão baseada em **árvore**
  - **SAX** (The Simple API for XML): visão baseada em **eventos**

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 **seqüência de eventos**

A API DOM

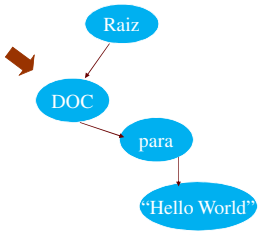
- Proposta pelo W3C
- API **independente** de linguagem e plataforma que permite programas e *scripts* acessarem e atualizarem o conteúdo e a estrutura de um documento dinamicamente



A API DOM

- Exemplo

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



37

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
```

38

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 seqüencial

39

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)

```
<carro>
  <modelo> Palio </modelo>   → Elemento
  <fabricante> Fiat </fabricante>
  <cor> Vermelho </cor>
</carro>
```

40

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>
```

41

Elemento Vazio

- Um elemento também pode ter um conteúdo vazio

```
<vazio/>
```

ou

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

Representações de um elemento vazio

42

Atributos

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

Atributos

```
<livro id="L01" ano="1997">
```

Representação de atributos

Atributo ou Elemento?

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

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

Definição do elemento 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, onde 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 id="L01" ano="1997">
        <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">
```

Referência a um arquivo externo

DTD - livraria.dtd

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

56

Declarando uma DTD externa

- Exemplo

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

DTD Externa

57

Declaração de Elementos

- Permite a um documento XML
  - **Restringir** os elementos que ocorrem no documento
  - **Especificar a ordem** em que estes elementos ocorrem

```
<!ELEMENT .....>
```

```
<!ELEMENT livro EMPTY>
```

```
<!ELEMENT titulo (#PCDATA)>
```

58

Declaração de Elementos

- (1) Seqüência de Elementos
- (2) Seleção a partir de uma lista de elementos
- (3) A ocorrência de um elemento é opcional
- (4) Um elemento ocorre zero ou mais vezes
- (5) Um elemento ocorre uma ou mais vezes
- (6) Um elemento contém qualquer outro elemento em qualquer ordem

59

Declaração de Elementos(1)

- Seqüência de Elementos
  - Especifica que um elemento consiste de outros elementos, **exatamente** na ordem em que for especificada a seqüência

```
<!ELEMENT pessoa (nome, endereco, telefone)>
```

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

Exemplo válido

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

Exemplo inválido

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

Exemplo inválido

60

Declaração de Elementos (2)

- Seleção a partir de uma lista de elementos
  - Especifica que um elemento consiste de **apenas um** dos elementos especificados na declaração

```
<!ELEMENT publicacao (livro|artigo)>
```

```
<publicacao>
  <livro>...
  <artigo>...
</publicacao>
```

Seleção

Exemplo inválido

Declaração de Elementos (3)

- A ocorrência de um elemento é opcional
  - Especifica que a participação de um elemento em um outro elemento é **opcional** e, caso exista, deve ser **única**

```
<!ELEMENT livro (titulo, autor, editora?)>
```

```
<livro>
  <titulo>...
  <autor>...
</livro>
```

Exemplo válido

```
<livro>
  <titulo>...
  <autor>...
  <editora>...
  <editora>...
</livro>
```

Exemplo inválido

Declaração de Elementos (4)

- Um elemento ocorre zero ou mais vezes
  - Especifica que um elemento consiste zero ou mais elementos marcados com *

```
<!ELEMENT pessoa (nome, endereco, telefone*)>
```

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

Exemplo válido

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

Exemplo válido

Declaração de Elementos (5)

- Um elemento ocorre uma ou mais vezes
  - Especifica que um elemento consiste um ou mais elementos marcados com +

```
<!ELEMENT pessoa (nome, endereco, telefone+)>
```

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

Exemplo inválido

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

Exemplo válido

Declaração de Elementos (6)

- Um elemento contém qualquer outro elemento em qualquer ordem
  - Especifica que um elemento consiste de **qualquer combinação** de elementos em qualquer ordem
  - O elemento também pode **conter caracteres**
  - O elemento pode conter outros elementos e caracteres em **qualquer ordem**

```
<!ELEMENT pessoa ANY>
```


Declaração de Atributos

- Uma declaração de atributos especifica o **nome**, o **tipo** e opcionalmente o **valor *default*** dos atributos associados a um elemento
  - Os nomes **não** podem ser repetidos no mesmo elemento

```
<!ATTLIST produto
          nome CDATA #REQUIRED
          preco CDATA #IMPLIED
          id ID #REQUIRED >
```

Formato padrão para definição de lista de atributos

Atributo String

- O valor de um atributo do tipo **String** é uma cadeia de caracteres de **qualquer** tamanho

```
<!ATTLIST produto nome CDATA>
```

Definição do atributo nome

Atributo Enumerado

- Cada um dos **valores possíveis** que o atributo pode assumir está explicitamente enumerado na declaração
- O atributo pode assumir **apenas um** dos valores especificados na sua declaração

```
<!ATTLIST produto qualidade
(BOA|RUIM|INDIFERENTE)>
```

Definição do atributo qualidade

Atributo ID

- Os **ID** identificam **unicamente** elementos individuais em um documento
- Todos os valores usados para ID em um documento devem ser **diferentes**
- Os elementos podem ter um **único atributo** ID
- O valor de um atributo do tipo ID deve ser **único em um documento** XML a fim de que o documento seja válido

```
<!ATTLIST produto codigo ID>
```

Definição do atributo codigo

```
<produto codigo="P123"/>
```

Exemplo válido de elemento produto

Atributo IDREF

- O valor de um atributo **IDREF** deve ser o valor de um **único atributo** ID em algum elemento no documento

```
<!ATTLIST estoque referencia IDREF>
```

Definição do atributo

```
<estoque referencia ="P123"/>
```

Documento válido

Atributo IDREFS

- É uma **variação** do tipo IDREF
- O valor de um atributo **IDREFS** pode conter valores IDREF **múltiplos** separados por espaços em branco

```
<!ATTLIST estoque referencias IDREFS>
```

Definição do atributo

```
<estoque referencias="P123 P456"/>
```

Documento válido

Definindo Referências

```
<!ELEMENT familia (pessoa)*>
<!ELEMENT pessoa (nome)>
<!ELEMENT nome (#PCDATA)>
<!ATTLIST pessoa
                id ID #REQUIRED
                mae IDREF #IMPLIED
                pai IDREF #IMPLIED
                filhos IDREFS #IMPLIED
```

Familia.dtd

Usando Referências

```
<familia>
  <pessoa id="joão" filhos=" joana pedro">
    <nome> João Lima </nome>
  </pessoa>
  <pessoa id="maria" filhos=" joana pedro">
    <nome> Maria Costa </nome>
  </pessoa>
  <pessoa id="joana" mae="maria" pai="joão">
    <nome> Joana Costa Lima </nome>
  </pessoa>
  <pessoa id="pedro" mae="maria" pai="joão">
    <nome> Pedro Costa Lima </nome>
  </pessoa>
</familia>
```

Família.xml

73

Valores Default

- *Required*
  - O atributo deve ter um **valor explicitamente especificado** em cada ocorrência do elemento no documento

```
<!ATTLIST produto nome CDATA #REQUIRED]>
```

- *Implied*
  - O valor do atributo **não é obrigatório**, e nenhum valor padrão é fornecido. Se um valor não é especificado, o processador XML deve proceder sem um

```
<!ATTLIST produto descricao CDATA #IMPLIED]>
```

74

Valores Default

- *Fixed*
  - Um valor é **fornecido** na declaração
  - **Nenhum** valor precisa ser fornecido no documento
  - O **processador** XML passará o valor para a aplicação
  - Se um valor for fornecido no documento então ele **deve corresponder** ao valor fornecido na declaração

```
<!ATTLIST produto qtd_minima CDATA
      #FIXED "15">
```

75

Declaração de Entidades

- Entidades são usadas para **representar** caracteres especiais
- As entidades também são usadas para **referenciar** um texto freqüentemente repetido ou alterado
- Declarações de entidades permitem **associar** um nome com algum fragmento de conteúdo
- O conteúdo pode ser
  - Trecho de **texto normal**
  - Trecho de declaração de **tipo de documento**
  - Uma referência a um **arquivo externo**

76

Declarações de Entidades

- Entidades **internas**
  - Entidades internas **associam um nome** com uma cadeia de caracteres ou texto
  - Entidades internas permitem **definir atalhos** para textos freqüentemente digitados ou textos a serem alterados
- Entidades **externas**
  - Entidades externas associam um nome com o conteúdo de um **outro arquivo**

77

Declarações de Entidade

- Entidade Interna

```
<!ENTITY Direitos "Este produto é fabricado pela Intel.">
```

Declaração de uma entidade interna

- Entidade Externa

```
<!ENTITY Direitos SYSTEM "direitos.txt">
```

Declaração de uma entidade externa

- Referenciando uma entidade

```
<manual>
  &Direitos;
</manual>
```

78

XML SCHEMA

79

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**

80

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>
```

81

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">
```

82

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"> ....
```

83

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>...
```

84

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 elemento 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

```
<?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 – 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**

91

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

92

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**

93

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>
```

94

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>
```

95

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"/>
```

96

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**

97

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**

98

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>
```

99

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**)

100

XML Schema - Grupos

- Grupos especificam restrições sobre um conjunto fixo de subelementos, que 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

101

XML Schema – Sintaxe Básica

- Os atributos de um *ComplexType* são declarados através da tag *<attribute>* e devem ser do tipo ***simpleType***
- Um atributo pode ser declarado como opcional através 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***

102

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>
.....
```


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 livros.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