

Bancos de Dados XML

Formas de Gerência de Dados XML

- **SGBDRs estendidos com suporte à XML**

- adequados a dados XML fortemente estruturados
(*“documentos orientados a registros”*)

```
<endereço>  
  <rua>Beira-Mar</rua><numero>104</numero><complemento>apto 203</complemento>  
  <bairro>centro</bairro><cidade>Florianópolis</cidade> <cep>88010-600</cep>  
</endereço>  
<endereço>  
  <rua>Lauro Linhares</rua><numero>761</numero><bairro>trindade</bairro>  
  <cidade>Florianópolis</cidade><cep>88040-900</cep>  
</endereço>
```

- aplicações que realizam intercâmbio de dados convencionais em XML
 - dados de BD, arquivos, docs bem formatados em geral

Formas de Gerência de Dados XML

■ SGBDs XML Nativos

- suporta um **modelo de dados** proprietário para dados XML (elementos, atributos, ordem, ...)
- Adequado a dados fortemente semi-estruturados (*“documentos orientados a textos”*)
 - mapeamento para BD relacional seria complexo!
 - necessidade de consultas envolvendo padrões textuais
 - aplicações que lidam apenas com dados no formato XML

```
<anuncio>  
<transacao>Vendo</transação>, por motivo de viagem,<produto>automóvel Gol I 97  
</produto>, cor azul, em ótimo estado de conservação. Preço: R$<preco>9000,00</preco>.  
Tratar com<contato><nome>Pedro</nome> fone</fone>99991111</fone></contato>  
</anuncio>
```

BDs XML Nativos - Características

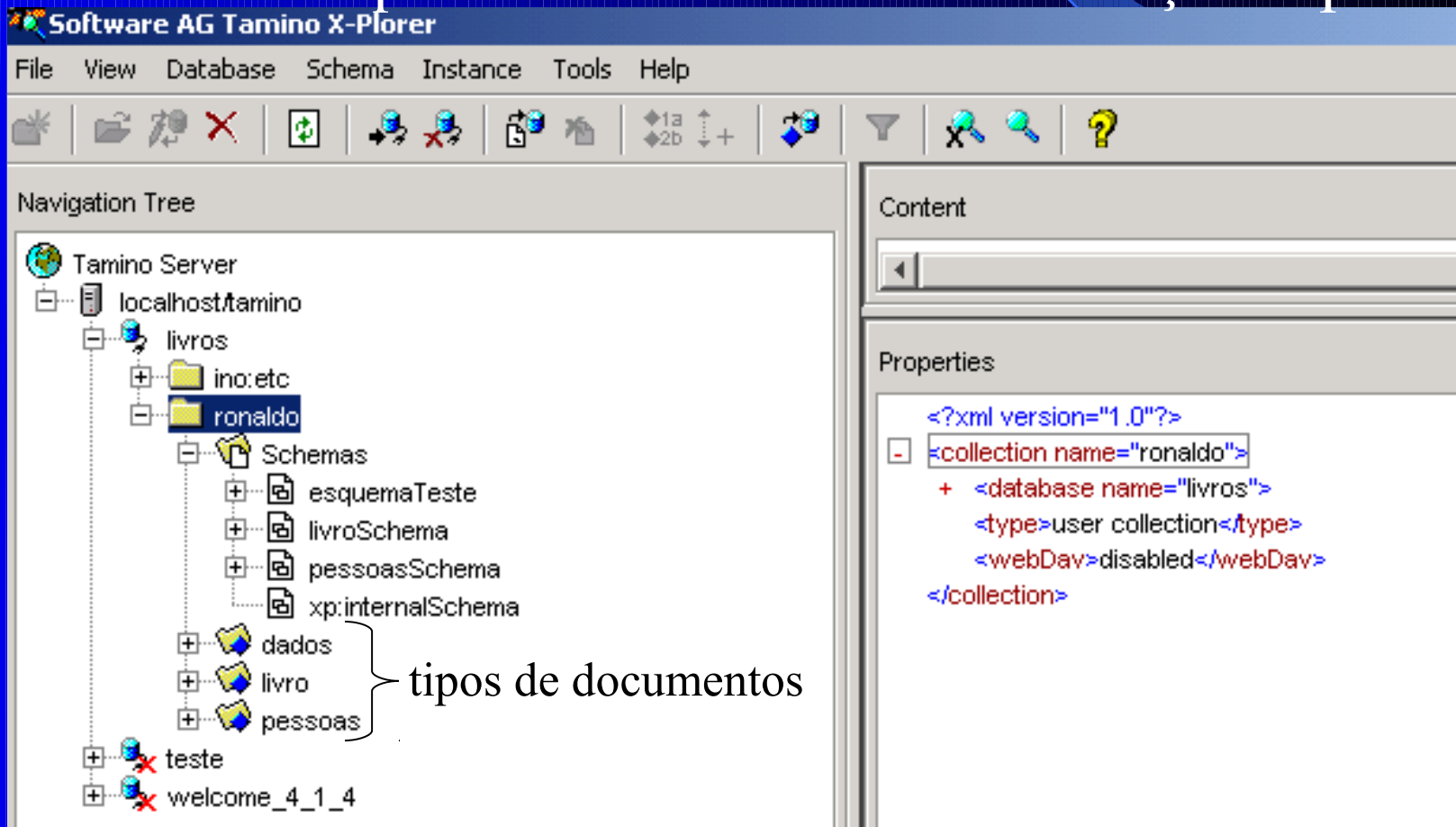
- Esquemas lógicos baseados em coleções
- Consultas
- Atualização
- Conectividade
- Armazenamento de dados
- Projeto de um BD XML

Coleções

- **Noção lógica de um conjunto de docs XML**
 - a decisão por quais docs XML pertencem a uma coleção fica em geral a cargo da aplicação
 - uma coleção pode estar restrita a um ou vários esquemas XML
 - geralmente representam dados de um mesmo domínio
- Consultas e atualizações podem ser direcionadas a coleções

Coleções - Tamino

- 1 BD – n coleções – n esquemas – n tipos de documentos
 - cada *tipo de documento* define um elemento raiz permitido
 - novo doc XML: inserido em uma coleção e válido para algum tipo doc
- Docs sem esquema mantidos em uma coleção específica



Consultas

- Suporte a pelo menos uma linguagem de consulta para XML
 - uso mais extensivo de *XPath*
 - uso de alguns dialetos da *XQuery* (tendência!)
- Características desejadas para uma linguagem de consulta para XML
 - buscas textuais (por palavras-chaves, por padrões, ...)
 - consultas declarativas
 - resultados de consultas
 - doc XML, fragmentos de docs XML ou novas estruturas XML

Consultas

■ Tamino

- consultas em *XPath* e *XQuery* estendidas
- suporta busca por padrão
 - `/livro[título ~= "*XML*"]/título`
- geração de docs XML como resultado

■ eXist

- consultas em *XPath* estendida
- suporta busca por padrão, por palavra-chave (em textos) e por proximidade
 - `/livro[título &= 'banco XML']/título`
 - `/livro/capitulo[near(., 'banco XML', 50)]/@nome`

Atualizações

- Capacidades de atualização são variadas
 - possibilidade apenas de substituição de um doc XML completo
 - API DOM para atualização de nodos
 - linguagens de atualização declarativas
 - tendência1: *XUpdate* (consórcio **XML:DB**)
 - **XML:DB**
 - consórcio de empresas responsável pelo desenvolvimento de tecnologias para BDs XML
 - tendência2: *XQuery* com capacidades de atualização

XUpdate

- Sintaxe XML
 - I / E de elementos, atributos e texto
 - A do conteúdo de elementos e atributos
- Exemplo 1:

```
<xupdate:appendselect="//autor[nome='Maria']/eMail" child="last() ">  
  <xupdate:element name="eMail">Maria@teste.com</xupdate:element>  
</xupdate:append>
```

(inclusão de um novo eMail para *Maria*)

- Exemplo 2:

```
<xupdate:remove select="/listalivros/livro[1]"/>
```

(remoção do primeiro livro)

Atualizações - Tamino

- *XQuery* possui capacidades de atualização
 - *insert, delete, rename e replace*

- Exemplos

- **update** (inserção de autor)
for \$liv in input()/livro
where \$liv/titulo = "XML"
do(**insert** (<autor><nome>João Silva</nome></autor>)
following \$liv/autor[last()])
- **update** (alteração de eMail de autor)
for \$aut in input()/livro/autor
where \$aut/nome = "Maria Souza"
do (**replace** \$aut/eMail with
(<eMail>ms@new.com</eMail>))

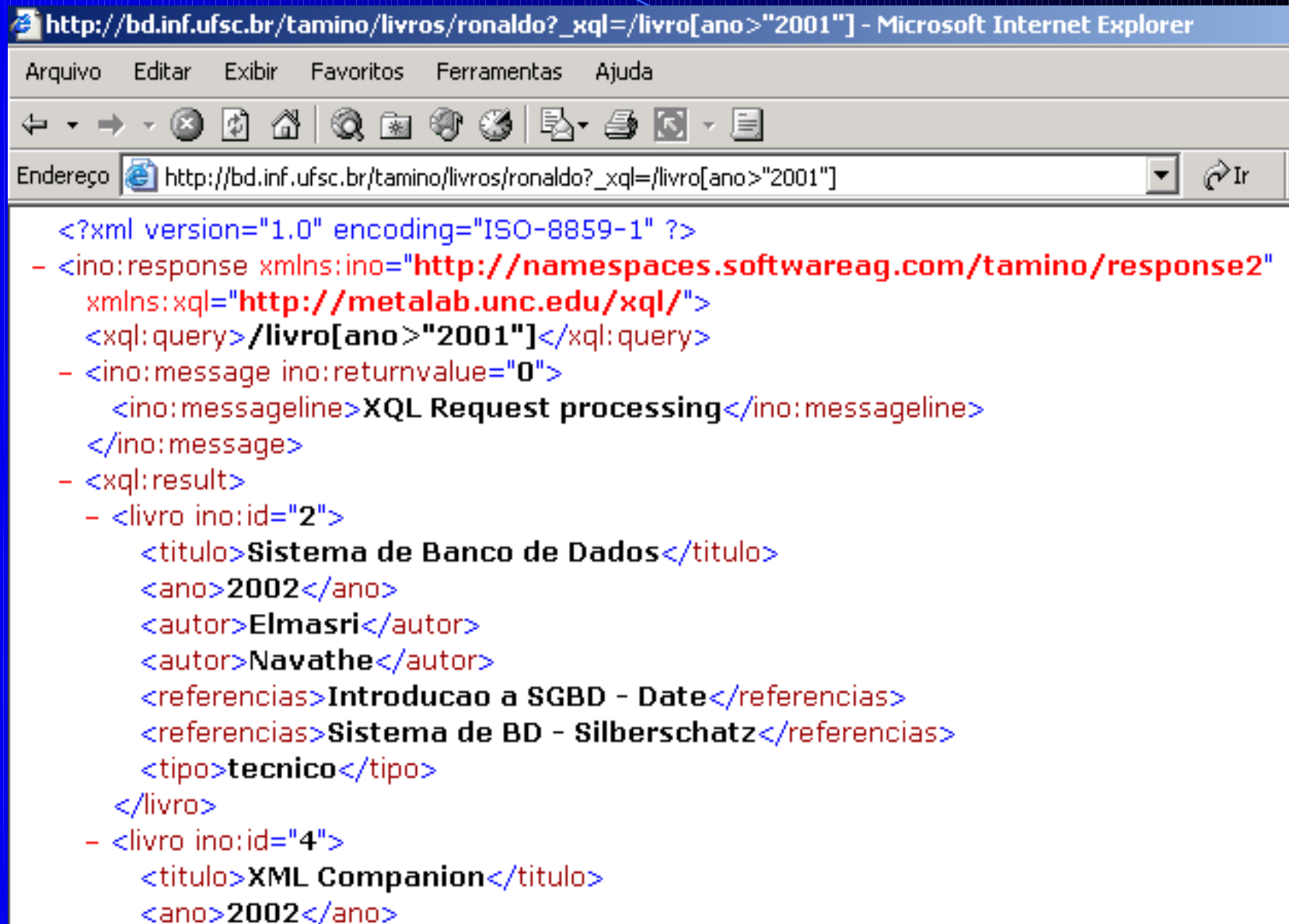
Conectividade – APIs

- XQuery API para Java (JQX)
 - JDBC como base
- Protocolos HTTP
 - acesso via *browsers Web*
- Consórcio XML:DB
 - proposta de uma API para BDs XML
 - manipulação de BDs e coleções; execução de consultas *Xpath* e *XUpdate*; acesso a resultados de consultas; controle de transações

APIs - Tamino

- Interface principal de acesso são *browsers Web*
 - um servidor Tamino deve estar sempre associado a um *Web server* (domínio *Internet*)
 - define uma API que encapsula chamadas HTTP
 - criação e manipulação de BDs, coleções e docs
 - acesso: `http://<nome_domínio>/tamino/<nome_BD>/[<nome_coleção>]<comando_API_HTTP>`
- Outras formas de acesso
 - API DOM para aplicações Java e Java Script
 - API XML:DB

Tamino – Conectividade HTTP



The screenshot shows a Microsoft Internet Explorer browser window. The address bar displays the URL: `http://bd.inf.ufsc.br/tamino/livros/ronaldo?_xql=/livro[ano>"2001"]`. The browser's menu bar includes "Arquivo", "Editar", "Exibir", "Favoritos", "Ferramentas", and "Ajuda". The toolbar contains various navigation and utility icons. The main content area displays an XML response, which is a list of books returned by a query. The XML is formatted with color-coding: red for namespace URIs and blue for element names. The response includes a root element `<?xml version="1.0" encoding="ISO-8859-1" ?>` followed by a `<ino:response>` element. Inside this, there is an `<xql:query>` element with the query `/livro[ano>"2001"]`, a `<ino:message>` element indicating "XQL Request processing", and a `<xql:result>` element containing two book records. The first record has `ino:id="2"` and lists books like "Sistema de Banco de Dados" and "Introducao a SGBD - Date". The second record has `ino:id="4"` and lists "XML Companion".

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <ino:response xmlns:ino="http://namespaces.softwareag.com/tamino/response2"
  xmlns:xql="http://metalab.unc.edu/xql/">
  <xql:query>/livro[ano>"2001"]</xql:query>
  - <ino:message ino:returnValue="0">
    <ino:messageline>XQL Request processing</ino:messageline>
  </ino:message>
  - <xql:result>
    - <livro ino:id="2">
      <titulo>Sistema de Banco de Dados</titulo>
      <ano>2002</ano>
      <autor>Elmasri</autor>
      <autor>Navathe</autor>
      <referencias>Introducao a SGBD - Date</referencias>
      <referencias>Sistema de BD - Silberschatz</referencias>
      <tipo>tecnico</tipo>
    </livro>
    - <livro ino:id="4">
      <titulo>XML Companion</titulo>
      <ano>2002</ano>
```

Armazenamento

- **Docs XML “in-natura”** (campo longo)
 - texto do doc preservado na íntegra (cabeçalho, comentários, ...)
 - armazenamento clusterizado de fragmentos do doc
- **Esquema de objetos** (DOM ou esquema similar a BDOO)

DOM	BDOO
preserva ordem de elementos	não preserva ordem de elementos
qualquer esquema tem a mesma estrutura (<i>document</i> , <i>element</i> , ...) <ul style="list-style-type: none">– intenção dos dados não fica clara	esquema de classes gerado de acordo com os tipos de elementos complexos <ul style="list-style-type: none">– intenção dos dados mais clara
clusterização por profundidade <ul style="list-style-type: none">– bom p/ buscas na ordem da hierarquia clusterização por largura <ul style="list-style-type: none">– bom p/ buscas por propriedades de um elemento	clusterização geralmente por instâncias da mesma classe <ul style="list-style-type: none">– bom para buscas por dados de determinados tipos de elementos

Projeto de um BD XML

- Não há uma metodologia consolidada
- Projeto tradicional de um BD
 - (i) especificação de requisitos; (ii) modelagem conceitual; (iii) modelagem lógica e (iv) modelagem física ou implementação
 - pode ser aplicado a um BD XML
 - no caso de dados XML **fortemente semi-estruturados**
 - **revisão da modelagem física**: considerar a existência de informação textual não-estruturada no conteúdo de elementos

Guia para Projeto de BD XML

1. Especificação de requisitos

- levantamento das necessidades de dados

2. Modelagem conceitual

- uso de um modelo de dados convencional (ex.: ER)

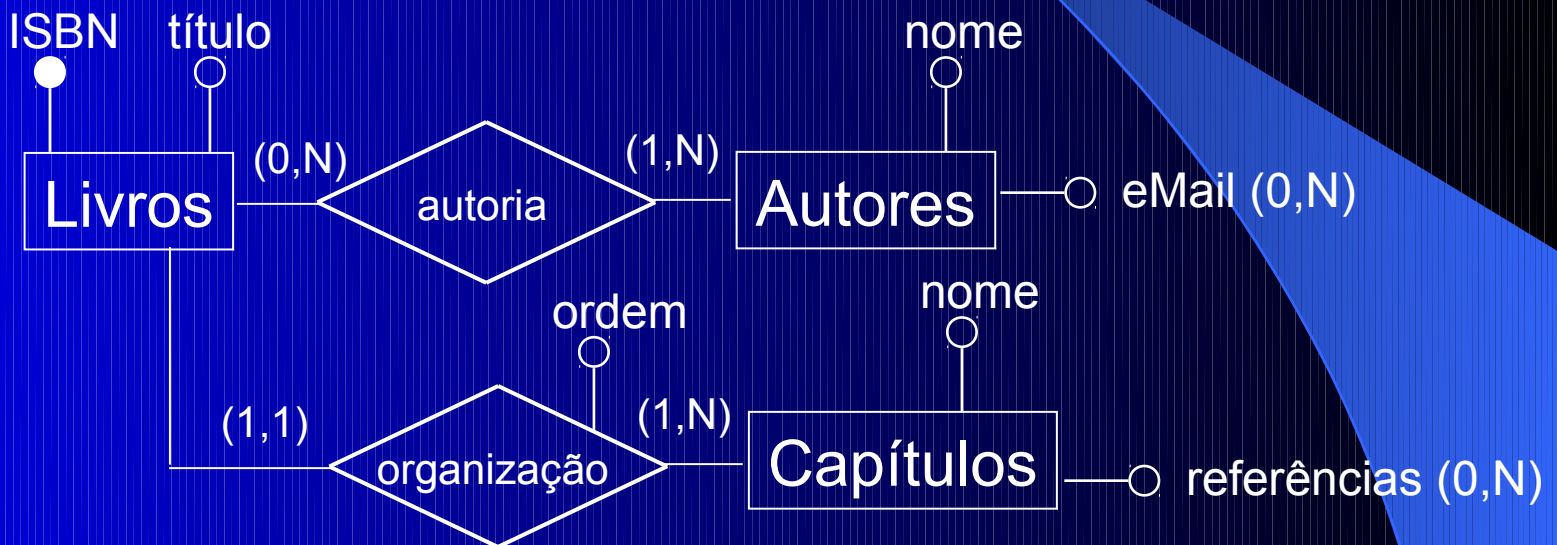
3. Modelagem lógica

- uso de um modelo de dados baseado em grafo
 - adequado à representação de uma hierarquia XML

4. Modelagem física

- especificação do esquema XML (DTD ou XSD)

Modelagem Conceitual - Exemplo



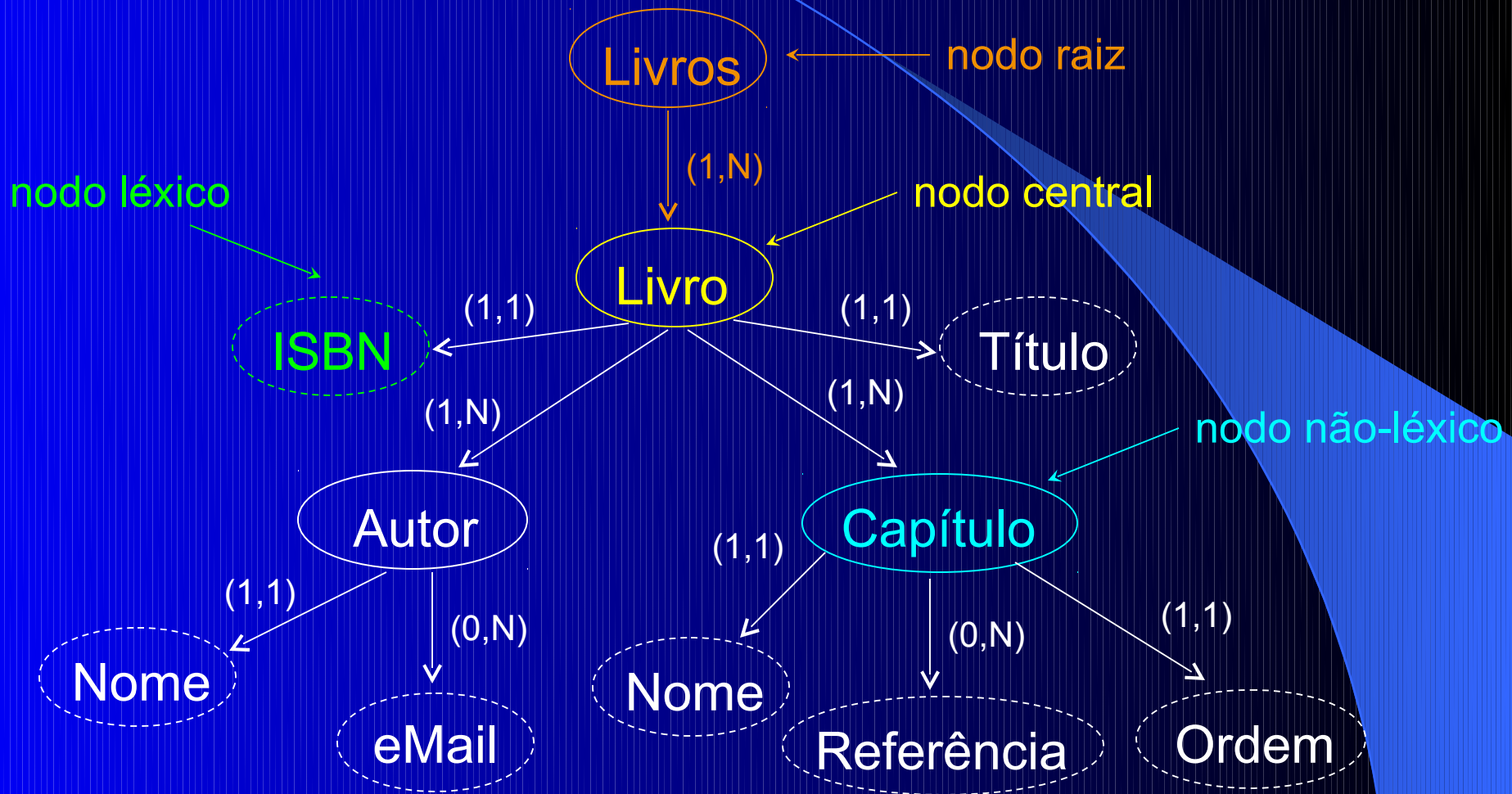
Modelagem Lógica

- Grafo orientado
- **Nodos não-terminais (ou não-léxicos)**
 - mapeamento de entidades do ER
 - modelam elementos compostos
- **Nodos terminais (ou léxicos)**
 - mapeamento de atributos do ER
 - modelam conteúdo de elemento ou de atributo
- **Arestas rotuladas com restrições de cardinalidade**
 - mapeamento de relacionamentos ou associações entidade-atributo do ER
 - modelam relacionamentos hierárquicos ou associações elemento-atributo

Modelagem Lógica

- Eleição do nodo não-léxico central
 - entidade central na modelagem conceitual
 - a partir dela uma hierarquia de nodos pode ser definida com base nos seus relacionamentos no ER
 - exemplo: **Livro**
 - um **nodo raiz** deve ser definido como pai deste nodo (cardinalidade 1:N)
 - sugestões de nomenclatura
 - conjunto de ocorrências da entidade central (ex.: **Livros**)
 - contexto do domínio (exemplos.: **Livraria**, **Biblioteca**, ...)
- Mais de um nodo central pode existir...
 - entidades “independentes” (ex.: livros e funcionários de uma biblioteca)
 - todos serão filhos do nodo raiz (ex.: biblioteca)

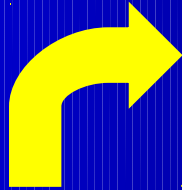
Modelagem Lógica - Exemplo



Modelagem Física

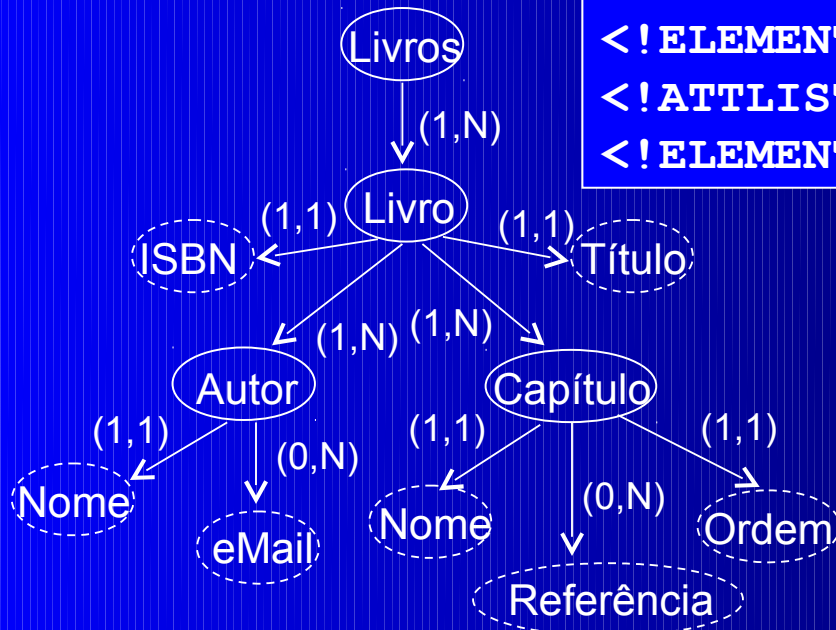
- Definição de elementos e atributos do esquema
 - **nodos não-léxicos** → **elementos compostos**
 - **nodos léxicos** → **elementos simples ou atributos**
- Definição da ordem de sub-elementos
 - posição das arestas que partem do nodo não-léxico
- Modelagem física de um nodo léxico
 - como **atributo**
 - economia de espaço no documento XML
 - pode-se definir algumas RIs específicas
 - como **elemento**
 - conteúdos extensos
 - deseja-se sub-elementos com cardinalidade > 1
 - melhor legibilidade do documento XML

Modelagem Física - Exemplo



```
<!ELEMENT Livros (Livro+)>
<!ELEMENT Livro (Título, Autor+, Capítulo+)>
<!ATTLIST Livro ISBN CDATA>
<!ELEMENT Título (#PCDATA)>
<!ELEMENT Autor (Nome, eMail*)>
<!ELEMENT Nome (#PCDATA)>
<!ELEMENT eMail (#PCDATA)>
<!ELEMENT Capítulo (Nome, Referência*)>
<!ATTLIST Capítulo ordem CDATA>
<!ELEMENT Referência (#PCDATA)>
```

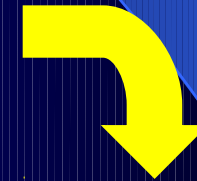
DTD



Modelagem Física - Revisão

```
<!ELEMENT Livros (Livro+)>
<!ELEMENT Livro (Título, Autor+, Capítulo+)>
<!ATTLIST Livro ISBN CDATA>
<!ELEMENT Título (#PCDATA)>
<!ELEMENT Autor (Nome, eMail*)>
<!ELEMENT Nome (#PCDATA)>
<!ELEMENT eMail (#PCDATA)>
<!ELEMENT Capítulo (Nome, Referência*)>
<!ATTLIST Capítulo ordem CDATA>
<!ELEMENT Referência (#PCDATA)>
```

capítulos com
conteúdo textual



```
<!ELEMENT Livros (Livro+)>
<!ELEMENT Livro (Título, Autor+, Capítulo+)>
<!ATTLIST Livro ISBN CDATA>
<!ELEMENT Título (#PCDATA)>
<!ELEMENT Autor (Nome, eMail*)>
<!ELEMENT Nome (#PCDATA)>
<!ELEMENT eMail (#PCDATA)>
<!ELEMENT Capítulo (Nome, Texto)>
<!ELEMENT Texto (#PCDATA | Referência)*>
<!ELEMENT Referência (#PCDATA)>
```