

# OD 2002

***Tendências em XML***

***XML Schema***

***XPath 2.0***

***XSLT 2.0***

***XQuery***

...

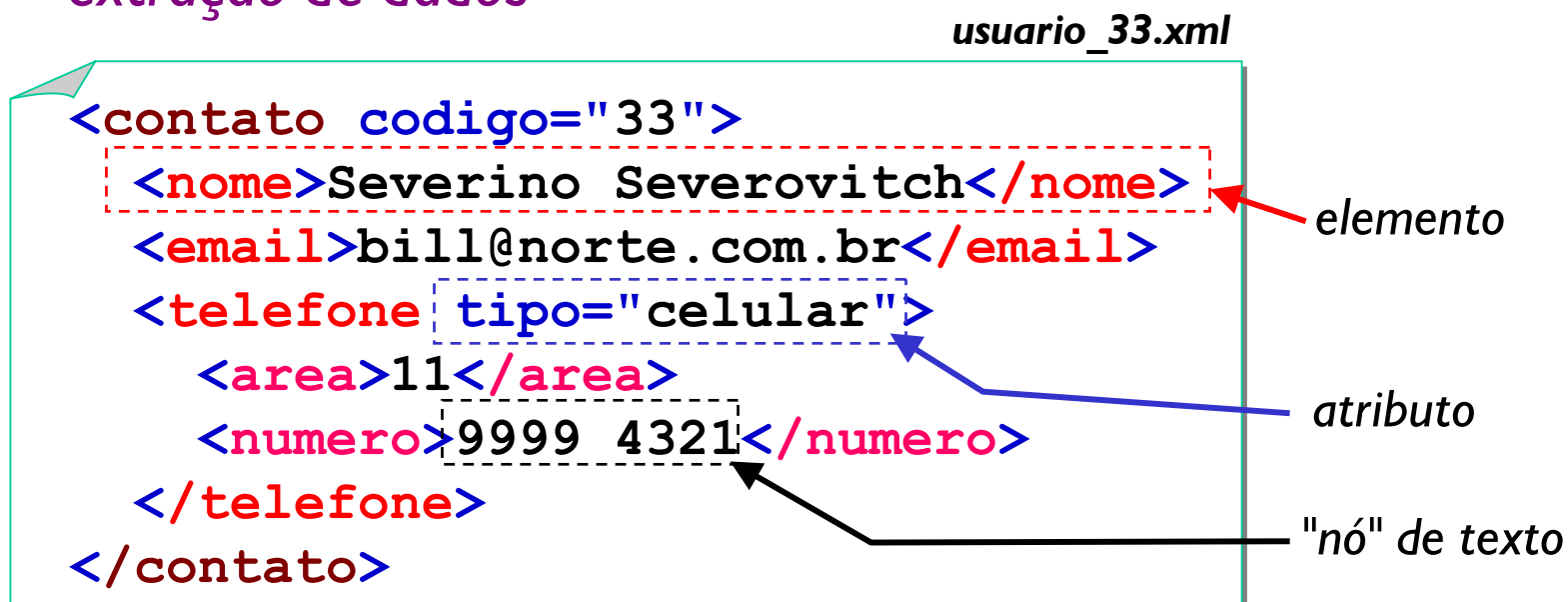
Helder da Rocha ([helder@acm.org](mailto:helder@acm.org))

- *Discutir as novidades desenvolvidas nos grupos de trabalho do W3C em relação à tecnologias relacionadas ao XML*
  - *Novas formas de **validar** XML (especificação XML Schema concluída em 2001)*
  - *Melhoramentos nos recursos para **pesquisa** (XPath 2.0, a ser concluída em 2002)*
  - *Novas linguagens funcionais para **extração de dados** (XQuery 1.0 a ser concluída em 2002)*
- *Finalidade: saber **quais** tecnologias usar e **como** usá-las para manipular dados em XML*

- *Fundamentos e XML Schema*
  - *XML e namespaces*
  - *Tecnologias de transformação*
  - *Validação e definição de tipos (DTD e XML Schema)*
  - *Fundamentos de XML Schema*
- *XPath e XSLT*
  - *O que é XPath / O que é XSLT*
  - *XPath 2.0*
  - *XSLT 2.0*
- *XQuery*
  - *O que é XQuery*
  - *Exemplos de XQuery*
  - *XQuery versus XSLT*

# Fundamentos e XML Schema

- XML é uma especificação que determina regras para a criação de documentos genéricos que
  - Acrescentam informação semântica a texto através de <marcadores>
  - São estruturados, facilitando a manipulação, pesquisa e extração de dados



# Documentos XML são representados ...

elemento raiz

declaração XML

nó raiz (/)

```
<?xml version="1.0" encoding="iso-8859-1" ?>
```

```
<cartao-simples>
```

```
<logotipo href="/imagens/logo14bis.gif" />
```

```
<nome>Alberto Santos Dumont</nome>
```

```
<endereco>Rua do Encanto, 22 - 2o. andar -  
Centro - 25600-000 - Petrópolis - RJ</endereco>
```

```
<email>dumont@14bis.com.br</email>
```

```
<telefone tipo="residencial">
```

```
<ddd>21</ddd>
```

```
<numero>2313011</numero>
```

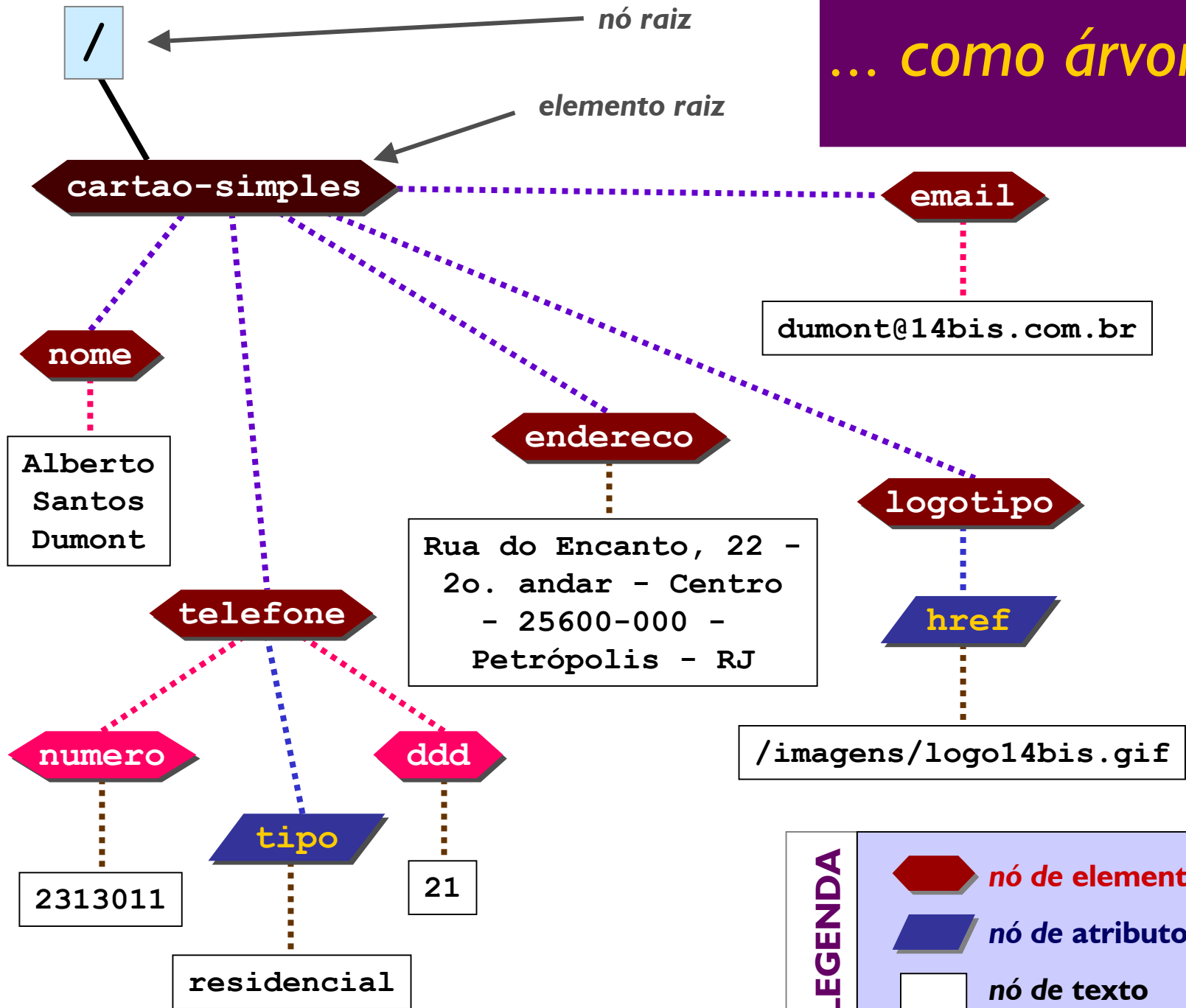
```
</telefone>
```

```
</cartao-simples>
```

atributos

elementos

# ... como árvores

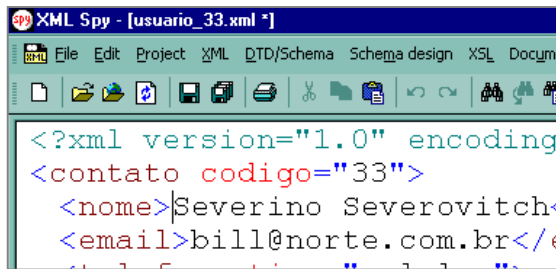


## LEGENDA

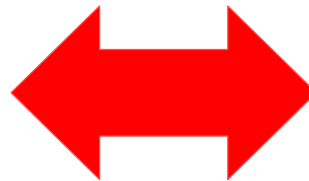
-  nó de elemento
-  nó de atributo
-  nó de texto

# Produção e manipulação de XML

- **XML** é texto e pode ser produzido em qualquer editor de textos...

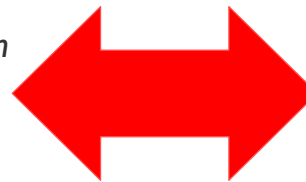
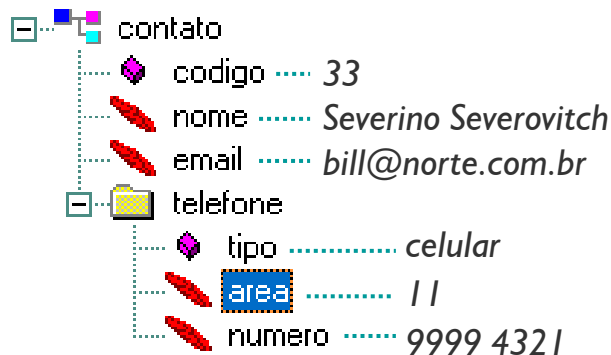


```
<?xml version="1.0" encoding="UTF-8" ?>
<contato codigo="33">
  <nome>Severino Severovitch</nome>
  <email>bill@norte.com.br</email>
  <telefone tipo="celular">
    <area>11</area>
    <numero>9999 4321</numero>
  </telefone>
</contato>
```



```
<contato codigo="33">
  <nome>Severino Severovitch</nome>
  <email>bill@norte.com.br</email>
  <telefone tipo="celular">
    <area>11</area>
    <numero>9999 4321</numero>
  </telefone>
</contato>
```

- ... depois pode ser facilmente manipulado como uma árvore na memória



```
<contato codigo="33">
  <nome>Severino Severovitch</nome>
  <email>bill@norte.com.br</email>
  <telefone tipo="celular">
    <area>11</area>
    <numero>9999 4321</numero>
  </telefone>
</contato>
```

# XML Namespaces

- Consiste da associação de um **identificador** a cada elemento/atributo da linguagem, que pode ser
  - **herdado** através do escopo de uma sub-árvore
  - **atribuído explicitamente** a elementos e atributos através de um **prefixo**
- Parte acrescentada posteriormente à especificação
  - **Evita conflitos e viabiliza a composição de documentos com vocabulários similares**

- **Exemplo**

```
<cadastro xmlns:firma="01.234.567/0001-99">  
  <nome>Severino Severovitch</nome>  
  <firma:nome>Sibéria Informática Ltda.</firma:nome>  
  <email>bill@norte.com.br</email>  
</cadastro>
```

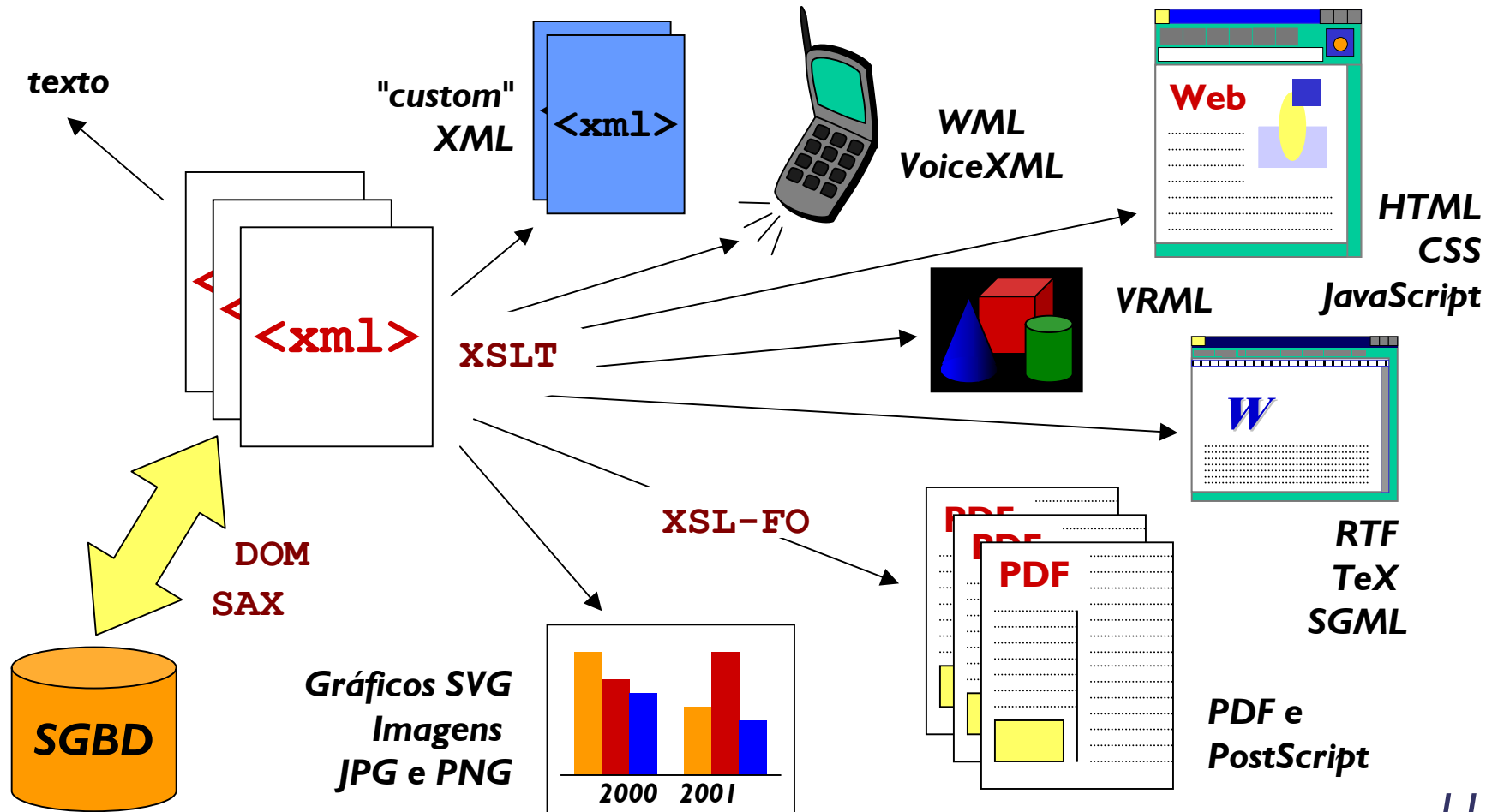
Este elemento <nome> pertence a outro namespace

# Tecnologias relacionadas ao XML

- Desde sua criação, várias "tecnologias relacionadas" têm surgido para facilitar a manipulação de XML
  - **DOM** e **SAX**: APIs para processamento em baixo nível
  - **XSLT** e **XQuery**: linguagens funcionais para extração de dados e transformação
  - **XPath** e **XPointer**: linguagens baseadas em caminhos para pesquisa e navegação na árvore XML
  - **XML Schema**: linguagem para validação, especificação e definição de tipos de dados
  - **XSL-FO** e **XHTML**: linguagens para formatação de XML com o objetivo de visualização
  - **Xlink**: linguagem para representar vínculos em XML

# Tecnologias para manipular XML

- Utilização das tecnologias para manipulação de XML



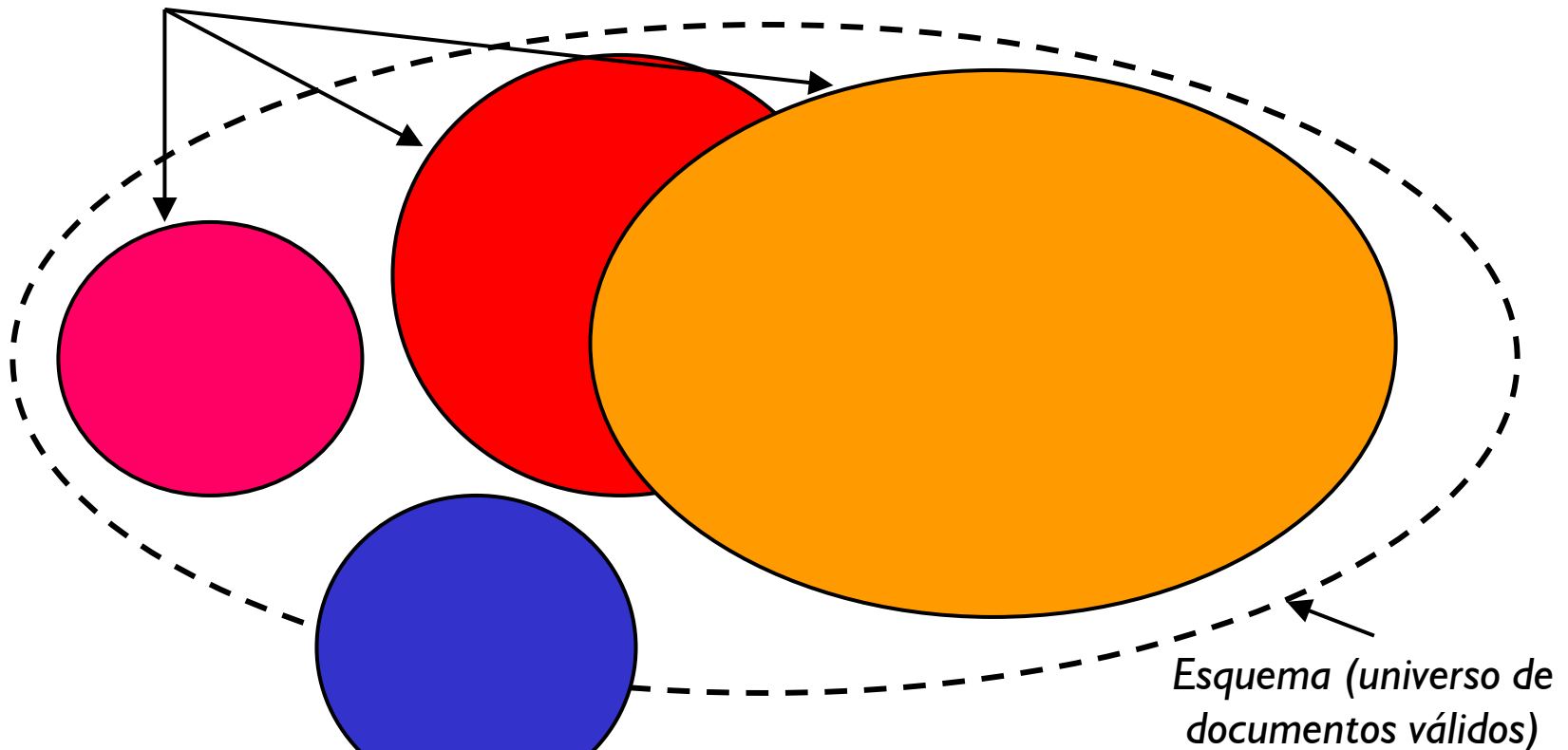
# Validação de XML

- *Para que possa ser manipulado como uma árvore, um documento XML precisa ser **bem formado***
  - *Segue regras mínimas de sintaxe*
  - *Documentos que não são bem formados não são XML*
- *Para ser utilizado em determinada aplicação, documento deve ser **válido***
  - *Deve ter determinado vocabulário, regras gramaticais, etc.*
- *Validade é definida em um **esquema***
  - ***Especifica** uma linguagem*
  - *Viabiliza a comunicação*
  - *Validação refere-se à comparação de um documento com um determinado esquema*

# Esquema

Documentos que aderem à especificação (válidos)

- O esquema representa uma **classe**
- Os documentos são **instâncias**



- Como definir esquemas:
  - **DTD** - Document Type Definition
  - **W3C XML Schema**

# DTD vs. XML Schema

- Um esquema é essencial para que haja **comunicação usando XML**
  - *Pode ser estabelecido "informalmente" (via software)*
  - *Uso formal permite validação usando ferramentas genéricas de manipulação de XML*
- Soluções

## DTD

```
<!ELEMENT contato  
    (nome, email, telefone)>  
<!ATTLIST contato  
    codigo NMTOKEN  
#REQUIRED
```

- *Simple mas não é XML*
- *Não suporta namespaces*
- *Limitado quando a tipos de dados*

## XML Schema

```
<xsd:schema  
    xmlns:xsd=".../XMLSchema">  
<xsd:element name="contato">  
    <xsd:complexType>  
        <xsd:attribute name="codigo"
```

- *É XML, porém mais complexo*
- *Suporta namespaces*
- *Permite definição de tipos*

# Exemplo de DTD

Exemplo de documento válido  
em relação a este DTD

```
<astro id="p05" nome="Jupiter">  
  <imagem href="jup31.jpg" />  
  <imagem href="jup32.jpg" />  
</astro>
```

Modelo de conteúdo  
(tipo de dados complexo)

```
<!ELEMENT astro (imagem*) >  
<!ELEMENT imagem EMPTY >
```

← Elementos

← Atributos

```
<!ATTLIST imagem href CDATA #REQUIRED >  
<!ATTLIST astro id ID #REQUIRED >  
<!ATTLIST astro nome CDATA #REQUIRED >  
<!ATTLIST astro diametrokm NMTOKEN #IMPLIED >
```

Atributo sempre  
associado a elemento

Tipos de dados simples  
(somente para atributos)

- XML Schema consiste de três especificações
  - *0: Primer - introdução*
  - *1: Structures - define estruturas (tipos complexos)*
  - *2: Datatypes - define coleção de tipos simples*
- Tipos **simples** representam tipos de dados básicos como texto, números, tokens, booleanos
  - *Podem representar qualquer informação não-XML (texto, banco de dados, etc.)*
  - *Novos tipos podem ser criados restringindo ou estendendo um tipo existente*
- Tipos **complexos** representam estruturas do documento como entidades, atributos, etc.
  - *Podem ser criados e derivados de outros*

# Exemplo de XML Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
  <xs:element name="astro" type="astroType" />  
  <xs:element name="imagem" type="imagemType"/>  
  
  <xs:attribute name="href" type="xs:anyURI"/>  
  <xs:attribute name="id" type="xs:ID"/>  
  <xs:attribute name="nome" type="xs:string"/>  
  <xs:attribute name="diametrokm" type="xs:decimal"/>  
  
  <xs:complexType name="imagemType">  
    <xs:attribute ref="href" use="required"/>  
  </xs:complexType>  
  
  <xs:complexType name="astroType">  
    <xs:sequence>  
      <xs:element ref="imagem" minOccurs="0"/>  
    </xs:sequence>  
    <xs:attribute ref="id" use="required"/>  
    <xs:attribute ref="nome" use="required"/>  
    <xs:attribute ref="diametrokm"/>  
  </xs:complexType>  
</xs:schema>
```

← Elementos

← Atributos

← Tipos simples

← Definição de tipos complexos

# Exemplo equivalente (design "boneca russa")

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="astro">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="imagem" minOccurs="0">
          <xs:complexType>
            <xs:attribute name="href" type="xs:anyURI"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="required"/>
      <xs:attribute name="nome" type="xs:string"/>
      <xs:attribute name="diametrokm" type="xs:decimal"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

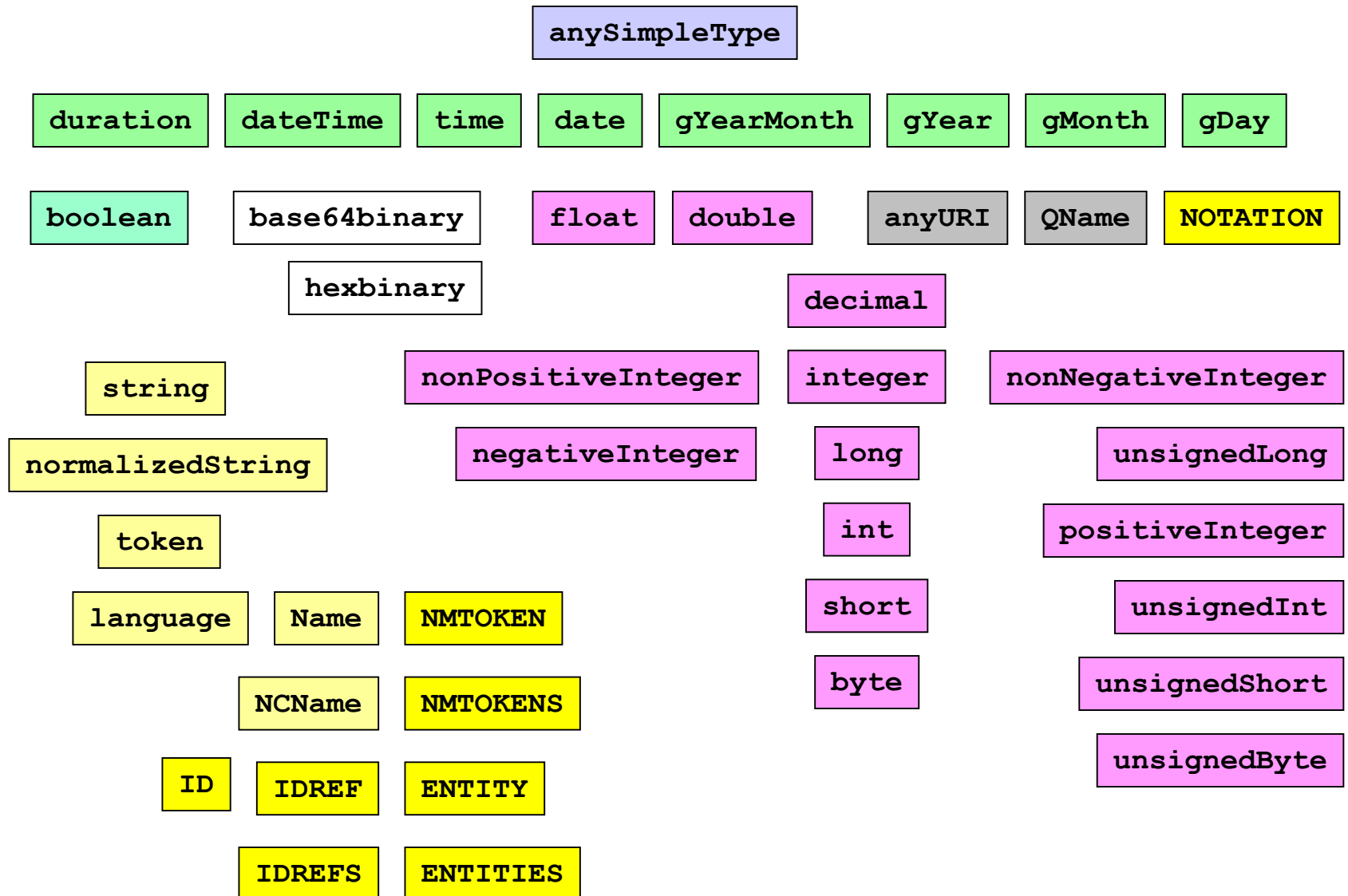
*Apenas um elemento (o elemento raiz) é visível*



*Tipos não podem ser reutilizados*

*Permite elementos de mesmo nome em contexto diferente*

# Tipos simples do XML Schema



# Como usar XML Schema

- É preciso usar um parser que suporte XML Schema
- Um documento XML pode estar associado a vários documentos XML Schema - um por namespace
  - XML Schema valida uma linguagem (namespace)
  - DTD valida um documento
- Associação pode ocorrer em qualquer elemento
- Associação sem namespace

```
<sisistemaEstelar
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="sisistemaEstelar.xsd">...
```

- Associação com dois namespaces

```
<se:sisistemaEstelar xmlns:se="http://cosmos.org.br"
  xmlns:sat="http://cosmos.org.br/sat"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://cosmos.org.br sistema.xsd
  http://cosmos.org.br/sat satelites.xsd">
```

# Por que usar XML Schema?

- *Mais recursos de validação*
- *Suporte à definição, extensão e reuso de tipos de dados simples e complexos*
- *Obrigatório para uso de outras tecnologias XML*
  - *Web Services (SOAP, WSDL, UDDI)*
  - *XSLT 2.0*
  - *XPath 2.0*
  - *XQuery*
- *Especificação estável (recomendação desde 2001)*
- *Já existem várias implementações estáveis*
  - *MSXML 4.0 em diante*
  - *Xerces 1.2*

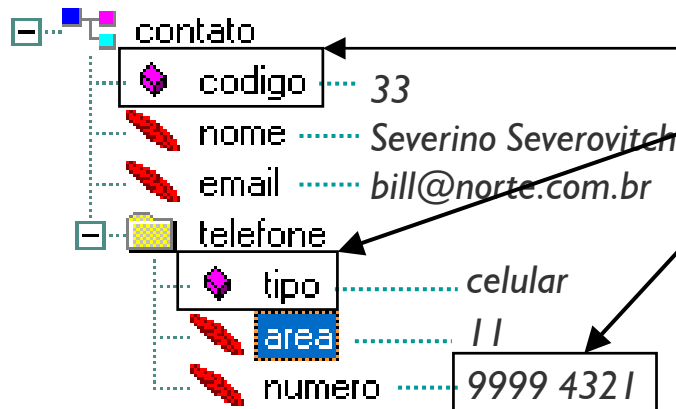
- *XML Schema 1.0 oferece recursos que permitem*
  - *Rigorosa especificação e validação de linguagens (e não apenas "documentos") XML*
  - *Reuso, importação e inclusão de esquemas*
  - *Criação, reuso, extensão e herança de tipos complexos*
  - *Reuso, derivação e restrição de tipos simples*
- *Pode-se usar apenas as especificações de interesse*
  - *É comum o uso apenas da especificação que define tipos de dados simples (XML Schema 1 - Datatypes)*
  - *Várias outras tecnologias relacionadas ao XML dependem da especificação "XML Schema 1 - Datatypes", entre elas XQuery, XPath 2.0, XSLT 2.0, tecnologias para Web Services e outras*

# **XPath e XSLT**

- *XPath (1.0)*
  - Usada para representar *caminhos* para os nós de um documentos XML
- *XSLT (eXtensible Stylesheet Language Transformations)*
  - Metade que se separou da especificação *XSL*
  - *XSL = XSLT (transformação) XSL-FO (formatação)*
  - Não visa necessariamente a apresentação: define regras para gerar texto ou XML a partir de dados extraídos de um documento-fonte XML
  - Requer um processador que saiba interpretar XSLT e aplicar as regras no documento-fonte
  - Depende de XPath para representar caminhos ou padrões de busca no documento-fonte

- **Consiste de três partes**

- **Modelo de dados:** tipos de primeira importância são strings, números, booleanos e node-sets (conjunto de nós)
- **Linguagem de expressões:** passos, eixos, predicados que representam um caminho na árvore XML



- **Caminhos absolutos**

- `/contato/@codigo`
- `/contato/telefone/@tipo`
- `/contato/telefone/numero/text()`

- **Relativos ao contexto** `/contato:`

- `@codigo` (ou `./@codigo`)
- `telefone@tipo` (ou `./telefone/tipo`)
- `telefone/numero/text()`

- **Operadores e funções:** para expressões booleanas, aritméticas, concatenação, indexação, etc.

# Expressões XPath

- Uma expressão XPath consiste de três dimensões

- **Passo** (separados por "/")

```
/contato/telefone/numero/text()
```

- **Eixo** (atributo, namespace, elemento pai, filho, etc.)

```
/child::contato/child::telefone/attribute::tipo
```

```
ou /contato/telefone/@tipo
```

```
/descendant-or-self::telefone/@tipo
```

```
ou //telefone/@tipo
```

- **Predicado** opcional (restringe busca com critério booleano)

```
/telefone[position()=2]/@tipo[.='residencial']
```

```
//telefone[numero='9999 4321' and not(area='21')]
```

- Em XSLT, XPath é usado dentro de atributos especiais para selecionar nós no documento-fonte

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="livro/titulo">
    <td><xsl:value-of select="." /></td>
```

## ■ XSL Transformations

- Linguagem funcional para criação de documentos que contêm regras de transformação para documentos XML
- Documentos escritos em XSLT são chamados de **folhas de estilo** e contêm
  - **Elementos XSLT**: <template>, <if>, <foreach>, ...
  - **Expressões XPath** para localizar nós da árvore-fonte
  - **Texto ou XML** a ser gerado no documento-resultado
- **Processador XSLT**

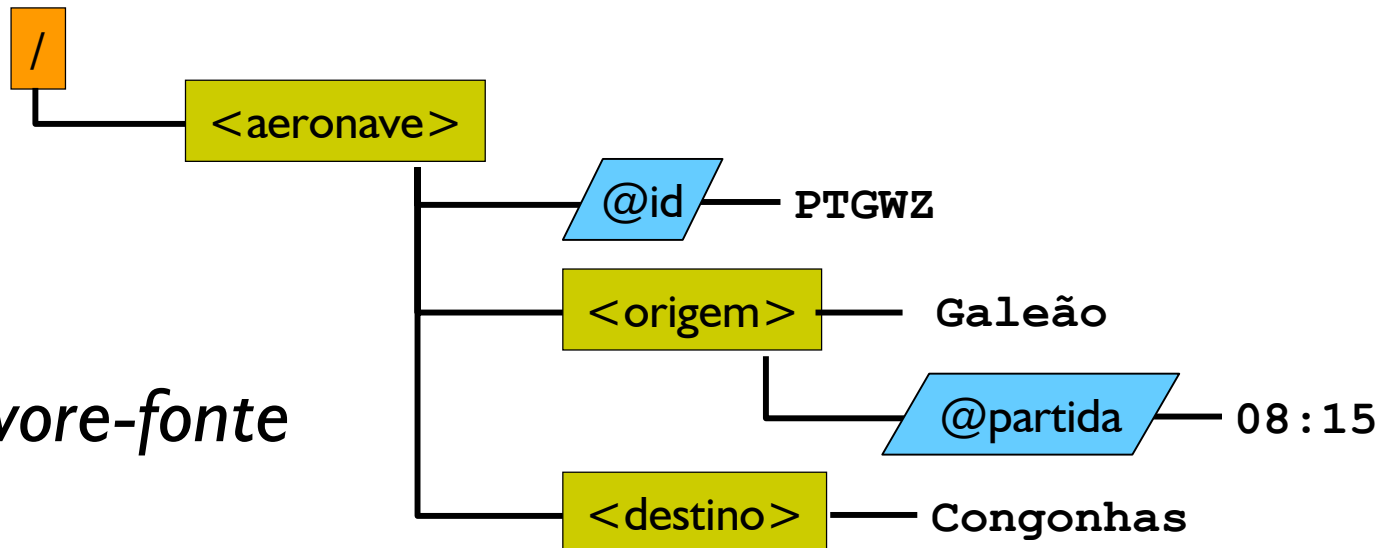


# Exemplo com XSLT: documento-fonte (I)

- Considere o seguinte documento-fonte:

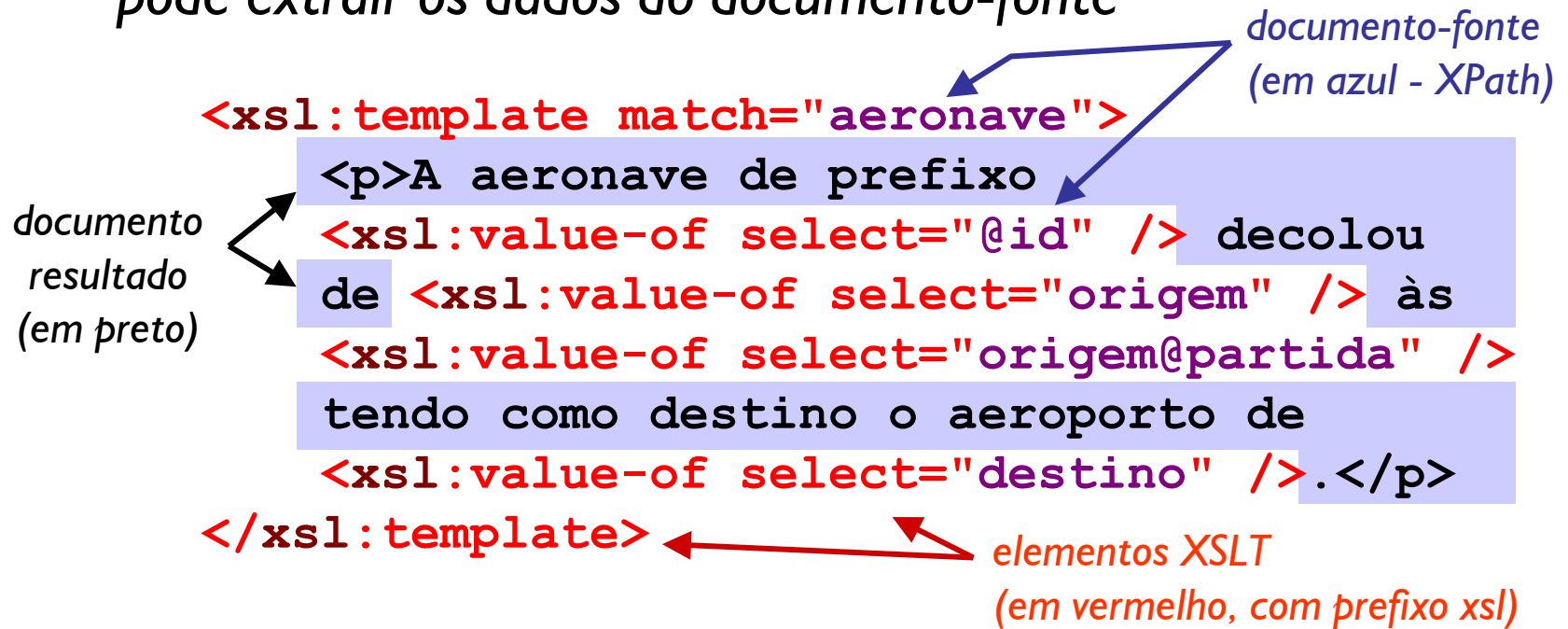
```
<aeronave id="PTGWZ">  
  <origem partida="08:15">Galeão</origem>  
  <destino>Congonhas</destino>  
</aeronave>
```

- *Árvore-fonte*



# XSLT: folha de estilos (2)

- O seguinte **template** (parte de uma folha de estilos XSLT) pode extrair os dados do documento-fonte



- Elementos XSLT geralmente são usados com um **prefixo** associado ao seu **namespace**: `<xsl:elemento>` para evitar conflitos com o documento-resultado.

## XSLT: documento-resultado (3)

- *Após a transformação, o resultado será*

```
<p>A aeronave de prefixo  
PTGWZ decolou  
de Galeão às  
8:15  
tendo como destino o aeroporto de  
Congonhas.</p>
```

- *Para obter outros resultados e gerar outros formatos com os mesmos dados, deve-se criar folhas de estilo adicionais*

# Problemas de XSLT e XPath

- *XPath não utiliza XML Schema*
  - *Só há quatro tipos de dados: número, string, booleano e node-set (conjunto de nós)*
- *XPath não lida com conjuntos ordenados*
  - *Não há um tipo "seqüência"*
- *XSLT não utiliza XML Schema*
  - *Geralmente, não se valida folhas de estilo uma vez que DTDs não sabem lidar com namespaces*
- *XSLT complica*
  - *Muito código é necessário para fazer coisas que são simples em outras linguagens (ex: agrupamento)*
  - *É linguagem completa (Turing-complete) em XML*

- **Nova especificação (esperada em dezembro 2002)**
  - *Fundamental para XQuery 1.0 e XSLT 2.0*
  - *Modelo de dados ampliado e melhorado*
  - *Suporta tipos XML Schema*
- **Novo objeto de primeira importância: **sequence****
  - *Contém valores de tipos simples (tipos XML Schema) ou nós (uma seqüência não pode conter outras seqüências)*
  - *Seqüência de um item só é o mesmo que o próprio item*
- **Modelo de dados compatível com XML Infoset**
  - *Definições compatíveis com outras especificações XML (XML InfoSet é um vocabulário uniforme que serve de referência para uniformizar definições através das diferentes especificações dos grupos de trabalho XML)*

# Algumas novidades do XPath 2.0

- **Comentários**

```
<xsl:value-of select="( -- Isto é um  
comentário --) /nome[@id='teste']"/>
```

- **Coringas de namespace**

```
<xsl:template match="*:description"> ...
```

- **Funções podem ser usadas como passos de uma expressão**

```
document("arquivo.xml")//elemento
```

- **Expressões entre parênteses podem ser passos**

```
/livro/(nota | capitulo)/titulo
```

- *Seqüência simples*  
*(1, 2, 3), ("aa", "bb"), (1 to 13)*
- *Podem ter tipos misturados*  
*(1, 2, 3, "abc", xf:date("2002-11-1"))*
- *Não podem ser aninhadas (uma seqüência não pode conter outra seqüência)*
- *Não são conjuntos (node-sets)*
  - *São ordenadas e podem conter elementos duplicados*
- *Equivalência*
  - *Seqüência contendo apenas um elemento é considerada igual ao próprio elemento*

# Exemplo de seqüência

- Usando folha de estilo XSLT

```
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <numeros>
      <xsl:for-each select="(1 to 5)">
        <n><xsl:value-of select="."/></n>
      </xsl:for-each>
    </numeros>
  </xsl:template>
</xsl:stylesheet>
```

seqüência XPath

- Resultado (documento gerado)

```
<?xml version="1.0" encoding="utf-8"?>
<numeros>
<n>1</n><n>2</n><n>3</n> <n>4</n><n>5</n>
</numeros>
```

# Operações com seqüências

- **União**

(1 to 10) | (5 to 15)

- Resultado: (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)

- **Intersecção**

(1 to 10) intersect (5 to 15)

- Resultado: (5, 6, 7, 8, 9, 10)

- **Diferença**

(1 to 10) except (5 to 15)

- Resultado: (1, 2, 3, 4)

# Novas expressões XPath 2.0

- *For*

```
for $var in expr return expr
for $var1 in expr,
    $var2 in expr,
    ...
    return expr
```

- *Condiciona*

```
if ( expr ) then expr else expr
```

- *Quantificadas*

```
some $nome in expr satisfies expr
```

```
every $nome in expr satisfies expr
```

# Funções e operadores do XPath 2.0

## ■ Funções

- Namespace: <http://www.w3.org/2001/12/xquery-functions> mapeado geralmente ao prefixo **xf**
- Várias são funções já existentes em XPath 1.0 adaptadas para usar tipos de dados do XML Schema
- Inclui construtores para tipos XML Schema

## ■ Operadores

- Namespace: <http://www.w3.org/2001/12/xquery-operators> mapeado geralmente ao prefixo **op**
- Vários são equivalentes aos já existentes: \*, /, -, +, and, or
- Outros lidam com datas, tipos específicos, etc.

# Exemplos de algumas funções XPath 2.0

- **xf:data (nó)**
  - Retorna uma *seqüência* de valores simples (devidamente identificados com tipos XML Schema)
- **xf:compare (string, string)**
  - Retorna 1, 0 ou -1 indicando resultado de comparação
- **xf:match (...)** e **xf:replace (...)**
  - Combinações e substituições com RegExp
- **xf:add-days (data?, decimal?)**
  - Acrescenta dias em uma data
- **xf:insert (item\*, decimal)**
  - Insere itens em determinada posição da seqüência
- **xf:current-dateTime ()**
  - Retorna data e hora atuais (tipo XML Schema xs:dateTime)

# Alguns operadores XPath 2.0

- **op:multiply (op1, op2)**
  - *Equivalente a \**
- **op:numeric-equal (op1, op2)**
  - *Equivalente a =*
- **op:boolean-or (boolean, boolean)**
  - *Equivalente a or*
- **op:concatenate (item\*, item\*)**
  - *Gera seqüência a partir da concatenação de dois itens ou seqüências*
- **op:datetime-less-than (dateTime, dateTime)**
  - *Retorna true se primeira data é anterior à segunda*
- **op:getDuration (dateTime, dateTime)**
  - *Retorna tempo transcorrido (tipo XML Schema xs:duration) entre duas datas*

- *Retornam tipos simples do XML Schema passando-se uma string como argumento*
- *Existe um para cada tipo do XML Schema*
- *Exemplos*
  - `xf:decimal ('12.3')`
  - `xf:integer ('12')`
  - `xf:token ('3testes')`
  - `xf:NMTOKEN ('teste')`
  - `xf:true ()`
  - `xf:date ('2002-12-12')`
  - `xf:gYear ('2002')`
  - `xf:anyURI ('http://www.aaa.com')`
  - ...

- *Atualização do XSLT compatível com XPath 2.0*
- *Suporta XML Schema*
  - *Folha de estilo pode ser validada!*
  - *Ordenação pode ser baseada no tipo XML Schema*
  - *Variáveis possuem atributo "type" que contém tipo XS*
- *Algumas otimizações*
  - *Mais fácil de lidar com agrupamentos*
  - *Variáveis podem receber node-sets*
  - *Pode gerar mais de um documento de saída*
  - *Suporte à formatação de datas*
- *Para definir um documento XSLT 2.0*
  - *Use o mesmo namespace*
  - *Defina o atributo `version="2.0"` em `<stylesheet>`*

# Novo elemento: `<xsl:for-each-group>`

- *Elimina a necessidade de sintaxe complexa para agrupamento por contexto*
  - *Em XSLT 2.0 era preciso usar recursos complicados como "método Muench" para agrupar elementos por contexto*

- *Sintaxe*

```
<xsl:for-each-group select="expr"  
  group-by="expr de string"  
  group-adjacent="expr de string"  
  group-starting-with="padrão">  
  <xsl:sort ... />  
  ...  
</xsl:for-each-group>
```

# XSLT 2.0 Permite definição de funções

- XSLT 1.0 apenas permite definição de templates reutilizáveis
- Funções são mais flexíveis e podem ser usadas por expressões XPath
- Exemplo: **fatorial(n)**

```
<xsl:function name="math:fatorial"
              xmlns:math="http://contas">
  <xsl:param name="n"
             type="xsd:nonNegativeInteger"/>
  <xsl:result type="xsd:positiveInteger"
             select="if ($n eq 0) then 1 else
                   $n * math:fatorial($n - 1)"/>
</xsl:function>
```

- *Value-of para seqüências*

- *Permite imprimir o conteúdo de uma seqüência*

```
<x><xsl:value-of select="(1,2,3,4)"  
separator="|" /></x>
```

- *Resultado*

```
<x>1|2|3|4</x>
```

- *Inclusão de texto*

- *XSLT 1.0 suporta apenas a inclusão de documentos XML (que são processados na inclusão) usando a função document()*

- *XSLT 2.0 suporta a inclusão de texto não processado*

```
<xsl:value-of  
select="unparsed-text('fragment.xml')"/>
```

- *XPath 2.0 melhora a linguagem XPath 1.0 com novos recursos e preserva a compatibilidade reversa*
  - *Suporte a XML Schema*
  - *Novas funções, novos operadores*
  - *Novo modelo de dados*
  - *Maior consistência e facilidade de integração com outras especificações de tecnologias relacionadas a XML*
- *XSLT 2.0 é incompatível com XSLT 1.0 mas*
  - *Suporta XML Schema*
  - *Suporta XPath 2.0*
  - *Oferece recursos que tornam a criação e utilização de folhas de estilo XSLT mais simples*

# XML Query

- *Especificação W3C que consiste de três partes*
  - **Modelo de dados** para documentos XML baseado no XML Infoset
  - **Conjunto de operadores** para o Modelo de Dados
  - **Linguagem de query** baseada nos operadores e no modelo de dados
- *XQuery é*
  - *Uma linguagem declarativa similar a SQL*
  - *Pode realizar pesquisas sobre um ou mais documentos*
  - *Pode construir novos documentos a partir dos resultados de uma seleção*
  - *Ideal para pesquisa: não suporta updates ou inserts*

# Modelo Relacional vs. XML

## Relacional

- **Banco relacional** contém tabelas
- **Tabela** relacional contém registros com mesmo esquema
- **Registro** relacional é lista de valores
- **SQL** query retorna conjunto não ordenado de registros

## XML

- **Banco XML** contém coleções
- **Coleção** contém documentos XML com mesmo DTD
- **Documento XML** é uma árvore de nós
- **XML Query** retorna uma seqüência não ordenada de nós

# Exemplo de uso de XQuery

- Adaptado de XQuery Use Cases (10)
- Documento a ser pesquisado (bib.xml):

```
<bib>
  <book year="1994">
    <title>TCP/IP Illustrated</title>
    <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
  </book>
  <book year="1992">
    <title>Advanced Programming in the Unix
      environment</title>
    <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
  </book>
  <book year="2000">
    <title>Data on the Web</title>
    <publisher>Morgan Kaufmann Publishers</publisher>
    <price>39.95</price>
  </book> ...
```

# Exemplo de Pesquisa XQuery

- Pesquisa mais simples é expressão XPath válida

```
document("bib.xml")//book
```

- Retorna todo o documento

- Pesquisa FOR

```
FOR $t
```

```
  IN document("bib.xml")/bib/book/title
```

```
  WHERE $b/publisher = "Addison-Wesley"
```

```
  RETURN $t
```

- Resultado

```
<title>TCP/IP Illustrated</title>
```

```
<title>Advanced Programming in the Unix  
Environment</title>
```

- Implementação de XQuery como um documento XML
  - *Objetivo é melhor processamento em detrimento da legibilidade*
- Exemplo do FOR (mais simples que o anterior) em XQueryX

```
<xq:query xmlns:xq="http://www.w3.org/2001/06/xqueryx">
  <xq:flwr>
    <xq:forAssignment variable="$t">
      <xq:step axis="CHILD">
        <xq:function name="document">
          <xq:constant datatype="CHARSTRING">bib.xml</xq:constant>
        </xq:function>
        <xq:identifier>bib</xq:identifier>
      </xq:step>
      <xq:step axis="CHILD">
        <xq:identifier>book</xq:identifier>
      </xq:step>
      <xq:step axis="CHILD">
        <xq:identifier>title</xq:identifier>
      </xq:step>
    </xq:forAssignment>
    <xq:return>
      <xq:variable>$b</xq:variable>
    </xq:return>
  </xq:flwr>
</xq:query>
```

```
FOR $t IN
  document("bib.xml")/bib/book/title
RETURN $b
```

# Expressões FLWR (FLoWeR)

- Representa as quatro tipos de sub-expressões do XQuery: **F**or, **L**et, **W**here, **R**eturn
- Sintaxe de FOR (*expr* = qq. expressão XQuery)
  - FOR var IN expr RETURN expr
  - FOR var IN expr [, var IN expr]\* [WHERE expr] RETURN expr
- Sintaxe de LET
  - LET var += expr RETURN expr
  - LET var += expr [, var := expr]\* [WHERE expr] RETURN expr
- Exemplo usando LET e FOR

```
FOR $book IN //book
  LET $authors := $book/author
  FOR $author IN $authors
    RETURN $author/last
```

# Exemplo com ordenação

## ■ Expressão

```
<bib> {  
  FOR $b IN  
    document("bib.xml")//book [publisher =  
      "Addison-Wesley" AND @year > "1991"]  
  RETURN <book>  
    { $b/@year }  
    { $b/title }  
  </book> SORTBY (title)  
} </bib>
```

*Veja mais exemplos em  
XQuery Use Cases no W3C  
(ref. (10))*

## ■ Resultado

```
<bib>  
  <book year="1992"><title>Advanced Programming in the  
  Unix Environment</title>  
  </book>  
  <book year="1994"><title>TCP/IP Illustrated</title>  
  </book>  
</bib>
```

# Diferenças entre XSLT e XQuery

- *Em tese, tudo o que se pode fazer com XQuery se pode fazer com XSLT*
- *XSLT é mais adequada à transformação de documentos (alteração);  
XQuery é mais adequada à pesquisa em documentos*
- *XSLT tem estrutura de documento (XML);  
XQuery tem estrutura de programa*
- *XSLT é linguagem funcional;  
XQuery é linguagem imperativa*

- *XQuery define um modelo de dados, operadores e linguagem que bancos de dados XML (ou simplesmente coleções de dados XML) sejam pesquisadas*
- *XQuery oferece recursos para XML similares aos recursos que SQL oferece bancos de dados relacionais*
  - *Quase. A versão atual ainda não implementa atualizações. XQuery pode ser usada para pesquisas*
- *XQuery 1.0 depende de XPath 2.0*
- *XQuery deve ser finalizado em dezembro de 2002*

# Referências

- 1. Priscilla Walmsley. *Definitive XML Schema*. Prentice-Hall, 2002
- 2. Michael Kay. *XSL Transformations 2.0*. W3C Working Draft, 16/08/2002.  
<http://www.w3.org/TR/xslt20/>
- 3. Anders Bergund et al. *XML Path Language (XPath) 2.0*. W3C Working Draft, 16/08/2002. <http://www.w3.org/TR/xpath20/>
- 4. W3C. *XML Query Specifications*. W3C, 16/08/2002.  
<http://www.w3.org/XML/Query>
- 5. Bas de Bakker and Irsan Widarto. *An Introduction to XQuery*. PerfectXML.  
<http://www.perfectxml.com/articles/xml/xquery.asp>
- 6. Elliotte Rusty Harold. *XSLT 2.0 and beyond*.  
<http://www.ibiblio.org/xml/slides/sd2002west/xslt2/index.html>
- 7. Anders Moeller. *XQuery Tutorial*.  
<http://www.brics.dk/~amoeller/XML/querying/>
- 8. Evan Lenz. *What's new in XPath 2.0*. O'Reilly XML.com.  
<http://www.xml.com/pub/a/2002/03/20/xpath2.html>
- 9. Evan Lenz. *What's new in XSLT 2.0*. O'Reilly XML.com.  
<http://www.xml.com/pub/a/2002/04/10/xslt2.html>
- 10. W3C. *XQuery Use Cases*. <http://www.w3.org/TR/2001/WD-xmlquery-use-cases-20010215>

*Obrigado.*

[\*www.argonavis.com.br\*](http://www.argonavis.com.br)