# Introdução de XML

## Banco de Dados II

**Prof. Guilherme Tavares de Assis**

**Universidade Federal de Ouro Preto – UFOP**

**Instituto de Ciências Exatas e Biológicas – ICEB**

**Departamento de Computação – DECOM**

# Dados da *Web*

- A *Web* representa, nos dias de hoje, um repositório universal de dados, onde:
  - a quantidade de sítios existentes e o volume disponível de dados são grandes;
  - a informação está espalhada e nem sempre organizada;
  - geralmente, é difícil manipular/consultar dados de múltiplas fontes.

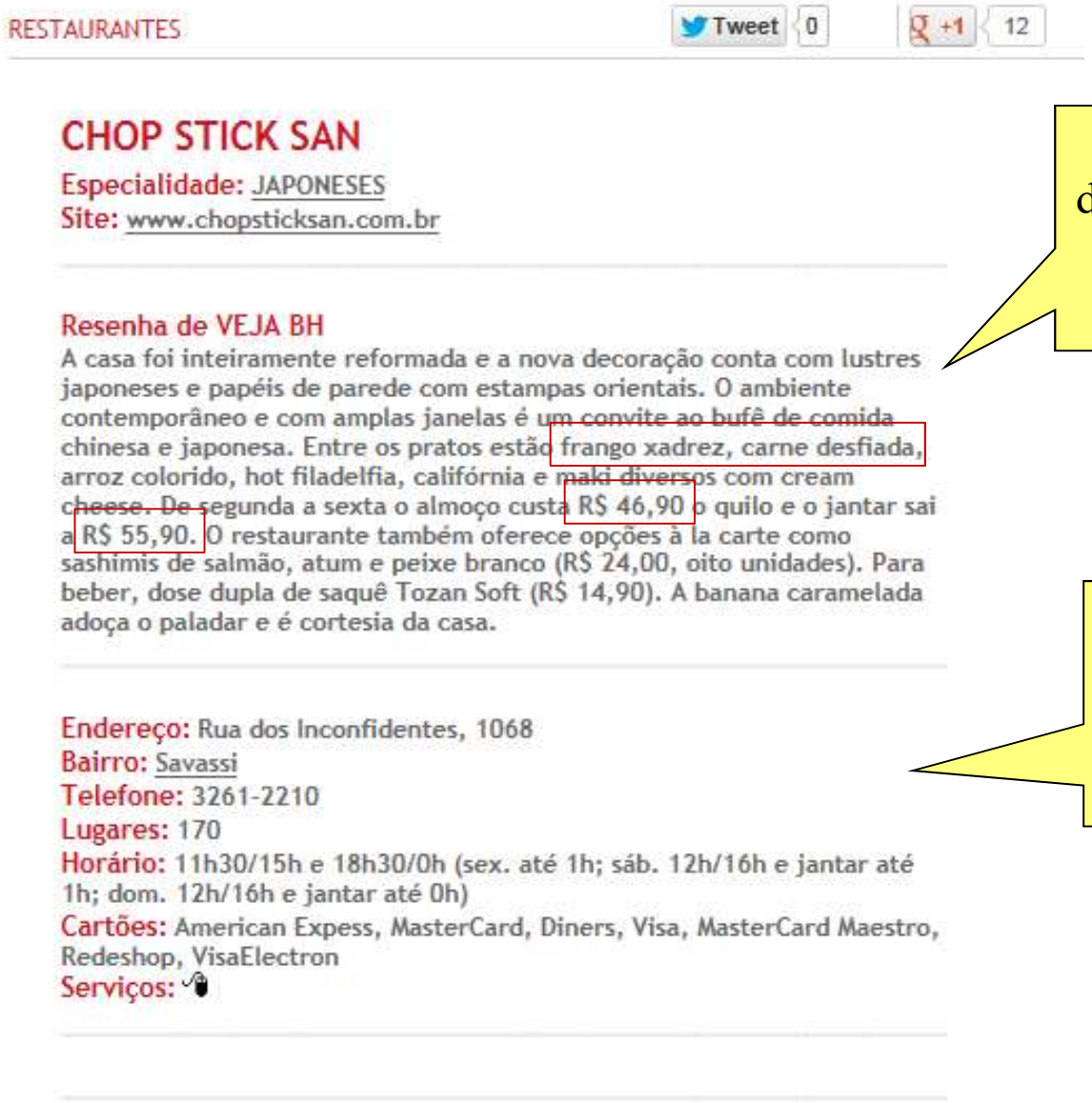
- Algumas características dos dados da *Web* são:
  - encontram-se disponíveis, geralmente, por meio de documentos textuais;
  - são constantemente alterados;
  - possuem estrutura implícita e não-declarada.

# Gerência de Dados da *Web*

- A área "Gerência de Dados da *Web*" trata de problemas relacionados a coleta, extração, consulta, modelagem, armazenamento, transformação e integração de dados existentes na *Web*.

- Exemplo:
  - Deseja-se gerar uma base contendo dados sobre restaurantes de comida japonesa na região da Savassi em Belo Horizonte.
  - Neste caso, deve-se tratar devidamente dados contidos em diferentes fontes (páginas) da *Web*.

# Gerência de Dados da *Web*



RESTAURANTES

Tweet 0 +1 12

**CHOP STICK SAN**

Especialidade: JAPONESES
Site: www.chopsticksan.com.br

Resenha de VEJA BH

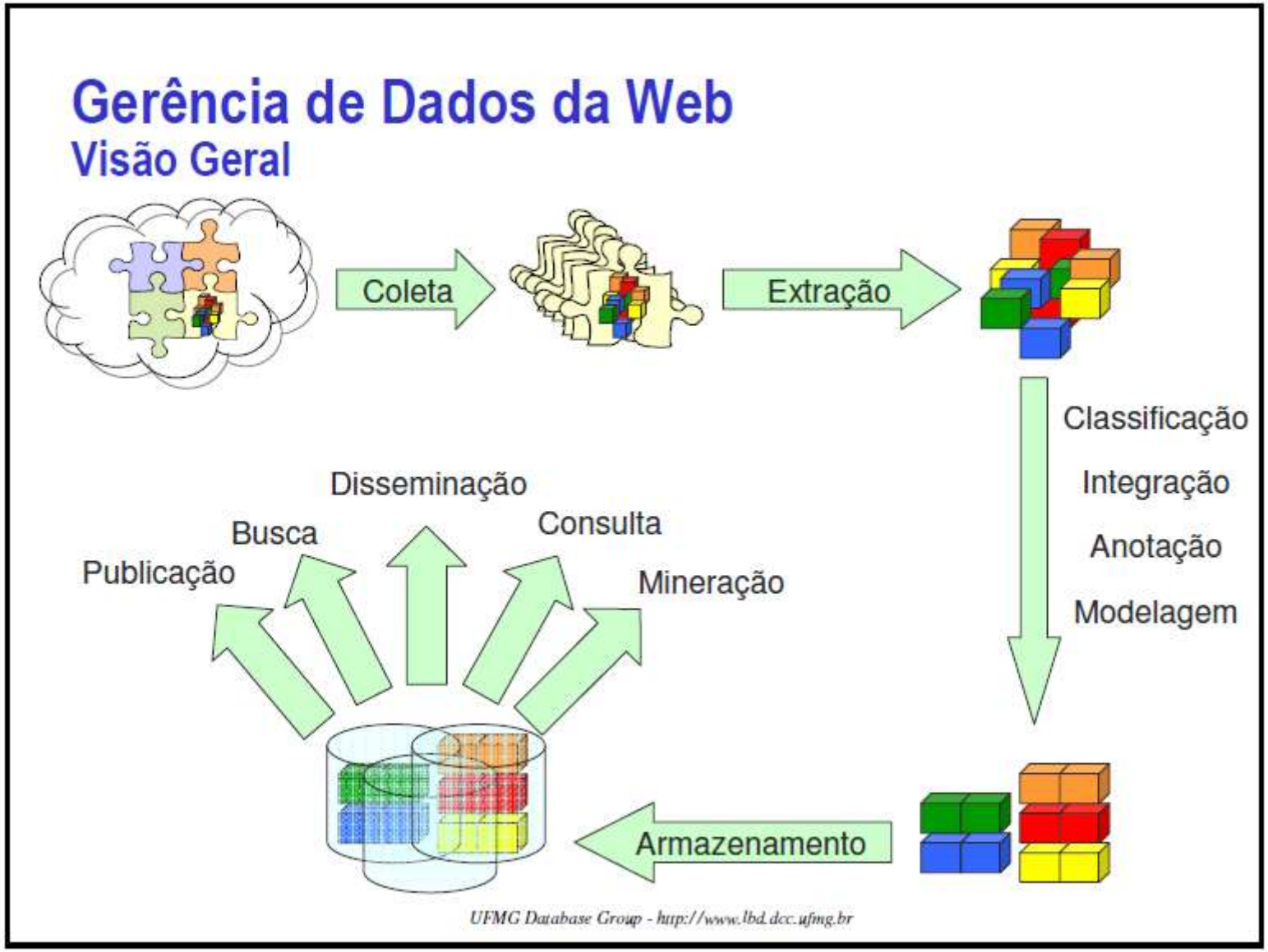
A casa foi inteiramente reformada e a nova decoração conta com lustres japoneses e papéis de parede com estampas orientais. O ambiente contemporâneo e com amplas janelas é um convite ao bufê de comida chinesa e japonesa. Entre os pratos estão frango xadrez, carne desfiada, arroz colorido, hot filadelfia, califórnia e maki diversos com cream cheese. De segunda a sexta o almoço custa R$ 46,90 o quilo e o jantar sai a R$ 55,90. O restaurante também oferece opções à la carte como sashimis de salmão, atum e peixe branco (R$ 24,00, oito unidades). Para beber, dose dupla de saquê Tozan Soft (R$ 14,90). A banana caramelada adoça o paladar e é cortesia da casa.

Endereço: Rua dos Inconfidentes, 1068
Bairro: Savassi
Telefone: 3261-2210
Lugares: 170
Horário: 11h30/15h e 18h30/0h (sex. até 1h; sáb. 12h/16h e jantar até 1h; dom. 12h/16h e jantar até 0h)
Cartões: American Express, MasterCard, Diners, Visa, MasterCard Maestro, Redeshop, VisaElectron
Serviços:

Na resenha, há alguns itens da culinária japonesa e preços que podem ser importantes para uma consulta.

Há alguns dados estruturados, referentes ao restaurante em questão, nesse sítio.

# Gerência de Dados da *Web*



# Dados Semi-estruturados

- Dados estruturados são dados que possuem uma estrutura bem definida e rígida.
  - Ex.: tuplas de uma relação em um esquema relacional.
- Dados não-estruturados são dados que não possuem alguma estrutura.
  - Ex.: páginas no formato HTML, onde aparecem *tags* pré-definidas que especificam apenas a formatação dos dados e não o significado dos mesmos.
- Dados semi-estruturados são dados que possuem uma estrutura flexível (não rígida).
  - Ex.: arquivos BibTex, dados da *Web*.

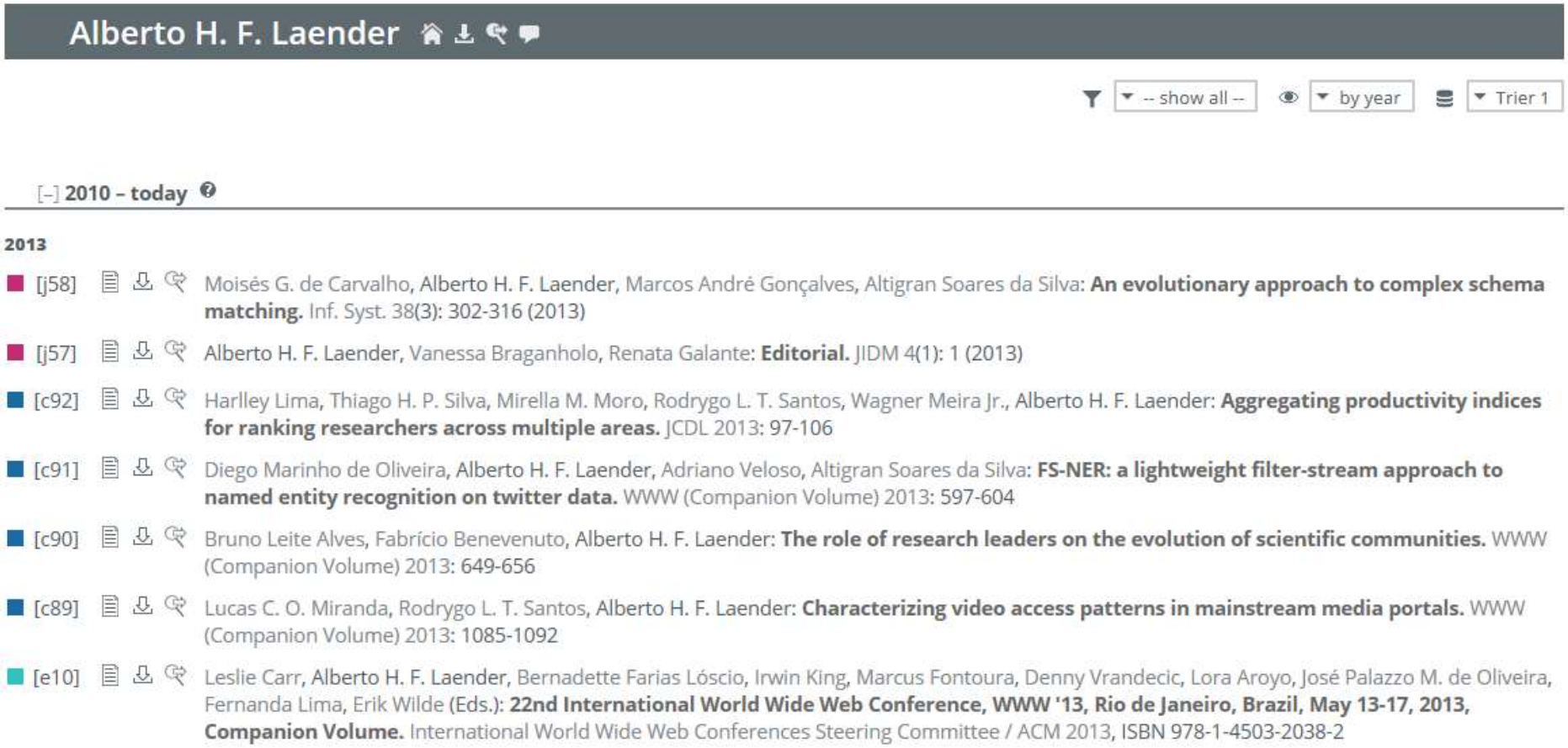
# Dados Semi-estruturados

```
@article{cha91,
      author = {S.K. Cha},
      title = {Kaleidoscope: A Cooperative Menu-Guided Query Interface (SQL Version)},
      journal = {IEEE Transactions on Knowledge and Data Engineering},
      year = 1991,
      volume = 3,
      number = 1,
      pages = {42-47}
}
@book{dlr95,
     author = {C. Delobel and C. Lécluse and P. Richard},
     title = {Databases: From Relational to Object-Oriented Systems},
     year = 1995,
     publisher = {International Thompson Computer Press},
     address = {London, UK}
}
@conference{el85,
      author = {R. Elmasri and J.A. Larson},
      title = {A Graphical Query Facility for ER Databases},
      booktitle = {Proceedings of 4th International Conference on Entity-Relationship Approach},
      year = 1985,
      address = {Chicago, Illinois},
      pages = {263-245}
}
```

Exemplo de dados semi-estruturados: arquivo BibTex

# Dados Semi-estruturados

Exemplo de uma página *Web* com dados semi-estruturados: dados bibliográficos da DBLP



Alberto H. F. Laender

-- show all -- | by year | Trier 1

[–] 2010 – today

2013

[j58] Moisés G. de Carvalho, Alberto H. F. Laender, Marcos André Gonçalves, Altigran Soares da Silva: **An evolutionary approach to complex schema matching.** Inf. Syst. 38(3): 302-316 (2013)

[j57] Alberto H. F. Laender, Vanessa Braganholo, Renata Galante: **Editorial.** JIDM 4(1): 1 (2013)

[c92] Harlley Lima, Thiago H. P. Silva, Mirella M. Moro, Rodrygo L. T. Santos, Wagner Meira Jr., Alberto H. F. Laender: **Aggregating productivity indices for ranking researchers across multiple areas.** JCDL 2013: 97-106

[c91] Diego Marinho de Oliveira, Alberto H. F. Laender, Adriano Veloso, Altigran Soares da Silva: **FS-NER: a lightweight filter-stream approach to named entity recognition on twitter data.** WWW (Companion Volume) 2013: 597-604

[c90] Bruno Leite Alves, Fabrício Benevenuto, Alberto H. F. Laender: **The role of research leaders on the evolution of scientific communities.** WWW (Companion Volume) 2013: 649-656

[c89] Lucas C. O. Miranda, Rodrygo L. T. Santos, Alberto H. F. Laender: **Characterizing video access patterns in mainstream media portals.** WWW (Companion Volume) 2013: 1085-1092

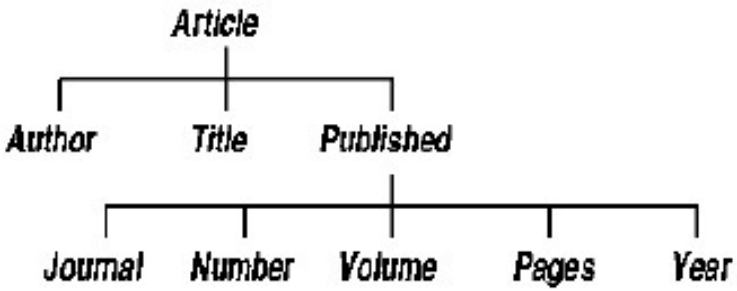
[e10] Leslie Carr, Alberto H. F. Laender, Bernadette Farias Lóscio, Irwin King, Marcus Fontoura, Denny Vrandecic, Lora Aroyo, José Palazzo M. de Oliveira, Fernanda Lima, Erik Wilde (Eds.): **22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume.** International World Wide Web Conferences Steering Committee / ACM 2013, ISBN 978-1-4503-2038-2

# Dados Semi-estruturados

Moisés G. de Carvalho, Alberto H. F. Laender, Marcos André Gonçalves, Altigran Soares da Silva: **An evolutionary approach to complex schema matching.** Inf. Syst. 38(3): 302-316 (2013)

Alberto H. F. Laender, Vanessa Braganholo, Renata Galante: **Editorial.** JIDM 4(1): 1 (2013)

Artigos de periódicos





Artigos de anais

Harlley Lima, Thiago H. P. Silva, Mirella M. Moro, Rodrygo L. T. Santos, Wagner Meira Jr., Alberto H. F. Laender: **Aggregating productivity indices for ranking researchers across multiple areas.** JCDL 2013: 97-106

Diego Marinho de Oliveira, Alberto H. F. Laender, Adriano Veloso, Altigran Soares da Silva: **FS-NER: a lightweight filter-stream approach to named entity recognition on twitter data.** WWW (Companion Volume) 2013: 597-604

# XML

- XML (*eXtensible Markup Language*) corresponde ao padrão W3C para complementação da HTML, visando o armazenamento e o intercâmbio de dados na *Web*.

- Motivação da criação da XML:
  - HTML descreve apenas apresentação (formato), não incluindo o esquema dos dados presentes em documentos.
  - XML descreve conteúdo.
  - Com documentos XML, é facilmente possível transferir dados estruturados ou semi-estruturados, via *Web*, entre distintas aplicações.

# XML

```
<h1> Bibliography </h1>
<p> <i> Foundations of Databases </i>
      Abiteboul, Hull, Vianu
      <br> Addison Wesley, 1995
<p> <i> Data on the Web </i>
      Abiteoul, Buneman, Suciu
      <br> Morgan Kaufmann, 1999
```

- HTML descreve apresentação.
  - Marcadores (*tags*) são definidos para indicar formato.

```
<bibliography>
    <book> <title> Foundations… </title>
           <author> Abiteboul </author>
           <author> Hull </author>
           <author> Vianu </author>
           <publisher> Addison Wesley </publisher>
           <year> 1995 </year>
    </book>
    …
</bibliography>
```

- XML descreve conteúdo.
  - Marcadores (*tags*) são definidos para indicar estrutura.

# XML

```
<bibliography>
    <description> SSD papers </description>
    <papers>
        <paper>
            <author> Abiteboul </author>
            <author> Vianu </author>
            <title> Regular path queries with constraints </title>
            <year> 1977 </year>
            <page> <first> 122 </first> <last> 133 </last> </page>
        </paper>
        <paper>
            <author> Abiteboul </author>
            <title> Querying semistructured data" </title>
            <year> 1977 </year>
        </paper>
        ...
    </papers>
</bibliography>
```

# XML – Sintaxe Básica

- Os componentes básicos de um documento XML são denominados elementos.

```
<author> Abiteboul </author>
<title> Regular path queries with constraints </title>
```

Os pares <author> </author> são denominados elementos e são definidos pelo usuário

- Os elementos podem ser decompostos em sub-elementos.

```
<page>
        <first> 122 </first>
        <last> 133 </last>
</page>
```

# XML – Sintaxe Básica

- Os elementos podem ser repetidos para representar uma determinada coleção.

```
<authors>
        <author> Abiteboul </author>
        <author> Vianu </author>
        ...
</authors>
```

# XML – Sintaxe Básica

- Elementos podem conter atributos, que constituem propriedades dos mesmos e são definidos como pares (nome = valor).

```
<product >
        <name language = "French"> trompete six trous </name>
        <price currency = "Euro"> 420.12 </price>
        <address format = "XLB56" language = "French">
                <street> 31 rue Croix-Basset </street>
                <zip> 92310 </zip>
                <city> Sevres </city>
                <country> France </country>
        </address>
</product >
```

# Documentos XML Bem Formados

- Um documento XML é considerado bem formado se:
  - todos os elementos estiverem corretamente aninhados;
  - os atributos de um mesmo elemento forem únicos.

- Os elementos encontram-se ordenados em um documento XML; porém, atributos não se encontram ordenados.

```
<person> <fname> John </fname >
         <lname> Smith </lname > </person>
<person> <lname> Smith </lname>
         <fname> John </fname> </person>
```

Os elementos não são equivalentes

```
<person fname = "John" lname = "Smith" />
<person lname = "Smith" fname = "John" />
```

Os elementos são equivalentes

# IDs e IDREFs

- Elementos podem conter referências que permitem a especificação de chaves identificadoras (únicas) e estrangeiras.
  - ID: atributo usado para identificar unicamente um elemento em um documento XML.
    - Valores associados devem ser distintos.
  - IDREF: atributo que referencia um elemento por meio do valor de seu atributo ID.
    - Valor associado deve existir como valor de algum atributo ID.
  - IDREFS: atributo que referencia um conjunto de elementos por meio dos valores de seus atributos ID.
    - Valores associados devem existir como valores de atributos ID.

# IDs e IDREFs

- Exemplo:

```
<state id="e3">
        <scode> MG </scode>
        <sname> Minas Gerais </sname>
</state>

<city id="c5">
        <ccode> OP </ccode>
        <cname> Ouro Preto </cname>
        <state-of-city idref="e3" />
</city>
```

Faz referência ao estado "e3"

# *Namespace*

- Já que XML permite que os autores criem seus próprios elementos, pode ocorrer confusão de nomenclaturas no intercâmbio ou na integração de distintos documentos XML, ou seja, uma mesma nomenclatura pode ser usada para definir elementos com distintas semânticas.

```
<codigo>
        ... if (x >= 90) { return 1; } ...
</codigo>
```

"codigo" refere-se a um programa escrito em uma determinada linguagem

```
<codigo>
        93253
</codigo>
```

"codigo" refere-se ao código de um produto qualquer de uma empresa.

# *Namespace*

- Para resolver uma possível confusão de nomenclaturas, utiliza-se *namespace* ao se definir os nomes dos elementos.
  - Logo, a finalidade do *namespace* é evitar conflito de nomes, semelhando-se à especificação de um sobrenome ao se definir o nome de um elemento.

```
<programacao:codigo>
        ... if (x >= 90) { return 1; } ...
</programacao:codigo>

<produto:codigo>
        93253
</produto:codigo>
```

Os namespaces "programacao" e "produto" são comumente chamados de prefixos de elementos

# *Namespace*

- Como garantir que um *namespace* é único?
  - Deve-se associar ao prefixo uma cadeia de caracteres que apenas o autor tenha controle e que, assim, ninguém irá usá-la.
    - Para tal associação formal, utiliza-se a palavra-chave "*xmlns*" no documento XML.
  - Pode ser usada qualquer cadeia de caracteres; geralmente, uma cadeia que se tem controle é um URI (*Uniform Resource Identifier*) com domínio de *internet*.
    - Na prática, são utilizadas URLs (*Universal Resource Locator*), uma vez que as URLs são garantidamente únicas.
    - As URLs não são verificadas, ou seja, podem nem existir de fato. São utilizadas apenas sintaticamente e como um padrão.

# *Namespace*

```
<execucao xmlns:programacao="sistema:debug:programa" >
        <programacao:codigo>
                ... if (x >= 90) { return 1; } ...
        </programacao:codigo>
</execucao>
```

Cadeia qualquer de caracteres

```
<venda xmlns:produto="http://www.empresax.com/produtos" >
        <produto:codigo>
                93253
        </produto:codigo>
</venda>
```

URL

# DTD

- Uma DTD (*Document Type Definition*) serve como uma gramática para o documento XML correspondente.

```
<db>
<person>
   <name> Alan </name>
   <age> 42 </age>
   <email> agb@abc.com </email>
</person>
<person>  ...  </person>
      ...
</db>
```

```
<!DOCTYPE db [
  <!ELEMENT db (person*)>
  <!ELEMENT person (name,age,
                    email+)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT age (#PCDATA)>
  <!ELEMENT email (#PCDATA)>
]>
```

Equivalente a um esquema de banco de dados

# DTD

- Declaração de atributos em uma DTD:

```
<product>
        <name language = "French" department = "Music">
        trompete six trous </name>
        <price currency = "Euro"> 420.12 </price>
</product>
```

```
<!ATTLIS name  language CDATA # REQUIRED
                department CDATA # IMPLIED>
<!ATTLIS price currency CDATA # IMPLIED>
```

# DTD

- Declaração de ID, IDREF e IDREFS em uma DTD:

```
<family>
    <person  id="jane" mother="mary" father="john">
        <name> Jane Doe </name>
    </person>
    <person id="john" children="jane jack">
        <name> John Doe </name>
    </person>
    <person id="mary" children="jane  jack">
        <name> Mary Doe </name>
    </person>
    <person  id="jack" mother=”mary" father="john">
        <name> Jack Doe </name>
    </person>
</family>
```

# DTD

- Declaração de ID, IDREF e IDREFS em uma DTD:

```
<!DOCTYPE family [
        <!ELEMENT family (person)*>
        <!ELEMENT person (name)>
        <!ELEMENT name (#PCDATA)>
        <!ATTLIST person id  ID #REQUIRED
                         mother  IDREF #IMPLIED
                         father  IDREF #IMPLIED
                         children  IDREFS #IMPLIED
]>
```

# Documentos XML válidos

- Um documento XML é considerado válido se, além de bem formado, possuir uma DTD e estiver estruturado de acordo com ela.
  - Quanto a referências, necessita-se apenas que os valores de atributos do tipo ID sejam únicos e que as referências dos tipos IDREF e IDREFS sejam para identificadores ID existentes.

- Outras propostas para representação de esquemas são:
  - DCD (*Document Content Description*);
  - XDR (*XML-Data Reduced*);
  - SOX (*Schema for Object-Oriented XML*);
  - *XML Schema*.

# Linguagens de Consulta

- Algumas linguagens de consulta são:
  - WebSQL (para dados da *Web*);
  - Lorel (para dados semi-estruturados);
  - XML-QL (para dados XML);
  - XQuery (para dados XML).

# XQuery: Exemplo

```
<empregados>
        ...
        <empregado cod="1000" dept="D01">
                <nome> João </nome>
                <sobrenome> Silva </sobrenome>
        </empregado>
        <empregado cod="1001" dept="D01">
                <nome> Ana </nome>
                <sobrenome> Oliveira </sobrenome>
        </empregado>
        ...
</empregados>
```

# XQuery: Exemplo

```
<emp-dept>
{
    for $e in doc('emps.xml')//empregado
    where $e/@dept= 'D01'
    order by $e/nome
    return $e/nome
}
</emp-dept>
```

Consulta em XQuery, simular a SQL

Retorna, como resultado, um elemento <emp-dept>, que não existe no documento XML de origem

```
<emp-dept>
    <nome>Ana</nome>
    <nome>João</nome>
</emp-dept>
```

# Armazenamento de Documentos XML

- Para armazenar um documento XML em um banco de dados, pode-se utilizar um SGBD para armazenar:
  - o documento XML como texto;
  - o conteúdo (elementos) do documento XML como registros.

# Armazenamento de Documentos XML

- Armazenamento do documento XML como texto.
  - Geralmente, tal forma é usada quando o documento XML não possui uma DTD.
  - Um SGBD relacional ou de objeto pode ser usado para armazenar documentos XML inteiros como campos de texto em tuplas ou objetos do esquema do banco de dados.

- Armazenamento do documento XML como registros.
  - Tal forma é usada quando o documento XML possui uma DTD.
  - Como o documento XML segue uma estrutura, deve-se projetar um banco de dados relacional ou de objeto para armazenar os elementos do mesmo.

# Extração de Documentos XML

- Para gerar um documento XML de um banco de dados, pode-se criar um documento XML personalizado a partir de um determinado banco de dados relacional, visando migração e exibição de dados via *Web*.
  - Tuplas de relações de bancos de dados relacionais podem ser formatadas como elementos de documentos XML.
  - Para tanto, um componente do SGBD é usado para tratar as conversões necessárias de tuplas de bancos de dados relacionais para elementos de documentos XML.

# XML versus JSON

- XML (*Extensible Markup Language*) e JSON (*JavaScript Object Notation*) são formatos úteis e populares para o armazenamento e intercâmbio de dados, apresentando semelhanças e diferenças (vantagens e desvantagens).
  - XML já existe há bastante tempo e é amplamente utilizada em ambientes empresariais, apresentando uma série de recursos que JSON não possui.
  - JSON é mais recente, mas está ganhando popularidade em organizações por apresentar uma sintaxe mais simples e por ser mais rápido para o intercâmbio de dados.

# XML versus JSON

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<students>
    <student>
        <id>01</id>
        <name>Tom</name>
        <lastname>Price</lastname>
    </student>
    <student>
        <id>02</id>
        <name>Nick</name>
        <lastname>Thameson</lastname>
    </student>
</students>
```

Documento XML

```json
{
 "student": [
    {
      "id":"01",
      "name": "Tom",
      "lastname": "Price"
    },
    {
      "id":"02",
      "name": "Nick",
      "lastname": "Thameson"
    }
 ]
}
```

Documento JSON equivalente

# O que é JSON?

- JSON é um formato de arquivo que usa texto legível e organizado para armazenar e transmitir conjuntos de dados representados por pares e matrizes atributo-valor.
  - É possível ler seu conteúdo sem a necessidade de um analisador, facilitando o armazenamento de dados advindos de outras aplicações.
  - É independente da linguagem de programação, tornando-a ideal para utilização no desenvolvimento *Web*, já que poderá necessitar de intercâmbio simultâneo de dados com distintas linguagens de programação como *Ruby* ou *JavaScript*.
  - Possibilita uma serialização (conversão) simples de dados complexos numa única cadeia, sendo um formato ideal para a partilha de dados entre APIs e aplicações *Web*.

# XML versus JSON: Semelhanças

- XML e JSON são formatos autodescritivos, baseados em textos e com estruturas hierárquicas.
  - Formatos autodescritivos foram concebidos para serem legíveis e escritos por seres humanos (fácil compreensão) e por máquinas (regras de formatação).
- XML e JSON apresentam um bom suporte para definição e validação de conteúdos.
  - Diversas linguagens de programação suportam JSON (*JavaScript*, *Python*, *Perl*, *Ruby*) e XML (*JavaScript*, PHP, C#).
- XML e JSON serializam dados, ou seja, convertem dados para um formato de armazenamento e transmissão.
  - Dados podem ser transmitidos de uma aplicação para outra por meio de um canal de comunicação (como HTTP) ou por uma API.

# JSON: Principais Vantagens

- Fornece suporte para distintos navegadores;
- Possui um sintaxe simples, facilitando leitura e escrita;
- É mais rápida;
- É reconhecida nativamente por *JavaScript*, podendo ser facilmente analisada (função *eval*) e suportada por suas principais estruturas;
- É suportada por distintas tecnologias de *back-end* e linguagens de programação modernas;
- Permite serializar e transmitir conjuntos de dados estruturados usando conexão de rede.

# JSON: Principais Desvantagens

- O objeto JSON possui um tipo;
- Não possui suporte para *namespace*, apresentando assim pouca extensibilidade;
- Não possui recursos de exibição de dados;
- É menos segura.

# XML: Quando usar?

- Segundo um estudo do Instituto Nacional de Normas e Tecnologia (NIST), que utilizou conjuntos de dados enviados por meio de canais inseguros, XML é mais segura do que JSON, embora ambos sejam vulneráveis a ataques.
- Por verificar erros mais eficientemente que a JSON, XML deve ser usada para tratar conjuntos de dados que exijam mais espaço que simples cadeias de caracteres (planilhas envolvendo milhares de linhas, colunas e valores por linha).

➢ XML é frequentemente usada para tratar conjuntos de dados que precisam de mais confiabilidade, segurança e estabilidade.

- Ex.: transações financeiras e bancárias.

# JSON: Quando usar?

- JSON é mais rápida já que possui uma sintaxe mais simples para editar novos documentos, tornando os erros de depuração nos seus dados mais acessíveis, e requer menos memória para armazenar a mesma quantidade de dados.
- JSON é mais flexível já que pode ser utilizada em muitas e distintas linguagens de programação.

➢ JSON é frequentemente usada para a construção de páginas, o armazenamento de dados e o intercâmbio de dados entre servidores e/ou dispositivos da *Web*, por ser mais leve (simples) e reconhecida por distintos navegadores, quando não se precisa dos recursos da XML.

# XML versus JSON

- A escolha entre XML e JSON depende das necessidades.
  - Grandes quantidades de dados: XML poderá ser melhor.
  - Comunicação com outra aplicação: JSON poderá ser melhor.
  - O adequado é pesquisar os prós e os contras de cada formato e decidir o que funciona melhor para o que se deseja.