

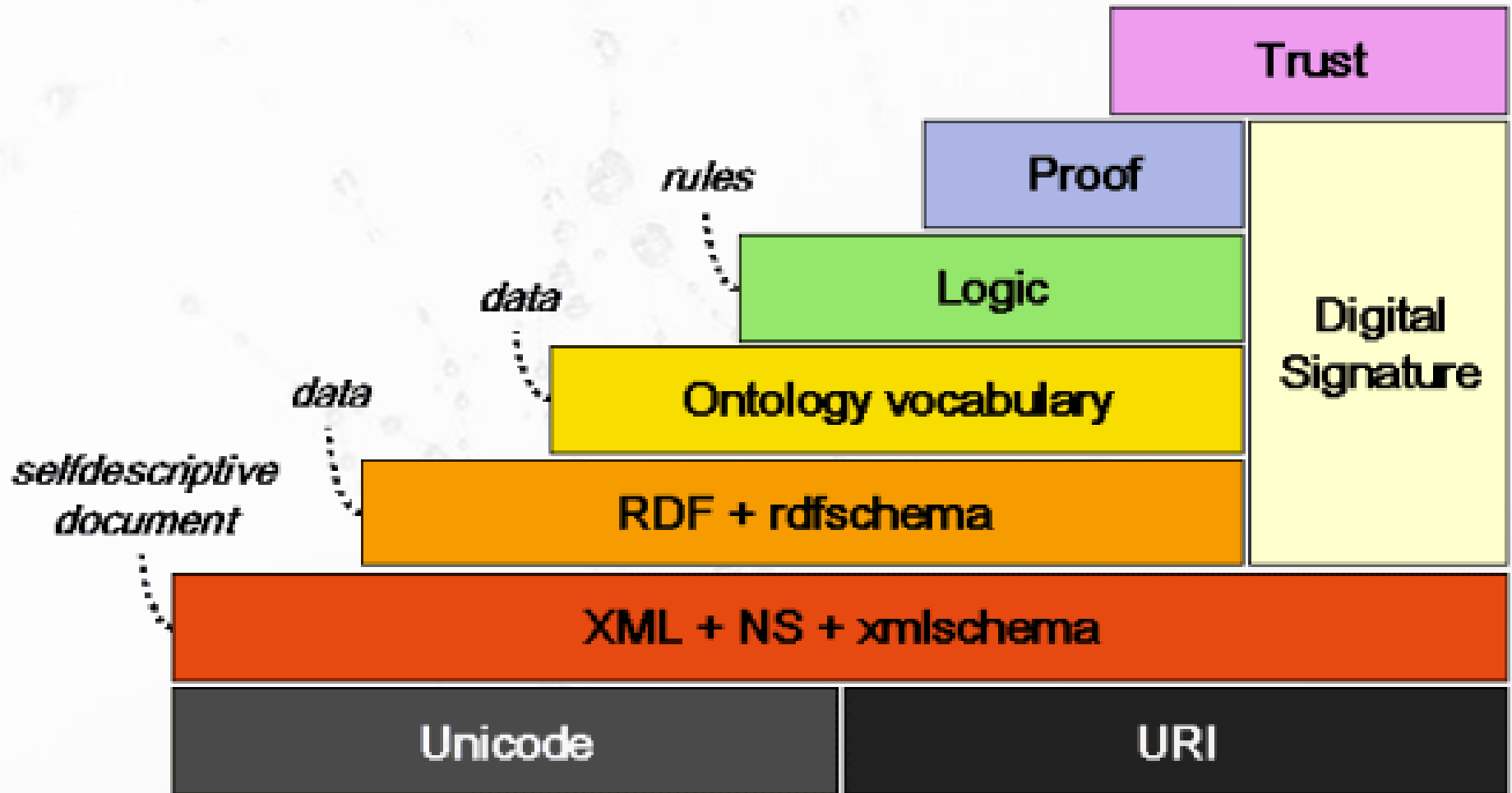
# Banco de Dados na Web

## Banco de Dados: Teoria e Prática

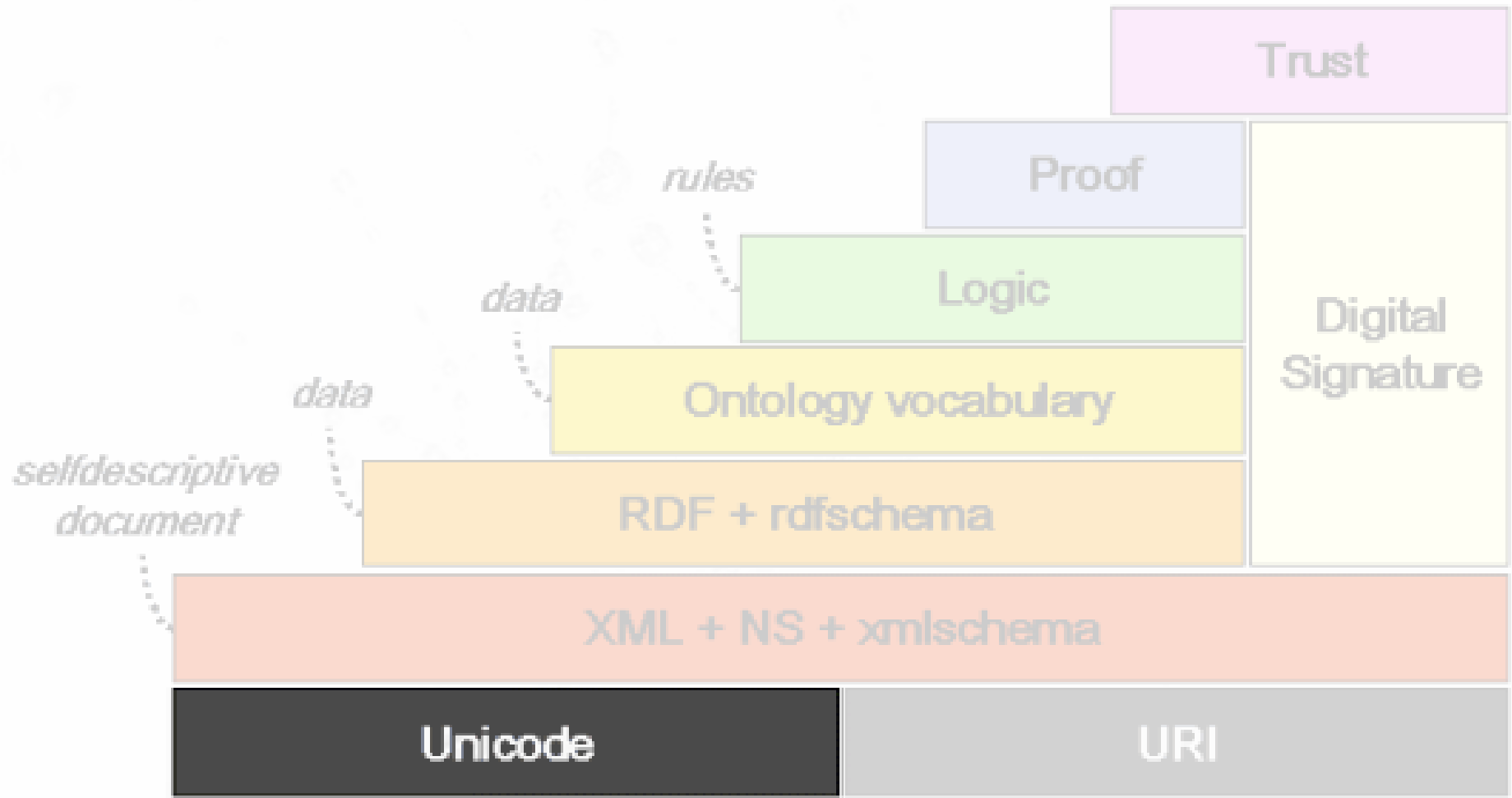
André Santanchè e Luiz Celso Gomes Jr  
Institute of Computing - UNICAMP  
Agosto 2013

# Web Semântica

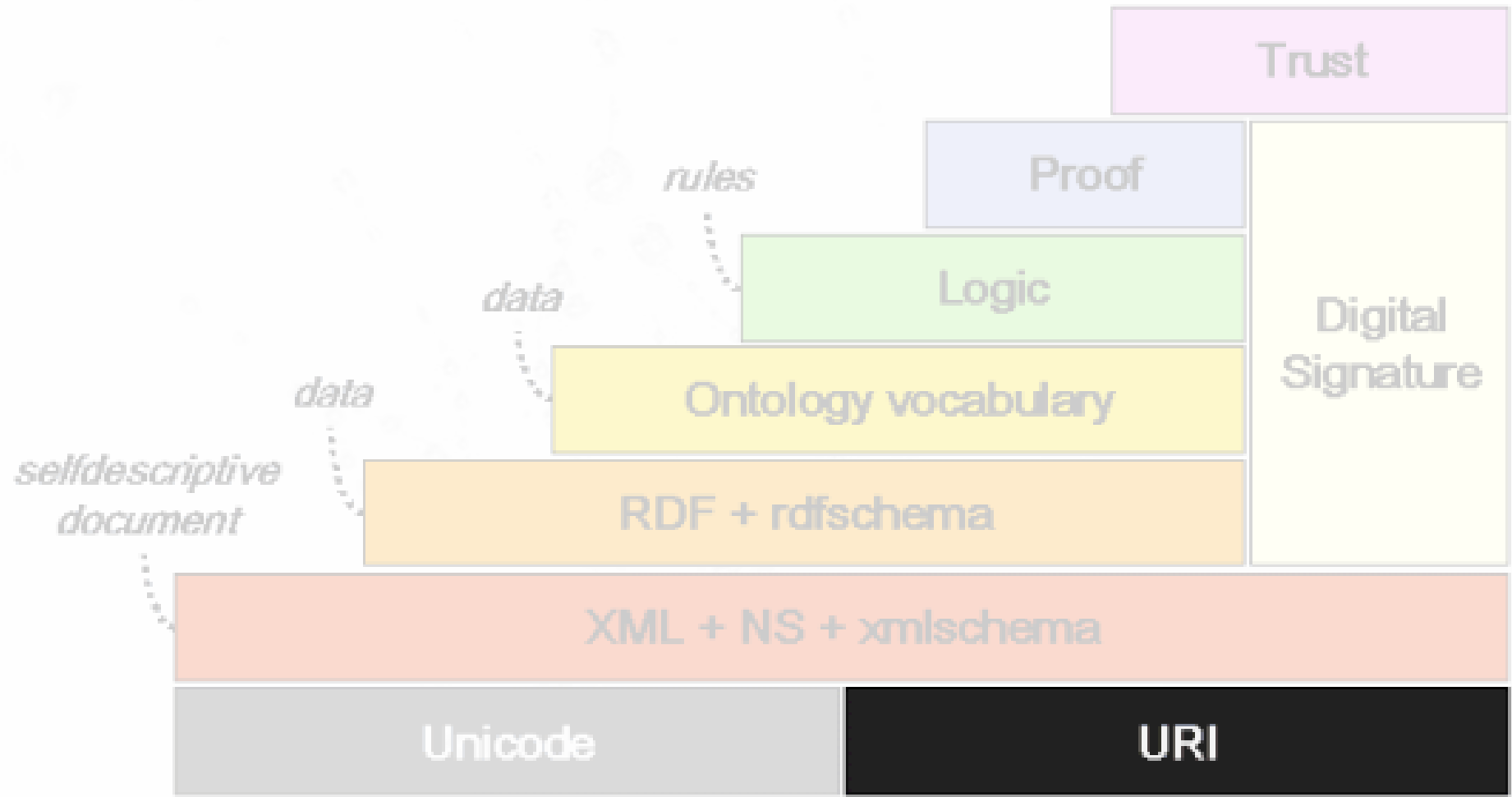
# Semantic Web



# Web Semântica



# Web Semântica





# URI

- A identificação de um recurso é feita através de um URI - Uniform Resource Identifier.
- URI = URL ou URN



- URL (*Uniform Resource Locator*): identifica recursos por meio de sua localização física na Internet.

Ex.: <http://www.paleo.org>

<ftp://ftp.unicamp.br>

<mailto:horacio@paleo.org>

- URN (*Uniform Resource Names*): identificador é relacionado indiretamente com sua localização física na rede (exige um resolver).

Ex.: <urn:ogc:def:uom:celsius>

<urn:mpeg:mpeg21:dii:iswc:T-041.220.506-1>

# Usando uma URN OGC

## ■ Como definir uma unidade Celsius?

`urn:ogc:def:uom:celsius`

1. É uma definição OGC

2. É uma unidade de medida (baseado na tabela)

3. Temperatura Celsius

<b>crs:</b>	coordinate reference systems
<b>datum:</b>	datums
<b>meridian:</b>	prime meridians
<b>ellipsoid:</b>	ellipsoids
<b>cs:</b>	coordinate systems
<b>axis:</b>	coordinate system axes
<b>coordinateOperation:</b>	coordinate operations
<b>method:</b>	operation methods
<b>parameter:</b>	operation parameters
<b>group:</b>	operation parameter groups
<b>derivedCRSType:</b>	derived CRS type codes
<b>verticalDatumType:</b>	vertical datum type codes
<b>pixelInCell:</b>	PixelInCell codes
<b>rangeMeaning:</b>	meaning codes
<b>axisDirection:</b>	axis direction codes
<b>uom:</b>	units of measure



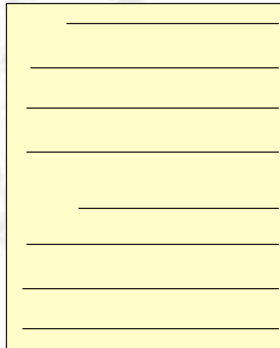
# Combinação de URL e URN

- URL Persistente:
  - Tal como URN: Identificador relacionado indiretamente ao endereço real
  - Tal como URL: Sob a forma de URL indica o resolver

Ex.: <http://purl.org/dc/elements/1.1/>  
<http://doi.acm.org/10.1145/274440.274441>

# Possibilidades da URI

- Fazer referência a um recurso.



<http://www.paleo.org/dinos.html>

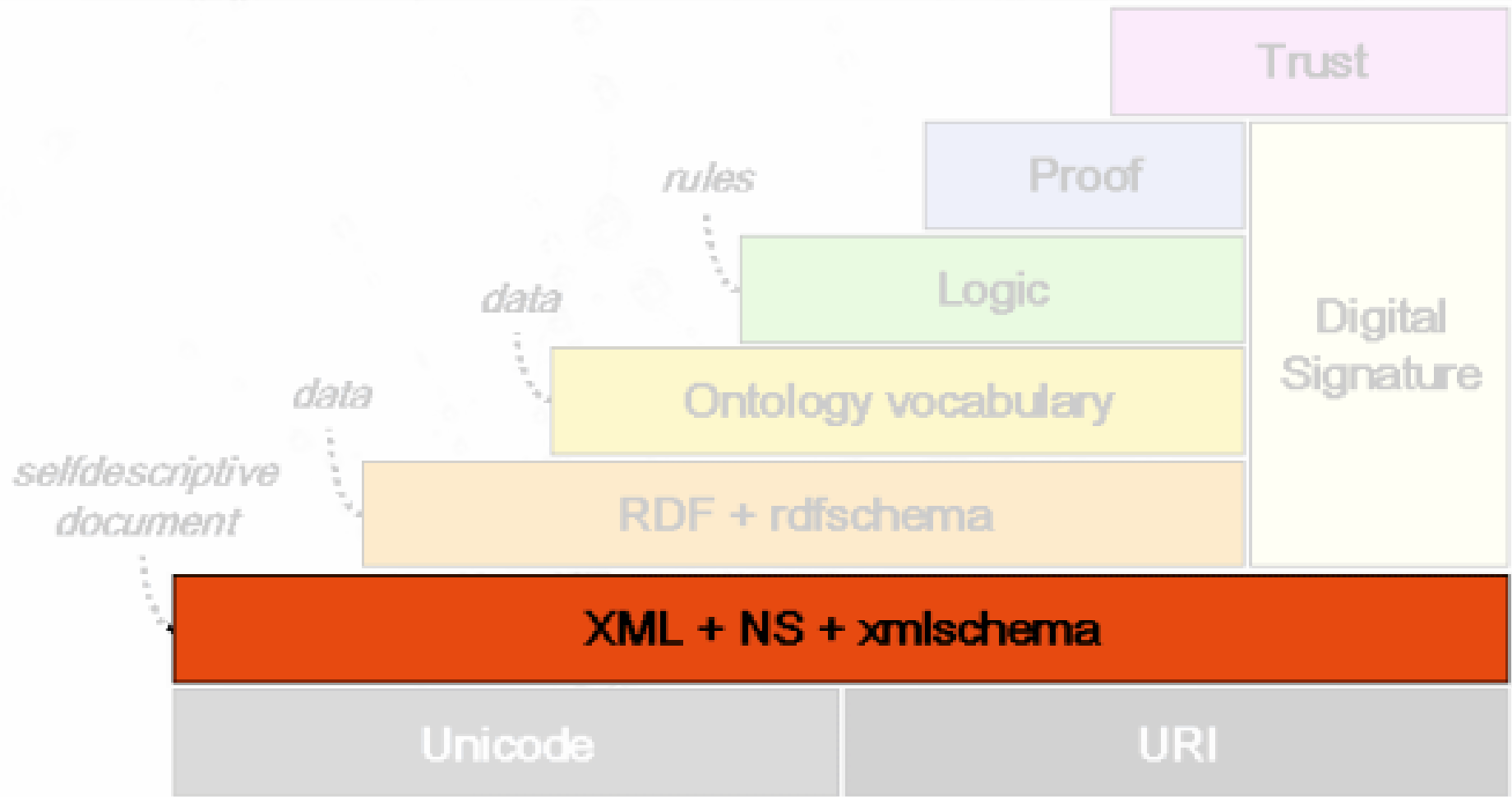
# Possibilidades da URI

- Fazer referência a um elemento dentro de um documento.

```
<__>  
  <__>  
    <__>.....</__>  
    <__ id="elem">  
      ...  
    </__>  
  <__/>  
</__>  
<__> ..... </__>  
</__>
```

<http://www.paleo.org/doc.xml#elem>

# Web Semântica



# Namespaces

- **URIs Dublin Core**

<http://purl.org/dc/elements/1.1/creator>

<http://purl.org/dc/elements/1.1/title>

<http://purl.org/dc/elements/1.1/publisher>

- **URIs vCard**

<http://nwalsh.com/rdf/vCard#Name>

<http://nwalsh.com/rdf/vCard#Address>

<http://nwalsh.com/rdf/vCard#Organization>

# Namespaces

- Demarca vocabulários
- Aumenta a legibilidade

# Namespaces

- URIs Dublin Core

dc: ⇒ <http://purl.org/dc/elements/1.1/>

<http://purl.org/dc/elements/1.1/creator>

<http://purl.org/dc/elements/1.1/title>

<http://purl.org/dc/elements/1.1/publisher>

dc:creator

dc:title

dc:publisher

# Namespaces

- URIs vCard

vcard: ⇒ <http://nwalsh.com/rdf/vCard#>

<http://nwalsh.com/rdf/vCard#Name>

<http://nwalsh.com/rdf/vCard#Address>

<http://nwalsh.com/rdf/vCard#Organization>

vcard:Name

vcard:Address

vcard:Organization



# Namespaces

- Qualifica nomes de elementos e/ou atributos, conforme o vocabulário a que pertencem.

```
<ger:sentença xmlns:publ='http://www.publicar.org/esquema/'
               xmlns:ger='http://www.gerais.org/vocab#'>
  <publ:autor cpf="487.526.548-74"> Horácio </publ:autor>
  <ger:ação> escreveu o
    <publ:publicação>
      <tipo> livro </tipo>
      <título> Vida dos Dinossauros </título>
    </publ:publicação>
  </ger:ação>
</ger:sentença>
```

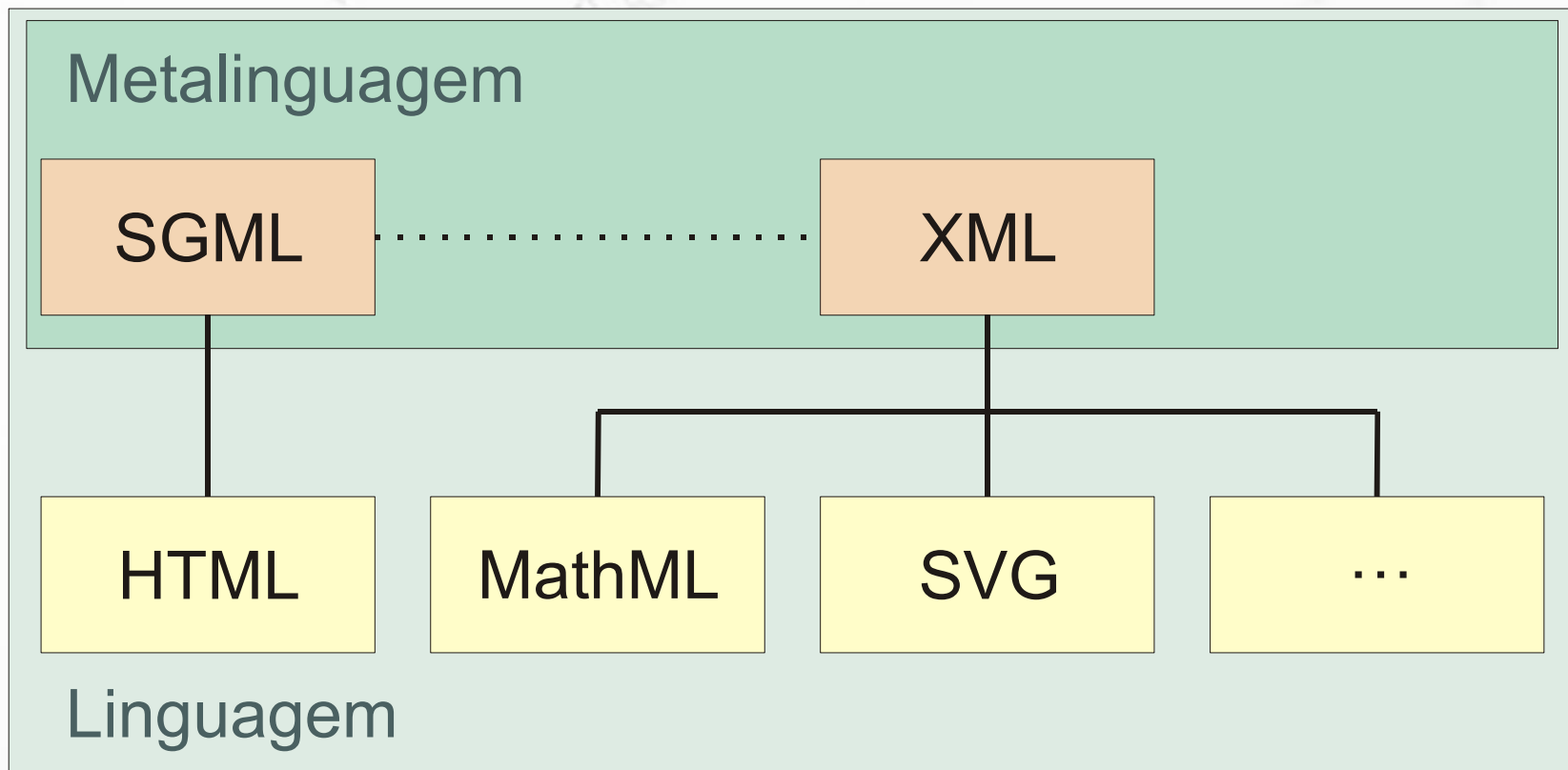
# XML - eXtensible Markup Language

# XML

- Lançada em 1996 como uma versão simplificada da SGML (*Standard Generalized Markup Language*), para ser utilizada na *Web*.

# Metalinguagem

- Tal como SGML, XML é uma metalinguagem.
- HTML ao contrário, foi escrita em SGML.



# Linguagem de Marcação

- Utiliza marcadores para agregar informações adicionais a documentos.
- Tomemos como exemplo a seguinte frase:  
Horácio escreveu o livro Vida dos Dinossauros.
- Desejamos agregar informações que identifiquem quem é o autor e qual a ação realizada.

# Linguagem de Marcação

- Os marcadores se diferenciam do conteúdo pelos símbolos “<” e “>” (seguem o mesmo princípio de HTML):

```
<autor>Horácio</autor> <ação>escreveu o livro Vida dos Dinossauros</ação>
```

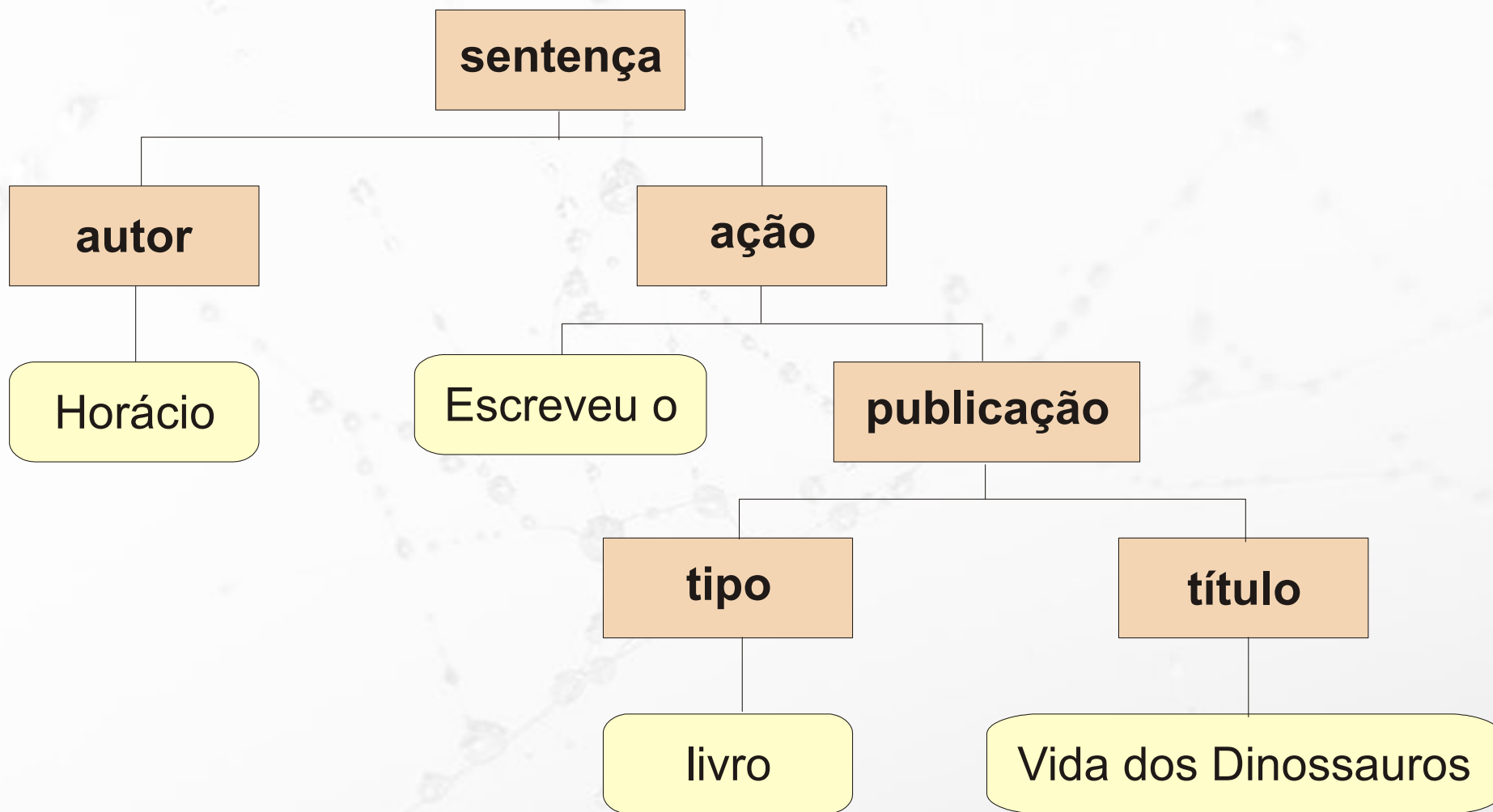
- Os marcadores delimitam unidades estruturais denominadas **elementos**.

# Estrutura Hierárquica

- Marcações podem ser agrupadas hierarquicamente.
- A interpretação de cada marcador está subordinada a seu contexto.

```
<sentença>  
  <autor>Horácio</autor>  
  <ação>escreveu o  
    <publicação>  
      <tipo>livro</tipo>  
      <título>Vida dos Dinossauros</título>  
    </publicação>  
  </ação>  
</sentença>
```

# Modelo de Dados XML





# Elementos e Atributos

- **Atributos:**

```
<autor cpf="487.526.548-74" nascimento="12/5/1960"> Horácio </autor>
```

- **Elementos vazios:**

```
<esgotado/>
```

- *Links* para elementos (#):

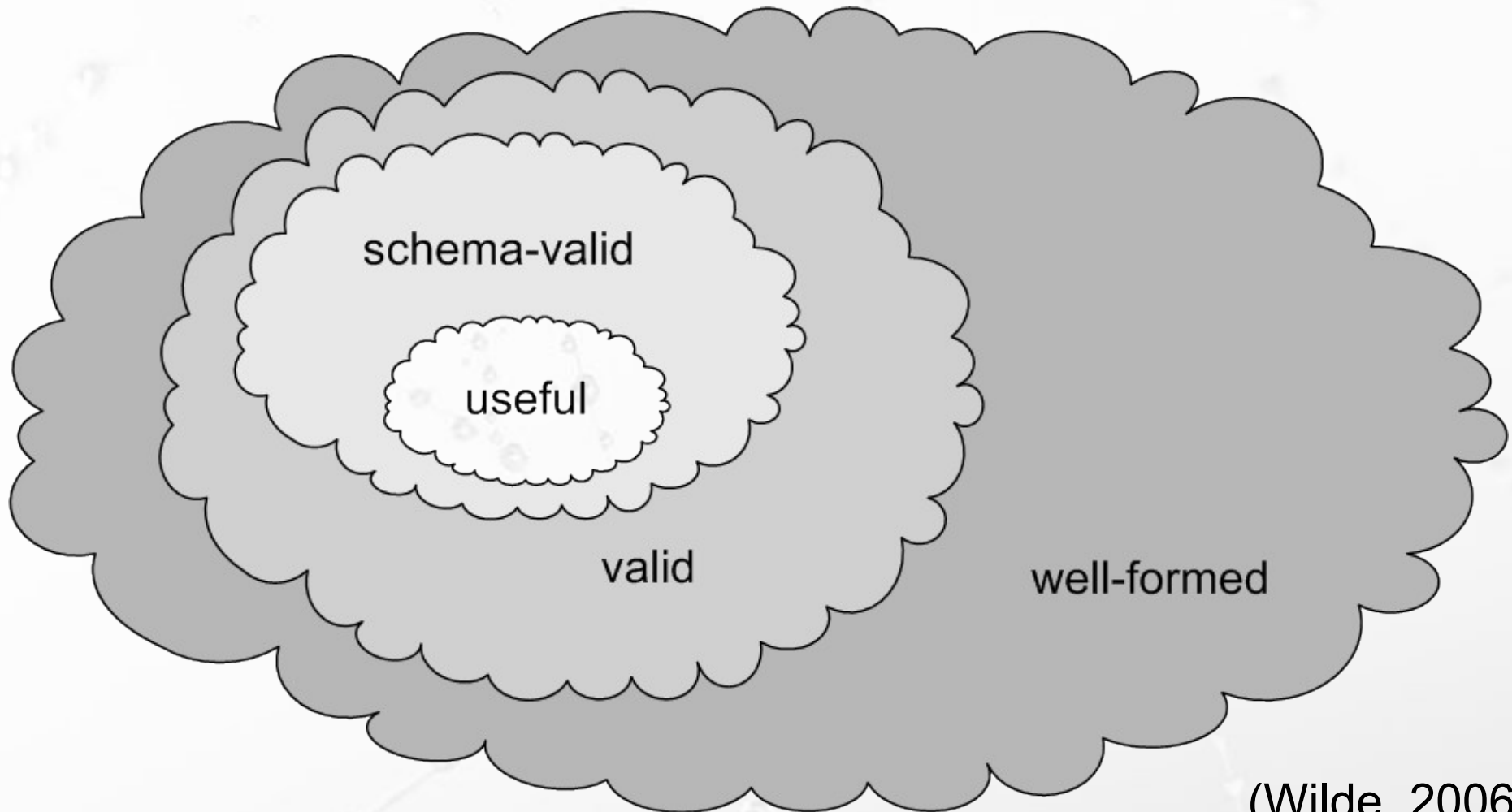
```
http://www.dominio.org/documento.html#bibliografia
```

- HTML usa esta estratégia em links para fragmentos.

# Validação de Documentos

- Documento bem formado:
  - atende às regras de construção XML
- Documento válido:
  - bem formado
  - atende a um esquema
    - DTD
    - XML Schema

# Validação de Documentos



(Wilde, 2006)

# DTD

- O documento XML pode se basear em uma gramática definida através de uma DTD (*Document Type Definition*).

```
<!ELEMENT documento (topico+)>  
<!ELEMENT topico (titulo, subtopico*)>  
<!ELEMENT titulo (#PCDATA)>  
<!ELEMENT subtopico (titulo, #PCDATA)>
```

# XML Schema

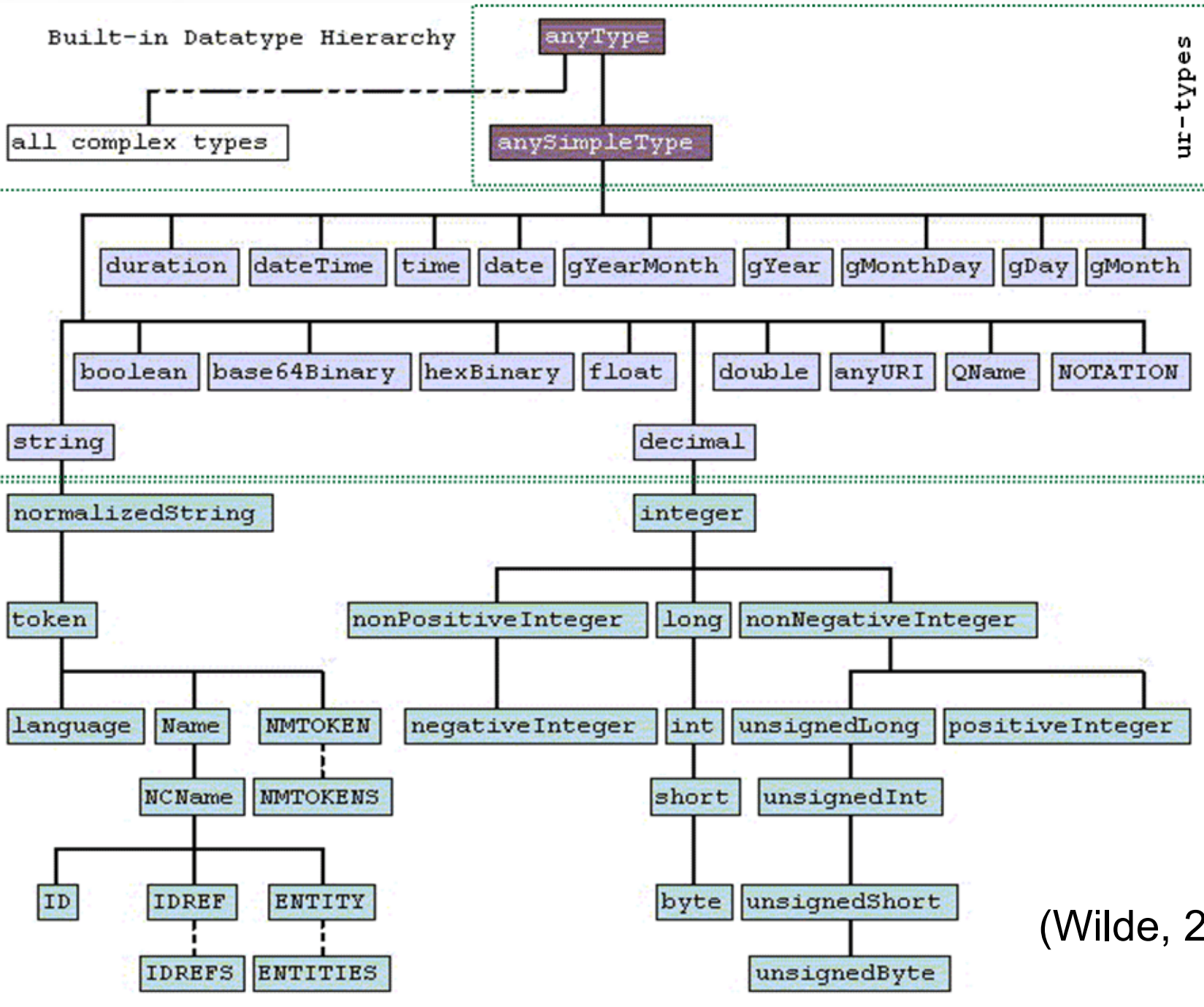
- Padrão para definição de esquemas XML
- Mais poderoso

# Tipos Simples

```
<xs:element name="business">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:maxLength value="30"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>
```

(Wilde, 2006)

Built-in Datatype Hierarchy



ur-types

primitive types

derived types

— derived by restriction

- - - derived by list

- · - · - derived by extension or

ur types

built-in primitive types

built-in derived types

(Wilde, 2006)



# Tipo Composto

```
<xs:schema>  
  <xs:element name="billingAddress" type="addressType"/>  
  <xs:element name="shippingAddress" type="addressType"/>  
  <xs:complexType name="addressType">  
    <xs:sequence>  
      <xs:element name="name" type="xs:string"/>  
      <xs:element name="street" type="xs:string"/>  
      <xs:element name="city" type="xs:string"/>  
      <xs:element name="state" type="xs:string" minOccurs="0"/>  
      <xs:element name="zip" type="xs:decimal"/>  
    </xs:sequence>  
    <xs:attribute name="country" type="xs:NMTOKEN"/>  
  </xs:complexType>  
</xs:schema>
```

(Wilde, 2006)



# Exercício 1

- Escreva uma sentença SQL para criar uma tabela cujo esquema seja compatível com este exemplo:

```
<fichario>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</fichario>
```

# Query

- **XPath**

- Especifica expressões na forma de caminhos que atendem padrões para alcançar nós específicos (elementos ou atributos)

- **XQuery**

- Queries para XML (usam XPath)

# XPath

---

**Figure 12.6**

Some examples of XPath expressions on XML documents that follow the XML schema file *company* in Figure 12.5.

1. `/company`
2. `/company/department`
3. `//employee [employeeSalary gt 70000]/employeeName`
4. `/company/employee [employeeSalary gt 70000]/employeeName`
5. `/company/project/projectWorker [hours ge 20.0]`

(Elmasri, 2011)

# XPath

/	no começo → nó raiz entre nós → separador hierárquico
//	precede nó em qualquer nível hierárquico
@	atributo
*	qualquer elemento

# Xpath Exemplos

<http://www.online-toolz.com/tools/xpath-editor.php>

```
<fichario>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</fichario>
```

# Xpath

## Exemplos

/fichario/individuo

```
<fichario>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</fichario>
```

# Xpath

## Exemplos

/fichario/individuo

```
<fichario>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</fichario>
```

```
<individuo nome="Asdrubal da Silva">
<idade>15</idade>
<genero>masculino</genero>
</individuo>
```

```
-----
<individuo nome="Quincas Borba">
<idade>33</idade>
<genero>masculino</genero>
</individuo>
```

```
-----
<individuo nome="Doriana Margarina">
<idade>42</idade>
<genero>feminino</genero>
</individuo>
```

# Xpath

## Exemplos

```
//individuo
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```



# Xpath

## Exemplos

//individuo

```
<fichario>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</fichario>
```

```
<individuo nome="Asdrubal da Silva">
<idade>15</idade>
<genero>masculino</genero>
</individuo>
-----
<individuo nome="Quincas Borba">
<idade>33</idade>
<genero>masculino</genero>
</individuo>
-----
<individuo nome="Doriana Margarina">
<idade>42</idade>
<genero>feminino</genero>
</individuo>
```

# Xpath

## Exemplos

```
//individuo/@nome
```

```
<fichario>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</fichario>
```

# Xpath Exemplos

```
//individuo/@nome
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

nome="Asdrubal da Silva"  
-----

nome="Quincas Borba"  
-----

nome="Doriana Margarina"

# Xpath

## Exemplos

/fichario/\*/idade

```
<fichario>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</fichario>
```

# Xpath

## Exemplos

/fichario/\*/idade

```
<fichario>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</fichario>
```

```
<idade>15</idade>
-----
<idade>33</idade>
-----
<idade>42</idade>
```

# XPath

[n]	enésimo
[last()]	último
[@atr]	seleciona elementos com atributo
[@atr=val] [@atr>val]	seleciona elementos com atributo que atende condição

# Xpath

## Exemplos

```
//individuo[2]
```

```
<fichario>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</fichario>
```

# Xpath

## Exemplos

```
//individuo[2]
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

```
<individuo nome="Quincas Borba">  
<idade>33</idade>  
<genero>masculino</genero>  
</individuo>
```



# Xpath

## Exemplos

```
//individuo[@nome="Quincas Borba"]
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

# Xpath

## Exemplos

```
//individuo[@nome="Quincas Borba"]
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

```
<individuo nome="Quincas Borba">  
<idade>33</idade>  
<genero>masculino</genero>  
</individuo>
```

# Xpath

## Exemplos

```
//individuo[@nome="Quincas Borba"]/idade
```

```
<fichario>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</fichario>
```

# Xpath

## Exemplos

```
//individuo[@nome="Quincas Borba"]/idade
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

```
<idade>33</idade>
```

# Xpath

## Exemplos

```
//individuo[@nome="Quincas Borba"]/idade/text()
```

```
<fichario>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</fichario>
```

# Xpath Exemplos

```
//individuo[@nome="Quincas Borba"]/idade/text()
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

33

# Xpath

## Exemplos

```
//individuo[idade>20]/@nome
```

```
<fichario>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</fichario>
```

# Xpath Exemplos

```
//individuo[idade>20]/@nome
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

```
nome="Quincas Borba"  
-----  
nome="Doriana Margarina"
```



# Exercício 2

- Construa uma comando SELECT que retorne dados equivalentes a este XPath

```
//individuo[idade>20]/@nome
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

```
nome="Quincas Borba"  
-----  
nome="Doriana Margarina"
```

# XQuery

FOR <variable bindings to individual nodes (elements)>

LET <variable bindings to collections of nodes (elements)>

WHERE <qualifier conditions>

RETURN <query result specification>

(Elmasri, 2011)

# XQuery Exemplos

<http://try.zorba.io/>

```
xquery version "1.0";  
let $message := 'Dinotopia'  
return  
<livro>{$message}</livro>
```

# XQuery Exemplos

<http://try.zorba.io/>

```
xquery version "1.0";  
let $message := 'Dinotopia'  
return  
<livro>{$message}</livro>
```

```
<?xml version="1.0" encoding="UTF-8"?>  
<livro>Dinotopia</livro>
```

# XML no resultado

- `{}` → para indicar valores de retorno dentro do XML
- `data()` → extrai o conteúdo literal do elemento ou atributo

# XQuery

## Label parte do XML externo

<http://try.zorba.io/>

```
xquery version "1.0";  
let $message := 'Dinotopia'  
return  
<livro>Título: {$message}</livro>
```

```
<?xml version="1.0" encoding="UTF-8"?>  
<livro>Título: Dinotopia</livro>
```

# XQuery

## Label dentro da { }

<http://try.zorba.io/>

```
xquery version "1.0";  
let $message := 'Dinotopia'  
return  
<livro>{data('Título:'), $message}</livro>
```

```
<?xml version="1.0" encoding="UTF-8"?>  
<livro>Título: Dinotopia</livro>
```

# XQuery XML Base

<http://www.ic.unicamp.br/~santanch/teaching/db/xml/fichario.xml>

```
<fichario>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</fichario>
```



# XQuery XML Base

`http://www.ic.unicamp.br/~santanch/teaching/db/xml/fichario.xml`

- Para simplificar vou chamá-lo de

`icunicamp:fichario.xml`

# XQuery

## let/return

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
return $fichariodoc/fichario
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

# XQuery

## let/return

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
return $fichariodoc/fichario
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

# XQuery

## Xquery e XPath

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
return $fichariodoc/fichario/individuo/idade
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

# XQuery

## Xquery e XPath

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
return $fichariodoc/fichario/individuo/idade
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

```
<idade>15</idade>  
<idade>33</idade>  
<idade>42</idade>
```

# XQuery

## Xquery e XPath

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
return $fichariodoc//individuo[idade>20][genero="masculino"]
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

# XQuery

## Xquery e XPath

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
return $fichariodoc//individuo[idade>20][genero="masculino"]
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

```
<individuo nome="Quincas Borba">  
  <idade>33</idade>  
  <genero>masculino</genero>  
</individuo>
```

# XQuery count()

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
return count($fichariodoc//individuo)
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```



# XQuery count()

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
return count($fichariodoc//individuo)
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

3

# XQuery count() e XPath

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
return count($fichariodoc//individuo[idade>20][genero="masculino"])
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

# XQuery count() e XPath

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
return count($fichariodoc//individuo[idade>20][genero="masculino"])
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

1

# XQuery for

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
for $i in ($fichariodoc//individuo)  
return $i/idade
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

# XQuery for

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
for $i in ($fichariodoc//individuo)  
return $i/idade
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

```
<idade>15</idade>  
<idade>33</idade>  
<idade>42</idade>
```

# XQuery where

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
for $i in ($fichariodoc//individuo)  
where $i[idade>17]  
return $i
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

# XQuery where

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
for $i in ($fichariodoc//individuo)  
where $i[idade>17]  
return $i
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

```
<individuo nome="Quincas Borba">  
  <idade>33</idade>  
  <genero>masculino</genero>  
</individuo>  
<individuo nome="Doriana Margarina">  
  <idade>42</idade>  
  <genero>feminino</genero>  
</individuo>
```

# XQuery

## XML no resultado

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
for $i in ($fichariodoc//individuo)  
where $i[idade>17]  
return <maior>{data($i/@nome)}</maior>
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```



# XQuery

## XML no resultado

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
for $i in ($fichariodoc//individuo)  
where $i[idade>17]  
return <maior>{data($i/@nome)}</maior>
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

# Exercício 3

- Escreva uma consulta SQL equivalente ao XQuery:

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
for $i in ($fichariodoc//individuo)  
where $i[idade>17]  
return {data($i/@nome)}
```

# XQuery

## XML no resultado

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
for $i in ($fichariodoc//individuo)  
where $i[idade>17]  
return <maior>{data($i/@nome)}</maior>
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

```
<maior>Quincas Borba</maior>  
<maior>Doriana Margarina</maior>
```

# XQuery

## XML no resultado

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
for $i in ($fichariodoc//individuo)  
where $i[idade>17]  
return <maior>{data($i/@nome)}</maior>
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

```
<maior>Quincas Borba</maior>  
<maior>Doriana Margarina</maior>
```

# XQuery

## Xquery dentro de outro XQuery

```
let $fichariodoc := doc('icunicamp:fichario.xml')
return
<classificacao>
{
  for $i in ($fichariodoc//individuo)
  where $i[idade>17]
  return <maior>{data($i/@nome)}</maior>
}
</classificacao>
```

```
<fichario>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</fichario>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<classificacao>
  <maior>Quincas Borba</maior>
  <maior>Doriana Margarina</maior>
</classificacao>
```

# Exercício 4

- Escreva um Xquery que transforme o arquivo XML em uma sequência de INSERTS.

# XQuery order by

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
for $i in ($fichariodoc//individuo)  
where $i[idade>17]  
order by $i/@nome  
return <maior>{data($i/@nome)}</maior>
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

# XQuery order by

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
for $i in ($fichariodoc//individuo)  
where $i[idade>17]  
order by $i/@nome  
return <maior>{data($i/@nome)}</maior>
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

```
<maior>Doriana Margarina</maior>  
<maior>Quincas Borba</maior>
```



# XQuery if

```
let $fichariodoc := doc('icunicamp:fichario.xml')  
  
for $i in ($fichariodoc//individuo)  
return if ($i[idade>=18])  
  then <maior>{data($i/@nome)}</maior>  
  else <menor>{data($i/@nome)}</menor>
```

```
<fichario>  
  <individuo nome="Asdrubal da Silva">  
    <idade>15</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Quincas Borba">  
    <idade>33</idade>  
    <genero>masculino</genero>  
  </individuo>  
  <individuo nome="Doriana Margarina">  
    <idade>42</idade>  
    <genero>feminino</genero>  
  </individuo>  
</fichario>
```

```
<menor>Asdrubal da Silva</menor>  
<maior>Quincas Borba</maior>  
<maior>Doriana Margarina</maior>
```

```
LET $d := doc(www.company.com/info.xml)
FOR $x IN $d/company/project[projectNumber = 5]/projectWorker,
    $y IN $d/company/employee
WHERE $x/hours gt 20.0 AND $y.ssn = $x.ssn
RETURN <res> $y/employeeName/firstName, $y/employeeName/lastName,
    $x/hours </res>
```

- 
1. FOR \$x IN  
doc(www.company.com/info.xml)  
//employee [employeeSalary gt 70000]/employeeName  
RETURN <res> \$x/firstName, \$x/lastName </res>
  2. FOR \$x IN  
doc(www.company.com/info.xml)/company/employee  
WHERE \$x/employeeSalary gt 70000  
RETURN <res> \$x/employeeName/firstName, \$x/employeeName/lastName </res>
  3. FOR \$x IN  
doc(www.company.com/info.xml)/company/project[projectNumber = 5]/projectWorker,  
\$y IN doc(www.company.com/info.xml)/company/employee  
WHERE \$x/hours gt 20.0 AND \$y.ssn = \$x.ssn  
RETURN <res> \$y/employeeName/firstName, \$y/employeeName/lastName, \$x/hours </res>

**Figure 12.7**

Some examples of XQuery queries on XML documents that follow the XML schema file *company* in Figure 12.5.

(Elmasri, 2011)

```
LET $d := doc(www.company.com/info.xml)
FOR $x IN $d/company/project[projectNumber = 5]/projectWorker,
    $y IN $d/company/employee
WHERE $x/hours gt 20.0 AND $y.ssn = $x.ssn
RETURN <res> $y/employeeName/firstName, $y/employeeName/lastName,
    $x/hours </res>
```

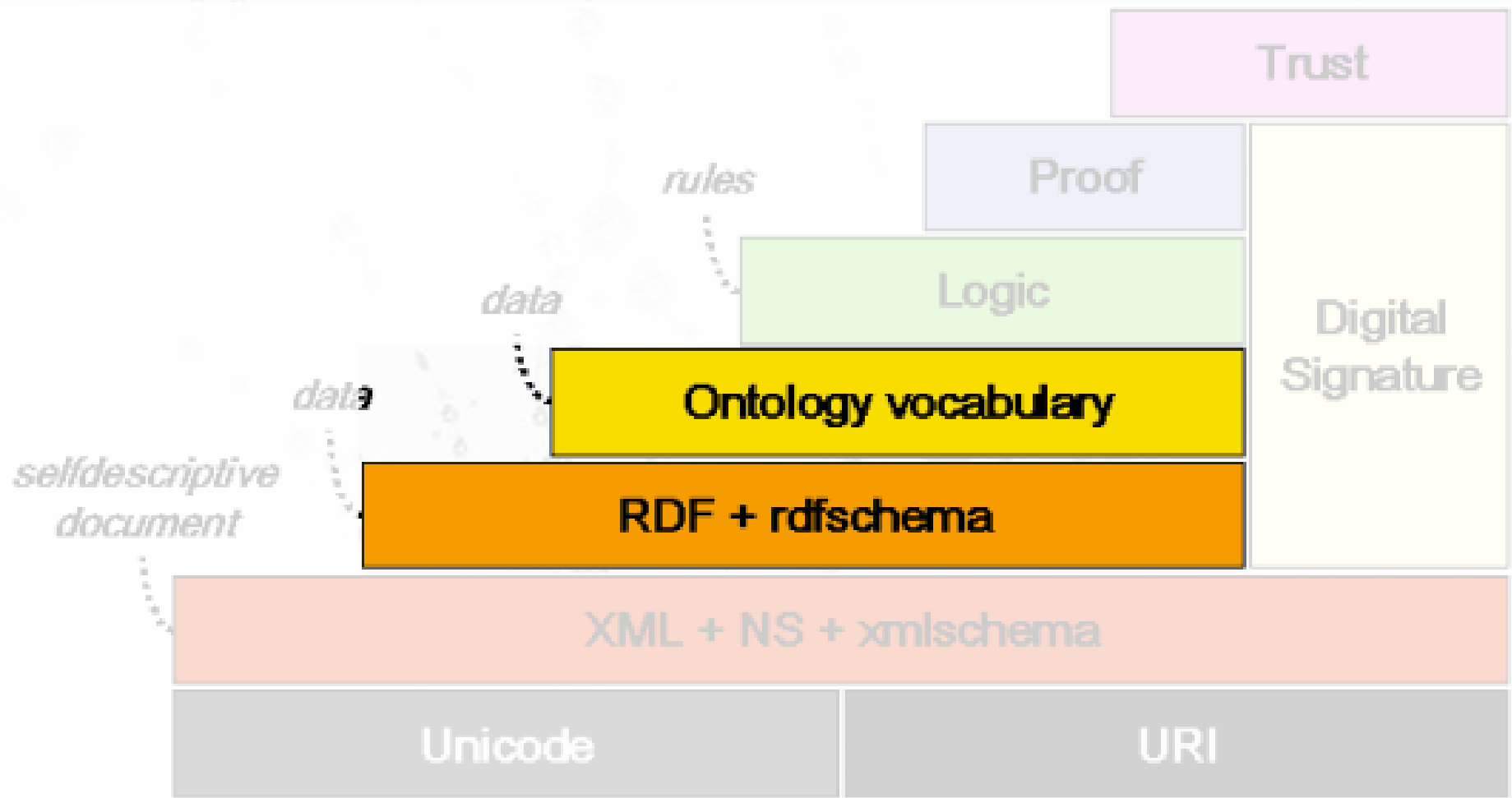
- 
1. FOR \$x IN  
doc(www.company.com/info.xml)  
//employee [employeeSalary gt 70000]/employeeName  
RETURN <res> \$x/firstName, \$x/lastName </res>
  2. FOR \$x IN  
doc(www.company.com/info.xml)/company/employee  
WHERE \$x/employeeSalary gt 70000  
RETURN <res> \$x/employeeName/firstName, \$x/employeeName/lastName </res>
  3. FOR \$x IN  
doc(www.company.com/info.xml)/company/project[projectNumber = 5]/projectWorker,  
\$y IN doc(www.company.com/info.xml)/company/employee  
WHERE \$x/hours gt 20.0 AND \$y.ssn = \$x.ssn  
RETURN <res> \$y/employeeName/firstName, \$y/employeeName/lastName, \$x/hours </res>

**Figure 12.7**

Some examples of XQuery queries on XML documents that follow the XML schema file *company* in Figure 12.5.

(Elmasri, 2011)

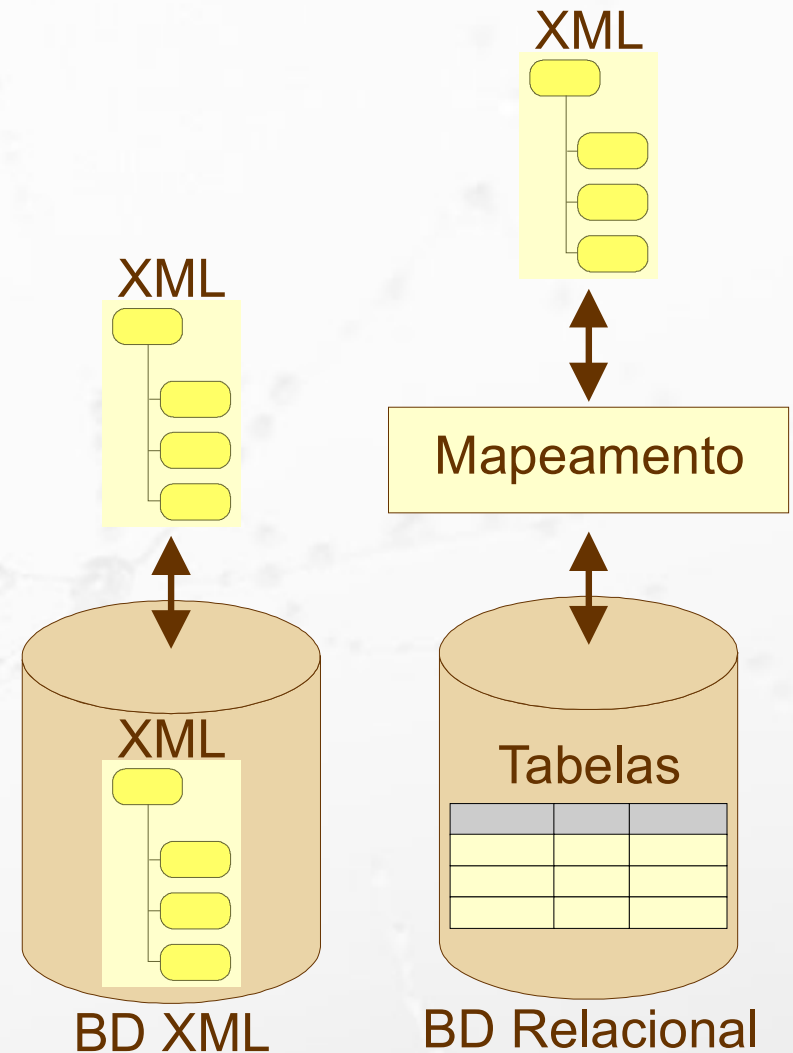
# Web Semântica



# XML e Bancos de Dados

# Banco de dados & XML

- Dois tipos:
  - SGBD XML nativo;
  - SGBD relacional que mapeia dados XML para sua estrutura interna e vice-versa.

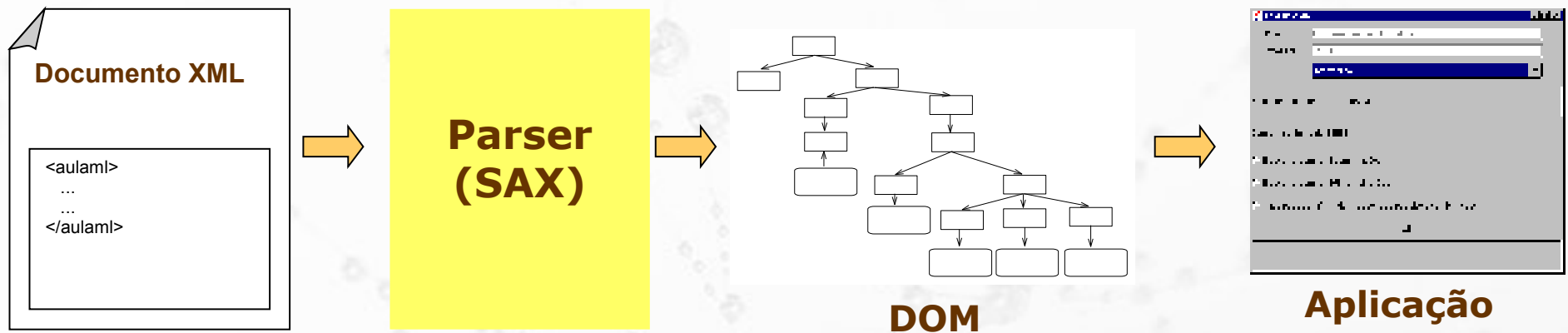


# Banco de dados & XML

## Aspectos conceituais

- Padrão para:
  - intercâmbio de dados × armazenamento.
- Modelo:
  - documentos × dados (registros).

# Parser XML



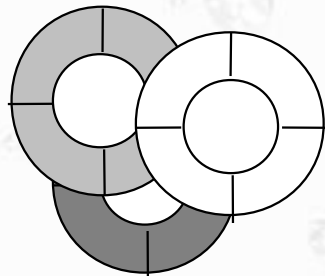


# Introdução

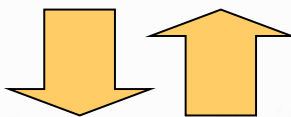
Diversas tecnologias têm sido criadas para o processamento de documentos XML.

# Aplicação

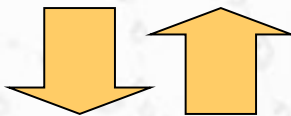
**Server Pages**



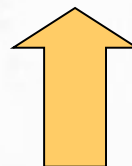
**Classes**



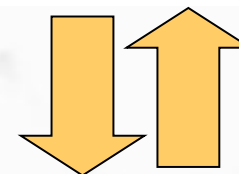
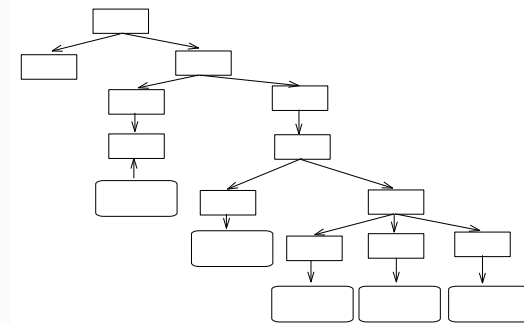
**Data-Binding**



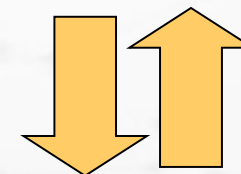
**Eventos**



**SAX**



**DOM**



```
<aulaml>
<curso>
...
</curso>
<quadro>
<texto>
...
</texto>
<teste>
...
</teste>
</quadro>
</aulaml>
```

```
<aulaml>
<curso>
...
</curso>
<quadro>
<texto>
...
</texto>
<teste>
...
</teste>
</quadro>
</aulaml>
```

```
<aulaml>
<curso>
...
</curso>
<quadro>
<texto>
...
</texto>
<teste>
...
</teste>
</quadro>
</aulaml>
```

```
<aulaml>
<curso>
...
</curso>
<quadro>
<texto>
...
</texto>
<teste>
...
</teste>
</quadro>
</aulaml>
```

**XML**

# Introdução

Dentre estas tecnologias duas se destacaram e se tornaram referência:

- SAX - Simple API for XML
- DOM - Document Object Model

# SAX

- API baseada em eventos.
- Se tornou a mais estável API XML largamente utilizada [DOD01].
- Iniciou como uma solução para acesso a documentos XML por programas Java.
- Hoje tem sido portada para outras linguagens de programação, tal como: C++, Pascal, Perl, Python, etc.

# SAX - Estudo de Caso

```
<FICHARIO>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</FICHARIO>
```

# SAX - Estudo de Caso

===== Início do Documento =====

Início de elemento: DOCUMENTO

Início de elemento: individuo

(atributos): nome=Asdrubal da Silva;

Início de elemento: IDADE

Texto: 15

Final de elemento : IDADE

Início de elemento: GENERO

Texto: masculino

Final de elemento : GENERO

Final de elemento : individuo

Início de elemento: individuo

(atributos): nome=Quincas Borba;

Início de elemento: IDADE

Texto: 33

Final de elemento : IDADE

Início de elemento: GENERO

Texto: masculino

Final de elemento : GENERO

Final de elemento : individuo

Início de elemento: individuo

(atributos): nome=Doriana Margarina;

Início de elemento: IDADE

Texto: 42

Final de elemento : IDADE

Início de elemento: GENERO

Texto: feminino

Final de elemento : GENERO

Final de elemento : individuo

Final de elemento : DOCUMENTO

===== Final do Documento =====

# Eventos de conteúdo

```
public class SAXBasico extends
    org.xml.sax.helpers.DefaultHandler
{

    public void startDocument () ...

    public void startElement (...) ...

    public void characters (...) ...

    public void endElement (...) ...

    public void endDocument () ...

}
```

# Eventos de conteúdo

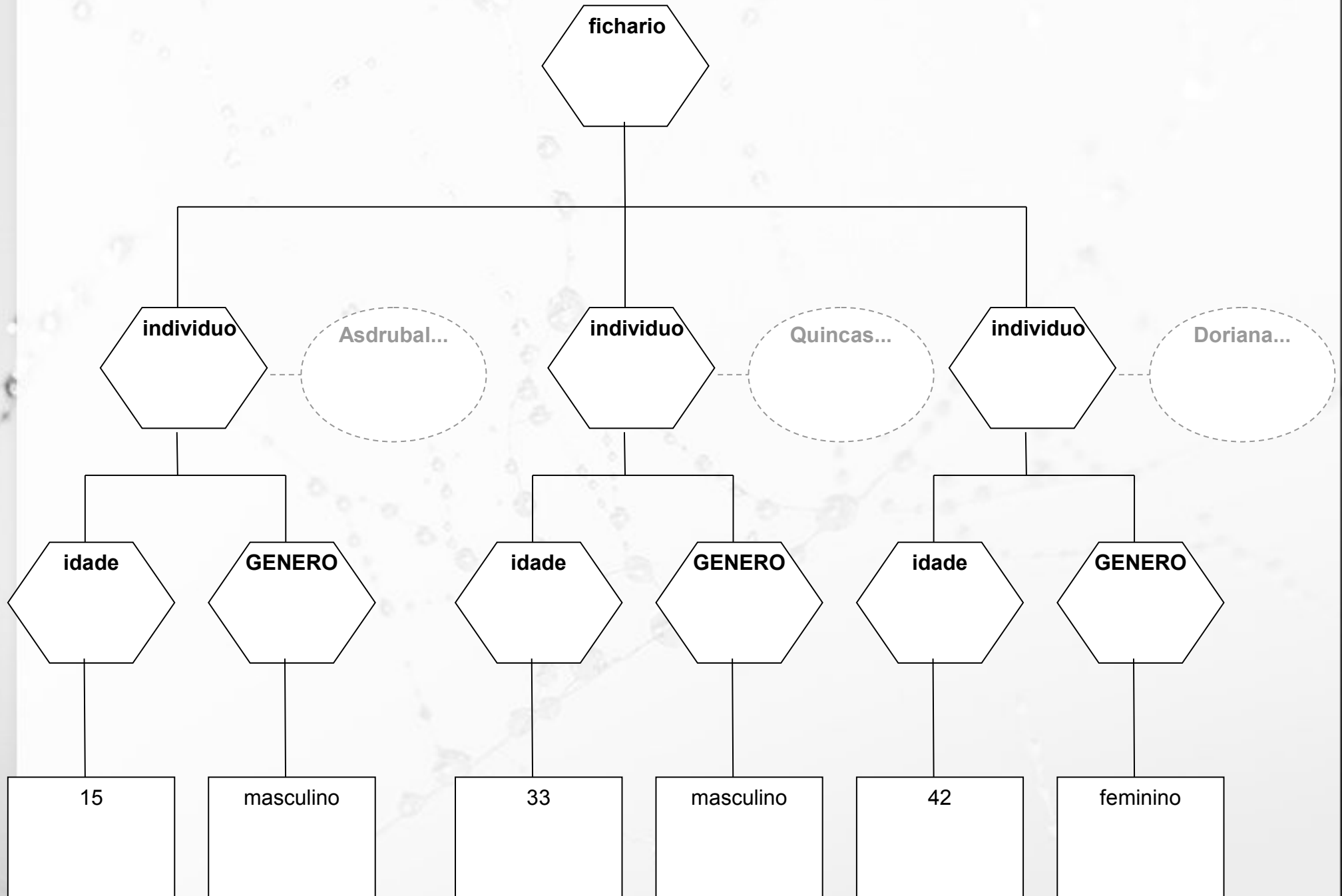
<b>Método</b>	<b>Acionado quando o <i>parser</i> encontra</b>
<code>startDocument</code>	início do documento
<code>startElement</code>	início de um elemento
<code>characters</code>	conteúdo texto
<code>endElement</code>	final de um elemento
<code>endDocument</code>	final do documento



# DOM

- DOM define uma API para documentos XML e HTML.
- Ele acrescenta ao padrão destas linguagens toda a funcionalidade e flexibilidade que um programa precisa para acessar e manipular documentos.
- Definido em IDL, ECMAScript e Java.

# Document Object Model



# DOM Level

- **Level 0** - define funcionalidades equivalentes ao Netscape Navigator 3.0 e o Microsoft Internet Explorer 3.0.
- **Level 1** - especifica recursos para navegação e manipulação de estrutura e conteúdo de documentos XML e HTML.
- **Level 2** - estende alguns recursos do *Level 1* e acrescenta suporte a: *Cascading Style Sheets*, Eventos, etc.
- **Level 3** - estende alguns recursos do *Level 2* e acrescenta suporte a: esquemas abstratos (DTD, XML Schema, etc.), recursos de leitura e gravação, etc.

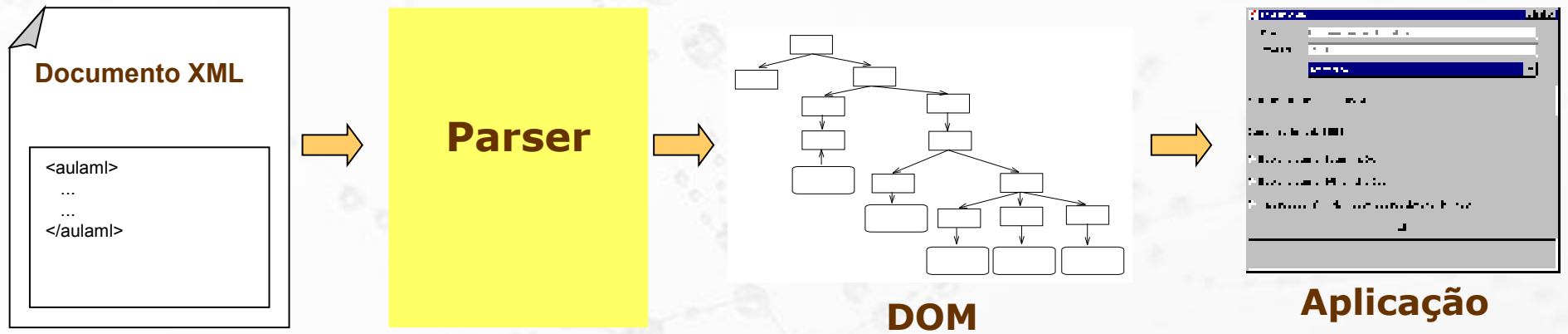
# DOM Core & DOM HTML

- O *DOM Level 1* é dividido em duas partes: DOM Core e DOM HTML.
- **DOM Core** - define o conjunto de funcionalidades básicas para documentos XML.
- **DOM HTML** - está montado sobre o DOM Core e acrescenta funcionalidades para lidar com HTML.

# DOM - Estudo de Caso

```
<FICHARIO>
  <individuo nome="Asdrubal da Silva">
    <idade>15</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Quincas Borba">
    <idade>33</idade>
    <genero>masculino</genero>
  </individuo>
  <individuo nome="Doriana Margarina">
    <idade>42</idade>
    <genero>feminino</genero>
  </individuo>
</FICHARIO>
```

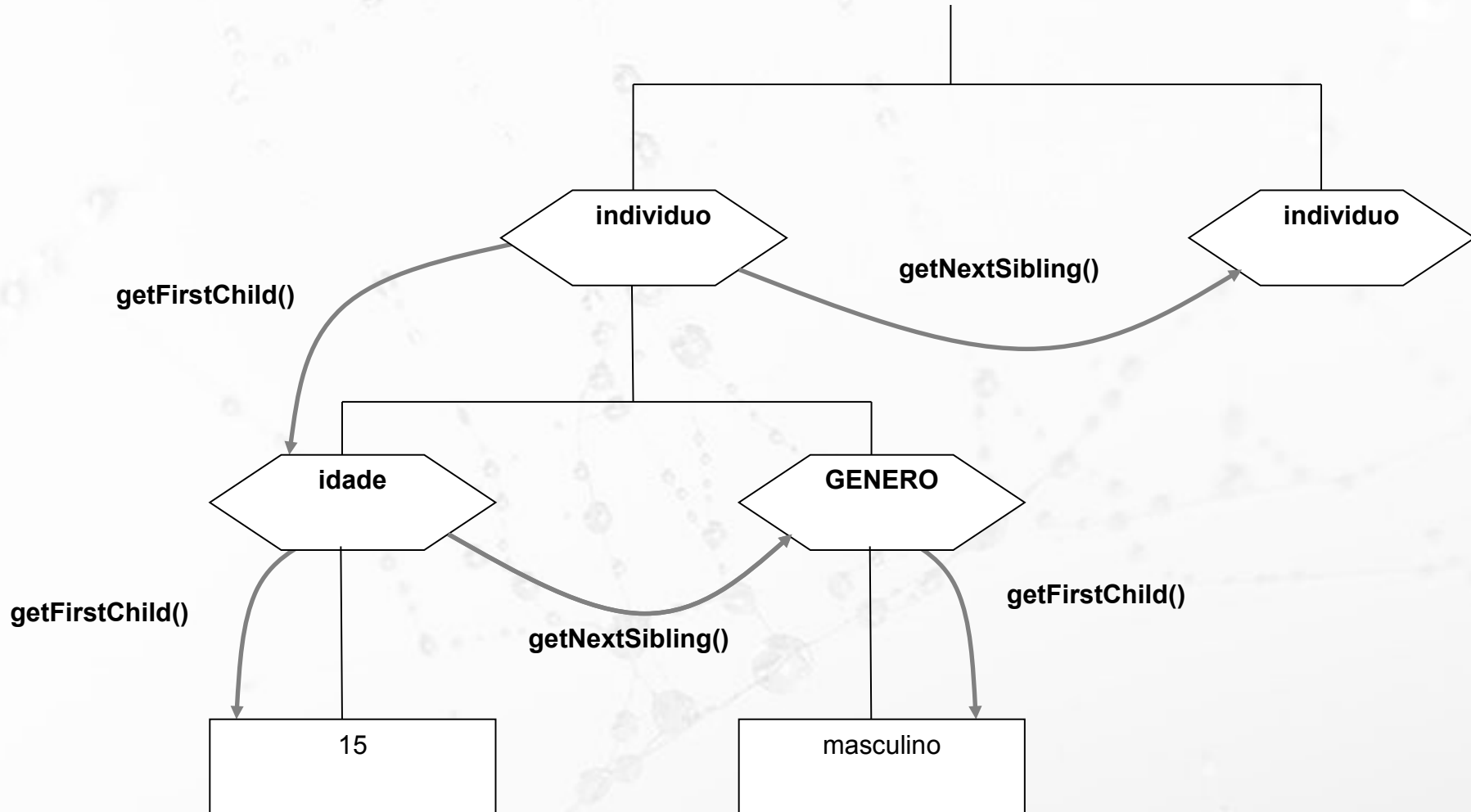
# Processo



# Interfaces

- **Node** - esta interface representa genericamente qualquer nó da árvore.
- **Element** - acrescenta propriedades e métodos específicos de um nó do tipo elemento.
- **Document** - interface do nó raiz da árvore que representa o documento completo.
- **NodeList** - representa uma lista de nós. Pode representar, por exemplo, a lista de filhos de um nó.

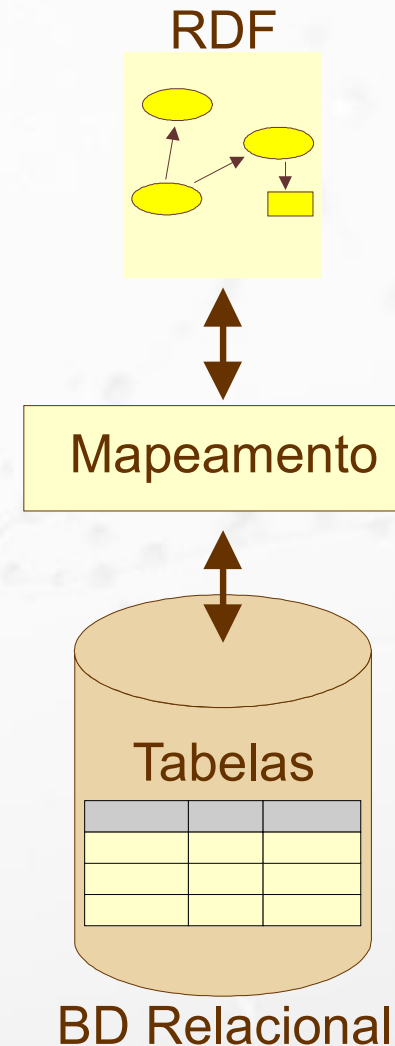
# Navegar pelo Documento



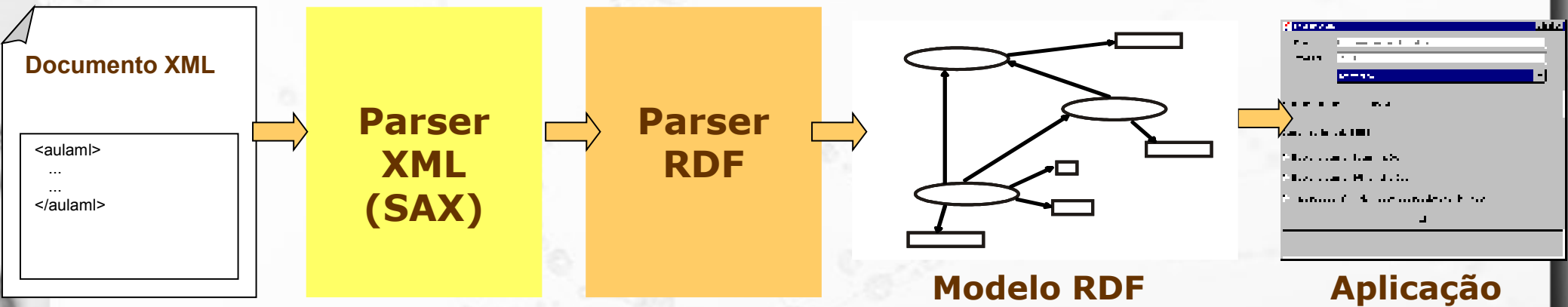


# Banco de dados & RDF

- Não associado a BD XML.
- Modelo de dados RDF mais próximo ao relacional que XML.



# Parser RDF



# Referências Bibliográficas

- Elmasri, Ramez; Navathe, Shamkant B. (2010) **Sistemas de Banco de Dados**. Pearson, 6a edição em português.
- Lee, T.B. **Notation 3**. March 2006. on-line:  
<http://www.w3.org/DesignIssues/Notation3.html>
- Lee, T.B.; Hendler, J. & Lassila, O. **The Semantic Web**. Scientific American, 2001, 284, 28-37
- Moats R. **URN Syntax**. Network Working Group, May 1997.
- Sollins, K. and Masinter, L. **Functional Requirements for Uniform Resource Names**. Network Working Group, December 1994.
- Wang, X.; Gorlitsky, R. & Almeida, J.S. **From XML to RDF: how semantic web technologies will change the design of 'omic' standards** Nat Biotech. 2005, 23, 1099-1103.
- Whiteside, Arliss. **URNs of definitions in ogc namespace**. version: 1.0.0, document: 05-010. January 2005.

# Referências Bibliográficas

- Leise, F.; Fast, K.; Steckel, M. **What Is A Controlled Vocabulary?** Boxes and Arrows, Dezembro 2002, online:  
[http://www.boxesandarrows.com/view/what\\_is\\_a\\_controlled\\_vocabulary\\_](http://www.boxesandarrows.com/view/what_is_a_controlled_vocabulary_)
- Amy J. Warner. **Taxonomy Primer**, online:  
<http://www.lexonomy.com/publications/aTaxonomyPrimer.html>,  
visitado em 20/08/2010.
- Wellisch, H. **Indexing from A to Z**. New York: H.W. Wilson, 1995. p. 214.
- Wilde, Erik. **XML Foundations** (slides). UC Berkeley iSchool, Aug 2006. <http://dret.net/lectures/xml-fall06/basics>

A black and white photograph of a spider web with several dew drops of varying sizes. The web is stretched across the frame, and the dew drops are scattered throughout, some in sharp focus and others blurred. The background is a soft, out-of-focus grey.

# André Santanchè

<http://www.ic.unicamp.br/~santanche>

# License

- These slides are shared under a Creative Commons License. Under the following conditions: Attribution, Noncommercial and Share Alike.
- See further details about this Creative Commons license at: <http://creativecommons.org/licenses/by-nc-sa/3.0/>