

XML - eXtensible Markup Language

■ Tópicos:

- ★ XML para transferência de dados
- ★ Estrutura hierárquica do XML
- ★ DTDs e XML Schema
- ★ Consultas de documentos XML:
 - ❖ Xpath
 - ❖ XQuery

■ Bibliografia:

- ★ Capítulo 30 do livro recomendado (7ª edição – Cap. 23 na 6ª edição)

O que é o XML?

- XML significa eXtensible Markup Language
- XML é uma linguagem de marcação (markup language) tal como o HTML.
- XML foi desenhado para descrever dados
- Marcadores (tags) em XML não estão predefinidos. Têm que ser definidos pelo programador
- O XML usa uma Document Type Definition (DTD) ou um XML Schema para descrever os tipos de dados
- XML com DTD ou XML Schema está desenhado para ser auto-descritivo
- XML é uma recomendação do W3C.
- Vem no seguimento do SGML (Standard Generalized Markup Language), mas é mais simples de usar

XML vs. HTML

- O HTML foi desenhado para mostrar os dados, focando **no seu aspecto**
- O XML foi desenhado para descrever dados, focando **no que eles são**
- No HTML, a sintaxe é fixa (o conjunto de tags está predefinido) e os marcadores definem simultaneamente os dados e a sua visualização.
- No XML, os dados, a sintaxe em que são descritos, e a forma como são visualizados são definidos separadamente.
- Ao contrário do HTML (que também deriva do SGML), o XML é **extensível**
- No XML, o utilizador pode adicionar novos tipos de *tags* e definir depois, separadamente, como lidar com as novas *tags* (nomeadamente para lidar com a apresentação).

Exemplo de XML

```
1.      <?xml version="1.0" encoding="ISO-8859-1"?>
2.      <recado>
3.          <de>Maria</de>
4.          <para>Manuel</para>
5.          <titulo>Lembrete</titulo>
6.          <corpo>Jantar hoje!</corpo>
7.      </recado>
```

- 1. contém o cabeçalho
- 2. contém o elemento do documento (significa que este documento é um recado)
- 3. a 6. contêm 4 elementos filhos da raiz
- 7. indica o fim da raiz
- **E o que é que isto tem a ver com Bases de Dados?!?**

XML para Bases de Dados

- A possibilidade de especificar novas tags, e de criar uma estrutura imbricada de tags, torna o XML uma ótima linguagem para troca de dados.
- As tags tornam os dados, de certa forma, auto-documentados

✱ E.g.

```
<banco>
  <conta>
    <num-conta> A-101    </num-conta>
    <balcao>    Lx       </balcao>
    <saldo>     500      </saldo>
  </conta>
  <depositante>
    <num-conta>  A-101   </num-conta>
    <nome-cliente> Paulo </nome-cliente>
  </depositante>
</banco>
```

XML: Motivação

- A troca de informação é uma tarefa crítica e com importância crescente

- ★ Exemplos:

- ❖ Banca: transferência de fundos
- ❖ Aplicações B2B (processamento de compras entre empresas)
- ❖ Comunicação de dados a autoridades (e.g. SAFT)
- ❖ Dados científicos
 - Química: ChemML, ...
 - Genética: BSML (Bio-Sequence Markup Language), ...

- ★ O fluxo de papel entre organizações tende a ser substituído por fluxo (eletrónico) de informação.

- Cada área de aplicação tem os seus próprios standards para representar a informação
- O XML tornou-se a base para a geração de formatos de troca de dados

XML: Motivação (Cont.)

- Cada standard baseado em XML define quais os elementos válidos a transmitir através de:
 - ★ Linguagens de especificação de tipos em XML
 - ❖ DTD (Document Type Definition)
 - ❖ XML Schema
 - ★ Descrição textual (ou formal via RDF, OWL) da semântica
- O XML permite que sejam definidas novas tags:
 - ★ Isto é restringido pelas DTDs
- Existe atualmente uma grande variedade de produtos para fazer parsing, transformar (em HTML) e pesquisar dados em documentos XML

Estrutura dos dados em XML

- **Tag**: label para uma secção de dados
- **Elemento**: secção de dados começada por `<tagname>` e terminada pelo correspondente `</tagname>`
- Os inícios e fins de secções têm que estar bem emparelhados (como estrutura de parêntesis)
 - ★ Exemplo correto
 - ❖ `<conta> ... <saldo> </saldo> </conta>`
 - ★ Exemplo incorrecto
 - ❖ `<conta> ... <saldo> </conta> </saldo>`
 - ★ Formalmente: A cada tag de início deve corresponder exatamente uma tag de fim, que está no mesmo contexto (onde um contexto é tudo o que está entre uma tag de início e a correspondente tag de fim)
- Um documento tem um único elemento de nível mais alto (raiz)

Exemplo de elementos imbricados

```
<banco-1>
  <cliente>
    <nome-cliente> Luís          </nome-cliente>
    <rua-cliente> 5 de Outubro </rua-cliente>
    <local-cliente> Lisboa      </ local-cliente>
    <conta>
      <num-conta> A-102   </num-conta>
      <agencia>  Caparica </agencia>
      <saldo>    400      </saldo>
    </conta>
    <conta>
      <num-conta> A-103   </num-conta>
      <agencia>  Lisboa </agencia>
      <saldo>    600      </saldo>
    </conta>
    <conta>
      ...
    </conta>
  </cliente>
  ...
</banco-1>
```

Motivação para dados imbricados

- A imbricação de dados não é suportada em bases de dados relacionais
 - ✦ Pode causar dados redundantes (e.g. informação de contas com vários clientes)
 - ✦ A normalização substitui estruturas imbricadas por existência de várias tabelas com restrições de chaves estrangeiras
- Mas imbricação de dados é importante na transferência de dados
 - ✦ A aplicação externa pode não ter acesso à tabela referenciada pelas chaves estrangeiras!

Estrutura dos dados em XML(Cont.)

- O XML permite misturar texto com sub-elementos

★ Exemplo:

<conta>

Esta conta já quase não é usada.

<num-conta> A-101 </num-conta>

<balcao> Lx </balcao>

<saldo> 500 </saldo>

</conta>

- ★ Isto é útil para marcação de documentos, mas **desencorajado** em representação de dados...

Atributos

- Os elementos podem ter **atributos**

```
<conta tipo-conta = "ordem" >  
    <num-conta> A-101    </num-conta>  
    <balcao>    Lx        </balcao>  
    <saldo>      500      </saldo>  
</conta>
```

- Os atributos são especificados dentro da tag de início, por pares da forma *nome=valor*
- Um elemento pode ter vários atributos, mas cada nome de atributo só pode ocorrer uma vez em cada tag
 - ★ `<conta tipo-conta = "ordem" limite-negativo="500">`

Atributos versus sub-elementos

■ Distinção entre subelementos e atributos

- ✦ No contexto de um documento, os atributos são partes da marcação, e subelementos são partes dos dados do documento propriamente dito.
- ✦ No contexto de representação de dados, a diferença não é tão clara
 - ❖ A mesma informação pode ser representada das duas formas
 - `<conta num-conta = "A-101"> </conta>`
 - `<conta>`
 `<num-conta> A-101</num-conta> ...`
 `</conta>`
- ✦ Sugestão: usar atributos para identificadores (ou chaves) de elementos, e usar subelementos para outros atributos

Mais sobre sintaxe XML

- A sintaxe de elementos sem sub-elementos nem texto, pode ser abreviada terminando a tag de início /> e evitando a tag de fim
 - ✱ <conta num-conta="A-101" agencia="Lx" saldo="200" />
- Para guardar strings que tenham a forma de tags (sem as querer interpretar como tal) usar
 - ✱ <![CDATA[<conta> ... </conta>]]>
 - ❖ Aqui, <conta> e </conta> são simplesmente strings de dados

Namespaces

- Os dados em XML podem servir para ser transferidos entre empresas
- A mesma tag pode ter significados diferentes em empresas diferentes, o que pode causar confusão na troca de documentos
- Solução: usar *nome-único:elemento-único*
- Para evitar a geração de nomes únicos grandes por todo o documento, podem usar-se os Namespaces do XML

```
<banco xmlns:FB='http://www.FirstBank.com'>
```

```
...
```

```
  <FB:agencia>
```

```
    <FB:nome-agencia> R. Ouro</FB:nome-agencia>
```

```
    <FB:local-agencia> Lx</FB:local-agencia>
```

```
  </FB:agencia>
```

```
...
```

```
</banco>
```

Esquemas de Documentos XML

- Em bases de dados, os esquemas definem que informação pode ser armazenada, com que estrutura, e quais os tipos dos vários valores
- Os documentos XML não têm obrigatoriamente um esquema associado
- Mas, quando para uso em transferência de dados, é importante associar esquemas a documentos XML
 - ★ Caso contrário, como poderia um site interpretar de forma automática dados recebidos de outro site?
- Dois mecanismos para especificar esquemas de XML:
 - ★ **Document Type Definition (DTD)**
 - ❖ Mais simples, ainda usado, mas cada vez menos
 - ★ **XML Schema**
 - ❖ Mais potente, e com uso cada vez maior

Document Type Definition (DTD)

- O tipo (esquema) de um documento XML pode ser especificado por uma DTD
- Uma DTD define a estrutura de dados permitida em XML
 - ✦ Que elementos podem aparecer
 - ✦ Que atributos pode ter (ou tem que ter) cada um dos elementos
 - ✦ Que subelementos podem (ou têm que) ocorrer dentro de cada elemento, e quantas vezes
- As DTD não servem para restringir tipos de dados
 - ✦ Todos os valores são representados como strings
- Sintaxe de uma DTD
 - ✦ `<!ELEMENT elemento (especificação-subelementos) >`
 - ✦ `<!ATTLIST elemento atributos >`

Especificação de elementos em DTDs

- Um subelemento pode ser especificado como:
 - ✱ nomes de elementos, ou
 - ✱ #PCDATA (parsed character data), i.e., strings
 - ✱ EMPTY (nenhum subelemento) ou ANY (tudo pode aparecer como subelemento)
- Exemplo
 - <!ELEMENT depositante (nome-cliente, num-conta)>
 - <!ELEMENT nome-cliente (#PCDATA)>
 - <!ELEMENT num-conta (#PCDATA)>
- A especificação de subelementos pode conter expressões regulares:
 - <!ELEMENT banco ((conta | cliente | depositante)+)>
 - ❖ Notação:
 - “|” - alternativas
 - “+” - 1 ou mais ocorrências
 - “*” - 0 ou mais ocorrências
 - “?” - 0 ou 1 ocorrência

Exemplo de DTD

```
<!DOCTYPE banco [  
  <!ELEMENT banco ( ( conta | cliente | depositante)+)>  
  <!ELEMENT conta (num-conta, agencia, saldo)>  
  <!ELEMENT cliente (nome-cliente, rua-cliente, local-cliente)>  
  <!ELEMENT depositante (nome-cliente, num-conta)>  
  <!ELEMENT num-conta (#PCDATA)>  
  <!ELEMENT agencia (#PCDATA)>  
  <!ELEMENT saldo (#PCDATA)>  
  <!ELEMENT nome-cliente (#PCDATA)>  
  <!ELEMENT rua-cliente (#PCDATA)>  
  <!ELEMENT local-cliente (#PCDATA)>  
>
```

Especificação de atributos em DTDs

■ Para cada atributo, especifica-se

- ★ Nome

- ★ Tipo do atributo

 - ❖ CDATA

 - ❖ ID (identificador) ou IDREF (referência para ID) ou IDREFS (múltiplos IDREFs)

- ★ Restrições adicionais

 - ❖ obrigatório (#REQUIRED)

 - ❖ Com valor por defeito (valor),

 - ❖ Nenhum dos dois (#IMPLIED)

■ Exemplos

- ★ `<!ATTLIST conta tipo-conta CDATA "ordem">`

- ★ `<!ATTLIST cliente`

 - `id-cliente ID #REQUIRED`

 - `contas IDREFS #REQUIRED >`

IDs e IDREFs

- Um elemento tem no máximo um atributo do tipo ID
- O valor dum atributo de tipo ID tem que ser único em todo o documento XML
 - ★ O atributo com tipo ID é um identificador do objeto (elemento)
- Um atributo do tipo IDREF tem que conter um valor que exista num atributo de tipo ID, no mesmo documento XML
- Um atributo do tipo IDREFS contém um conjunto de (0 ou mais) valores, onde cada um desses valores tem que existir como ID de algum elemento do mesmo documento

Exemplo de DTD com atributos

```
<!DOCTYPE banco-2[
  <!ELEMENT conta (agencia, saldo)>
  <!ATTLIST conta
    num-conta      ID      #REQUIRED
    clientes       IDREFS #REQUIRED>
  <!ELEMENT cliente (nome-cliente, rua-cliente,
    local-cliente)>
  <!ATTLIST cliente
    id-cliente      ID      #REQUIRED
    contas          IDREFS #REQUIRED>
  ... ..
]>
```

Documento XML com atributos ID e IDREF

```
<?xml version = "1.0" standalone = "no"?>
<!DOCTYPE banco-2 SYSTEM "http://centria.fct.unl.pt/~jja/banco2.dtd">
<banco-2>
  <conta num-conta="A-401" clientes="C100 C102">
    <agencia> Caparica </agencia>
    <saldo>500 </saldo>
  </conta>
  <cliente id-cliente="C100" contas="A-401">
    <nome-cliente> Luís </nome-cliente>
    <rua-cliente> R. República </rua-cliente>
    <local-cliente> Lx </local-cliente>
  </cliente>
  <cliente id-cliente="C102" contas="A-401">
    <nome-cliente> Maria </nome-cliente>
    <rua-cliente> R. 5 de Outubro </rua-cliente>
    <local-cliente> Porto </local-cliente>
  </cliente>
</banco-2>
```

Limitações de DTDs

- Não é possível especificar tipos de elementos e atributos
 - ✱ Todos os tipos são strings. Não há inteiros, reais, etc.
- É difícil especificar conjuntos (não ordenados) de subelementos
 - ✱ Em bases de dados a ordem é (normalmente) irrelevante
 - ✱ $(A \mid B)^*$ permite especificar um conjunto não ordenado, mas não permite garantir que cada um de A e B ocorre pelo menos uma vez (como fazer?)
- Os IDs e IDREFs são não tipados
 - ✱ O atributo *cliente* de uma conta pode ter referências a outras contas (o que não faz sentido!)
 - ❖ *clientes* deveria apenas conter referências para identificadores de clientes

XML Schema

- XML Schema é uma outra linguagem (mais sofisticada) para especificação de estrutura de documentos XML, que endereça os problemas dos DTDs.
- Tem mecanismos para
 - ★ Definir tipos de dados para elementos e atributos
 - ❖ E.g. inteiros, string, etc.
 - ❖ Valores min/max
 - ★ Tipos de dados para elementos e atributos *definidos pelo utilizador*
 - ★ Ao contrário do que acontece com DTDs, a sintaxe do XML Schema é a sintaxe do XML, tornando-os extensíveis a futuras adições
 - ★ Integrado com namespaces
 - ★ Outras características
 - ❖ Listas, restrições de chave primária e estrangeira, herança, etc...

Exemplo de DTD

```
<!DOCTYPE banco [  
  <!ELEMENT banco ( ( conta | cliente | depositante)+)>  
  <!ELEMENT conta (num-conta, agencia, saldo)>  
  <!ELEMENT cliente (nome-cliente, rua-cliente, local-cliente)>  
  <!ELEMENT depositante (nome-cliente, num-conta)>  
  <!ELEMENT num-conta (#PCDATA)>  
  <!ELEMENT agencia (#PCDATA)>  
  <!ELEMENT saldo (#PCDATA)>  
  <!ELEMENT nome-cliente (#PCDATA)>  
  <!ELEMENT rua-cliente (#PCDATA)>  
  <!ELEMENT local-cliente (#PCDATA)>  
>
```

Versão XML Schema do exemplo anterior

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="banco" type="TipoBanco"/>
  <xsd:element name="conta">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="num-conta" type="xsd:string"/>
        <xsd:element name="agencia" type="xsd:string"/>
        <xsd:element name="saldo" type="xsd:decimal"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

..... Definições para cliente e depositante....

```
<xsd:complexType name="TipoBanco">
  <xsd:sequence>
    <xsd:element ref="conta" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="cliente" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="depositante" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Versão XML Schema do exemplo anterior

- A escolha de “xsd:” é nossa – qualquer outro prefixo do espaço de nomes (namespace) podia ter sido escolhido.
- O elemento “banco” tem o tipo “TipoBanco” que é definido separadamente.
 - ★ xsd:complexType é usado a seguir para criar o tipo de dados complexo, com nome, “TipoBanco”
- O elemento “conta” tem o seu tipo definido in-line

Outras Características de XML Schema

■ Atributos especificados por *tag* `xsd:attribute`:

- ✱ `<xsd:attribute name = "num-conta"/>`

- ✱ Adicionando o atributo `use = "required"` , o valor passa a ter que ser obrigatoriamente especificado

■ Restrições de chaves: “números de conta formam a chave de elementos *conta* debaixo do elemento raiz *banco*”:

```
<xsd:key name = "accountKey">  
  <xsd:selector xpath = "/banco/conta"/>  
  <xsd:field xpath = "num-conta"/>  
</xsd:key>
```

■ Restrições de chaves estrangeira de depositante para conta:

```
<xsd:keyref name = "depositorAccountKey" refer="accountKey">  
  <xsd:selector xpath = "/banco/depositante"/>  
  <xsd:field xpath = "num-conta"/>  
</xsd:keyref>
```

Transformação e Consulta de dados XML

- Tradução de informação de um esquema XML para outro
- Relacionado com pesquisa de informação em documentos XML (e tratado pelas mesmas ferramentas)
- Linguagens para transformação/pesquisa em documentos XML
 - ★ XPath
 - ❖ Linguagem simples, que consiste em path expressions
 - ★ XQuery
 - ❖ Linguagem mais complexa de pesquisa de informação em documentos XML (com algumas semelhanças com SQL)
 - ★ XSLT
 - ❖ Linguagem desenhada para tradução de documentos XML para XML e XML para HTML
- Existem outras linguagens:
 - ★ XML-QL, Quilt, XQL, Xcerpt, ...

Modelo em árvore de documentos XML

- A maior parte das linguagens de transformação e pesquisa são baseadas no **modelo em árvore** de documentos XML
- Um documento XML pode ser visto como uma árvore, onde os nós correspondem a elementos e a atributos
 - ✦ Os nós-elementos têm filhos, que podem ser atributos ou subelementos
 - ✦ Texto num elemento é visto como um nó de tipo texto (sem filhos)
 - ✦ Os filhos dum nó são ordenados de acordo com a ordem em que aparecem no documento XML
 - ✦ Os nós de elementos e atributos têm sempre um único pai.
 - ✦ A raiz da árvore contém um único filho, correspondendo ao elemento raiz do documento

Documento XML de exemplo

```
<?xml version = "1.0"?>
<banco-2>
  <conta num-conta="A-401" clientes="C100 C102">
    <agencia> Caparica </agencia>
    <saldo>500 </saldo>
  </conta>
  <cliente id-cliente="C100" contas="A-401">
    <nome-cliente> Luís </nome-cliente>
    <rua-cliente> R. República </rua-cliente>
    <local-cliente> Lx </local-cliente>
  </cliente>
  <cliente id-cliente="C102" contas="A-401">
    <nome-cliente> Maria </nome-cliente>
    <rua-cliente> R. 5 de Outubro </rua-cliente>
    <local-cliente> Porto </local-cliente>
  </cliente>
</banco-2>
```