Bancos de Dados Semiestruturados

XML (eXtensible Markup Language)

```
<professora>
   <nome> Monica Pereira</nome>
</professora>
```

Universidade Comunitária da Região de Chapecó – Unochapecó
Escola Politécnica – Ciência da Computação/Sistemas de Informação
Bancos de Dados Semiestruturados
XML (eXtensible Markup Language)

- Características:

BD Semiestruturados
Alta heterogeneidade dos dados.
Inexistência de um esquema uniforme.
Consultas realizadas por palavras-chave ou busca exaustiva.
Representação estrutural heterogênea.
Auto descritivo.
Definição a posteriori das estruturas de dados, a partir da análise do conjunto de dados.
Estrutura irregular – dados com mesma semântica podem estar representados e organizados de forma heterogênea.
Estrutura implícita – determinada pela organização dos dados.
Distinção dos dados e estrutura não é clara.

- BD Semiestruturados

Estrutura parcial: a estrutura não precisa estar totalmente declarada.
Estrutura extensa: elementos aninhados podem ter várias extensões.
Estrutura evolucionária: a estrutura dos dados podem variar tanto quanto o conjunto de dados.
Estrutura descritiva e não prescritiva: é determinada de acordo com o conjunto de dados e não determina os dados que podem ser aceitos pela estrutura.

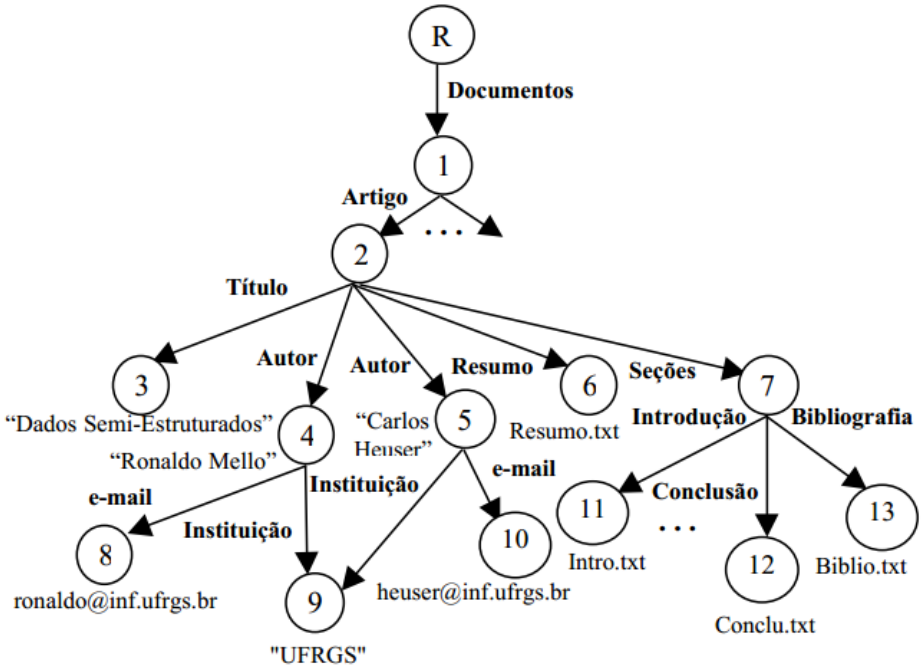
Tabela 1.1 - Diferenças entre dados tradicionais e dados semi-estruturados

Dados tradicionais	Dados semi-estruturados
Esquema predefinido	Nem sempre há um esquema predefinido
Estrutura regular	Estrutura irregular
Estrutura independente dos dados	Estrutura embutida no dado
Estrutura reduzida	Estrutura extensa
Estrutura fracamente evolutiva	Estrutura fortemente evolutiva
Estrutura prescritiva	Estrutura descritiva
Distinção entre estrutura e dado é clara	Distinção entre estrutura e dado não é clara

Fonte: Heuser et al.

- Modelagem de Dados

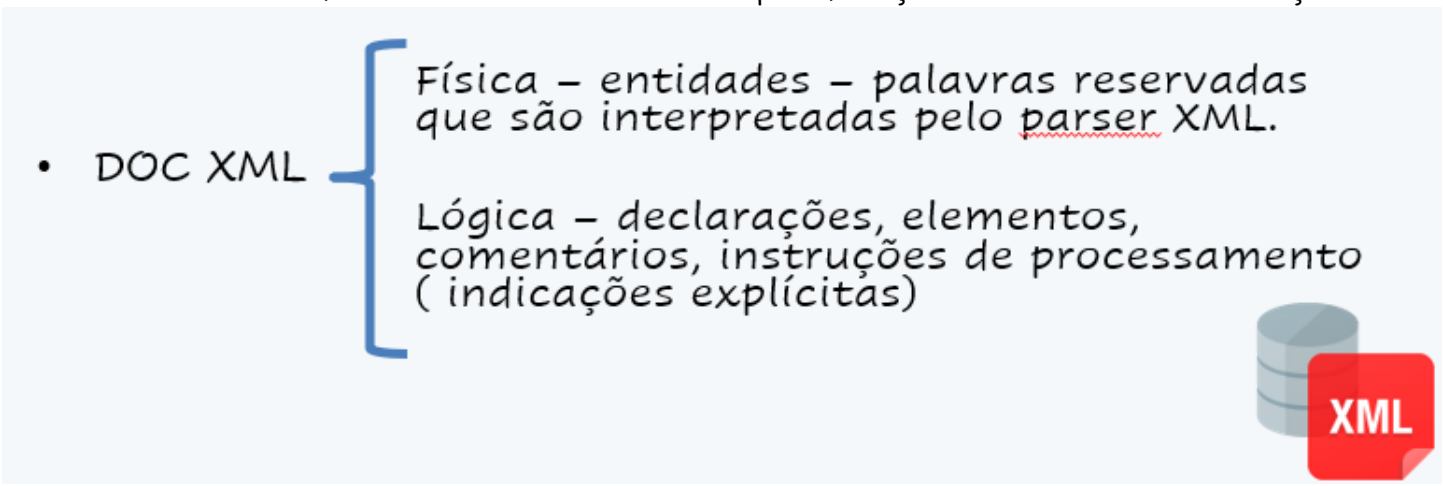
A forma mais usual de modelar dados semiestruturados é através de grafos direcionados rotulados.
Os vértices representam objetos identificáveis e as arestas são arcos para outros objetos que fazem parte da sua estrutura, que é hierárquica.
Um rótulo presente em um arco indica o nome do atributo do objeto de onde o arco parte (objeto origem), podendo ainda informar o tipo de dado do objeto alcançável através do arco (objeto destino).



Fonte: Heuser et al.

- XML

Linguagem de marcação aplicada a interoperabilidade de aplicações na WEB.
Linguagens de marcação são formadas por um conjunto de chave: valor ou nome: valor e descrevem objetos em ambientes computacionais.
Documentos XML válidos tem tags iniciais e finais.
Documentos bem formados atendem as especificações XML e suas restrições.



Elementos: descrevem objetos de qualquer tipo e podem ter um conjunto de atributos, sub elementos ou ainda ser um elemento vazio.

Elemento vazio:

```
<livro></livro>
```

Elemento:

```
<livro>
   <titulo>Iaiá Garcia</titulo>
   <autor>Machado de Assis</autor>
   <ano>1920</ano>
```

```
    <preco>1.30</preco>
  </livro>
```

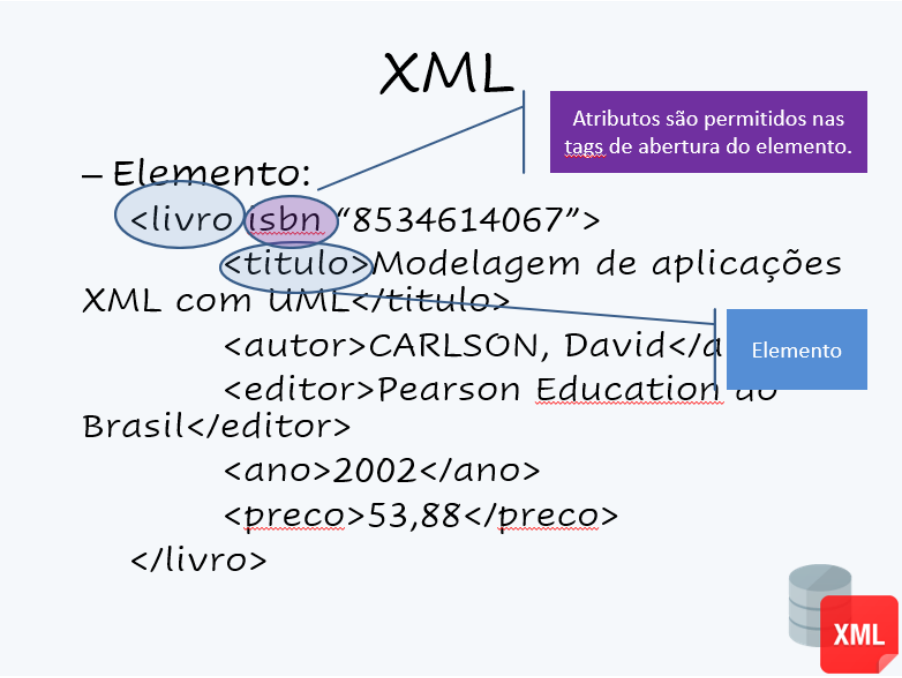
Elementos podem ser multivalorados:

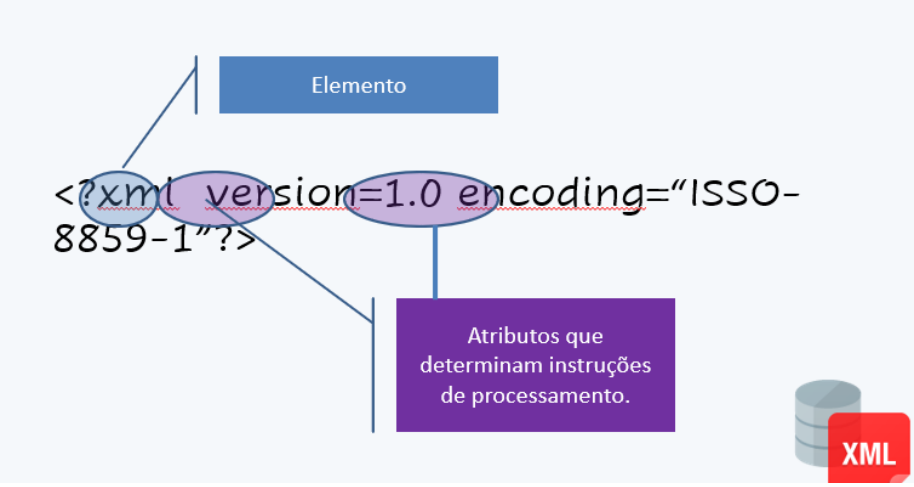
```
<livro>
  <titulo>XQuery Kick Start</titulo>
  <autor>James McGovern</autor>
  <autor>Per Bothner</autor>
  <autor>Kurt Cagle</autor>
  <autor>James Linn</autor>
  <autor>Vaidyanathan Nagarajan</autor>
  <ano>2003</ano>
  <preco>49.99</preco>
</livro>
```

Elementos podem conter atributos:

```
<livro isbn = 978-0596004200 categoria="WEB">
  <titulo lang="en">Learning XML</titulo>
  <autor>Erik T. Ray</autor>
  <ano>2003</ano>
  <preco>39.95</preco>
  <data>'01/01/2015'</data>
</livro>
```

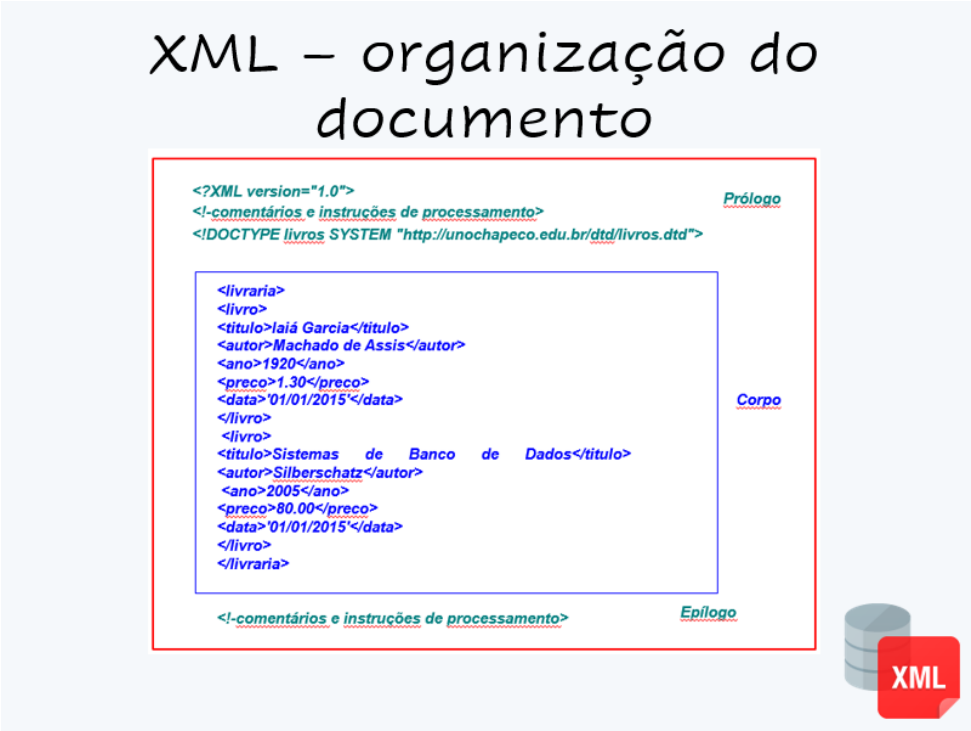
Composição de elementos XML:





Atributos:
Atributos: propriedades dos elementos.
Atributos não podem ser multivalorados, elementos sim.
Atributos não podem ser estruturados, elementos podem sempre ser estendidos, com elementos pai e filhos.
Validação de documentos XML pode ser feita através de arquivos DTD ou XML Schema.

- **Organização de um DOC XML:**
  - Prólogo
    - Instruções de processamento e declarações.
  - Corpo
    - Dados de usuários – elementos (chave: valor)
  - Epílogo
    - Comentários



- Regras para um DOC XML:
  - Prólogo
    - Pode conter as instruções:
    - Version – versão da linguagem XML
    - Encoding – codificação para caracteres
    - Standalone – indica se o documento está associado ou não a folha de estilos.

  - **Toda árvore XML deve iniciar com o elemento raiz e ter o seu fechamento declarado.**

    Pode conter os atributos:

    - xml:lang – pode ser vinculado a qualquer elemento

    <cumprimento xml:lang="en">Hello</cumprimento>
    < cumprimento xml:lang="pt-br">Olá</cumprimento>
    < cumprimento xml:lang="fr">Bonjour</cumprimento>

    - xml:space – relativo aos espaços em branco do documento. Pode ter os valores "preserve" ou "default".

  Comentários: podem ser inseridos em qualquer parte do documento e tem o formato:

  <!-- Comentário -->

  - Instruções de processamento: fornecem indicações para os aplicativos que acessam as fontes de dados XML, de como processar estes dados.
    - <**?**xml version=1.0 encoding="ISSO-8859-1"**?**>
  - Referências de entidades: estas entidades precisam estar codificadas para a interpretação do parser XML. Entidades equivalem a palavras reservadas.

Documentos bem formados, conforme W3C:

Qualquer texto é um documento XML bem formado se:

Tomado como um todo, ele corresponde a um documento rotulado, marcado.

Ele atende todas as restrições de boa formação dadas pela especificação do W3C em:
http://www.w3.org/TR/2006/REC-xml11-20060816/#sec-well-formed

Cada uma das entidades, quando analisada, direta ou indiretamente referenciada no documento é bem formada.

Tipos de documentos XML:
Inválidos: documentos que não seguem as regras de especificação XML, conforme o W3C;

Válidos: documentos que seguem a especificação XML e também as determinações no seu DTD ou XMLSchema;
Documentos bem formados: obedecem as regras de especificação mas não tem uma definição explícita de esquema, via DTD ou XMLSchema.

XML – estrutura em árvore
W3C Document Object Model (DOM)
XML Schema Definition
Define a estrutura de uma arquivo XML, definindo se o arquivo é um arquivo válido ou não.

- <?xml version="1.0"?>

  <xs:schema>
  ...
  ...
  </xs:schema>

Para verificar a validade de um arquivo xml de acordo com um arquivo xsd, acesse:
http://www.utilities-online.info/xsdvalidation/

- Manipulando arquivos XML

Acesse ou baixe o gerenciador de bases XML BaseX.
Execute o arquivo BaseX.jar
Crie um novo banco de dados a partir de um diretório com arquivos XML.
Para habilitar as alterações no banco XML, execute o comando conforme o exemplo:

- **XML – XPath**
  Determina como navegar na árvore XML.

Consultas baseadas na estrutura da árvore XML:
Retorna o primeiro elemento da árvore XML
doc("livros.xml")/livraria/livro[1]

doc("pessoas.xml")//dados/pessoa[1]

Retorna o último elemento da árvore XML
doc("livros.xml")/livraria/livro[last()]

Consultas baseadas na estrutura da árvore XML:
Retorna o penúltimo elemento da árvore XML
doc("pessoas.xml")//dados/pessoa[last()-1]

Retorna os nomes das pessoas com idade maior que 20 anos
doc("pessoas.xml")/dados/pessoa[idade>20]/nome

XML – XPath

Funções

https://www.w3.org/2005/xpath-functions/

XML – XQuery

XQuery estende XPath

Operadores relacionais:
=, !=, <, <=, >, >=

Comparação de valores:

eq = equal
ne = not equal
lt = less then (menor que)
le = less equal (menor ou igual)
gt = greater then (maior que)
ge = greater equal (maior ou igual)

Para inserir um novo nodo no arquivo xml execute:

```
insert node
<livro>
   <codigo>0</codigo>
   <titulo>O último reino</titulo>
   <autor>Bernard Cornwell</autor>
   <ano>2004</ano>
   <preco>38.61</preco>
 </livro>
as first into doc("livros.xml")/livraria
```

Para inserir vários elementos no documento

```
for $n in doc("livros.xml")/livraria/livro
      return insert nodes <data>'01/01/2015'</data> into $n

for $n in doc("livros.xml")/livraria/livro
      return insert nodes <custo>5.0</custo> into $n
```

Para inserir vários elementos no documento

```
let $date := current-date() return $date

for $n in doc("livros.xml")/livraria/livro
      return insert nodes <data>{$date}</data>  into $n
```

Para atualizar valores de nodos filhos

```
for $n in doc("livros.xml")/livraria/livro[codigo=2]
       return replace value of node $n/preco with 56.80

for $n in doc("livros.xml")/livraria/livro/preco return
       replace value of node $n with $n*1.1
```

Eliminando nodos

```
for $n in doc("livros.xml")/livraria/livro
       return delete node$n/custo

for $n in doc("livros.xml")/livraria/livro
       where $n/preco < 50.00
       return delete node$n/ano

for $n in doc("livros.xml")/livraria
return delete node$n/livro
```

Renomeando nodos

```
for $n in doc("livros.xml")/livraria/livro/custo
       return rename node $n as ‘preco_custo'

for $x in doc("livros.xml")/livraria/livro
      where $x/preco > 40
      order by $x/preco
      return $x/preco

for $x in doc("livros.xml")/livraria/livro
      where $x/ano eq '2005'
      return $x/ano

for $l in doc("livros.xml")/livraria/livro
       return <dados>{$l/titulo,$l/autor}</dados>
```

Comparando strings

```
for $l in doc("livros.xml")/livraria/livro
       return <dados>{$l/autor gt 'J%',$l/autor}</dados>

for $l in doc("livros.xml")/livraria/livro
       return <dados>{$l/autor eq       'Silberschatz',$l/autor}</dados>
```

Concatenando valores

```
for $l in doc("livros.xml")/livraria/livro
      return concat($l/titulo, " ", $l/preco)
```

Exemplos de consultas

```
doc("livros.xml")/livraria/livro[autor="Tannenbal"]

doc("livros.xml")/livraria/livro[preco>40 and preco<100]

for $x in doc("livros.xml")/livraria/livro
      return $x/autor


for $x in doc("livros.xml")/livraria/livro
     return $x/autor | $x/titulo


for $x in doc("livros.xml")/livraria/livro
      where $x/preco > 100.00
      return $x/autor | $x/preco

for $x in doc("livros.xml")/livraria/livro
      where $x/autor = 'Tannenbal'
      return $x/autor | $x/preco

for  $x in doc("livros.xml")/livraria/livro/titulo
      return if (contains($x,'XQuery')) then $x/text()   else 'diferente'

for $x in doc("livros.xml")/livraria/livro
      where $x/preco < 100.00
      order by $x/preco
      return ($x/titulo, $x/autor, $x/preco)
```

Funções de agregação

Xpath

```
sum(doc("livros.xml")/livraria/livro/preco)
sum(doc("livro.xml")/livraria/livro[ano>2005]/preco)
avg(//livraria/livro/preco)
```

Xquery

```
for $x in doc("livros.xml")/livraria/livro/preco
where $x > avg(//livraria/livro/preco) return $x
```

Funções de data e hora

```
      current-date()
```

```
current-dateTime()
year-from-date(xs:date('2015-07-19'))
year-from-date(xs:date(current-date()))
```

Retornando o ano da data corrente:

```
let $ano := year-from-date(xs:date(current-date())) return $ano

return
<results>
  <ano>{$ano}</ano>
</results>
```

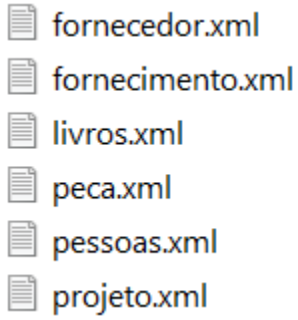
XSD
Validação de arquivos
http://www.utilities-online.info/xsdvalidation/

Para gerar o arquivo xsd e validar xml
https://www.freeformatter.com/xml-validator-xsd.html

Para gerar o arquivo xsd
https://www.liquid-technologies.com/online-xml-to-xsd-converter

XML
Links para pesquisa:
http://www.w3.org/standards/xml/core.html
http://www.w3schools.com/xml/
http://www.utilities-online.info/xsdvalidation/
https://msdn.microsoft.com/en-us/data/bb190600.aspx

Exercícios:
Baixe do material de apoio os arquivos XML para esta atividade.
Crie bancos de dados utilizando o software BaseX que também está disponível no material de apoio. É um único arquivo em formato jar.
Os arquivos para teste são:



1. **Etapa 1: A partir dos arquivos anexados, gere instruções Xquery para:**

a) Retornar os dados da penúltima peça da árvore XML.
b) Inserir um atributo com a data em todos os fornecimentos, a data deve ser inserida no formado YYYY-MM-DD.
c) Atualizar o status dos fornecedores de Londres para 50.
d) Retornar o código, a cidade e cor de todas as peças.
e) Obter o somatório das quantidades dos fornecimentos.
f) Obter os nomes dos projetos de Paris.
g) Obter o código dos fornecedores que forneceram pecas em maior quantidade.
h) Excluir os projetos da cidade de Atenas.
i) Obter os nomes das peças e seus dados de fornecimento.
j) Obter o preço médio das peças.

2. **Etapa 2: No PostgreSQL gere o script da base de dados com as tabelas Peca, Fornecedor, Projeto e Fornecimento que está no material de apoio.**
   a. Crie uma aplicação em qualquer linguagem de programação que integre os dados da tabela Peca do PostgreSQL com os dados semiestruturados do arquivo Fornecimento.xml, de forma a fazer uma junção entre as duas fontes de dados.
   b. Faça upload dos dois arquivos, das etapas1 e 2 no repositório do GitHub que está compartilhado com a professora.