

Programação Orientada a Objetos

Objetos, Dados e Serialização

André Santanchè

Laboratory of Information Systems - LIS

Instituto de Computação - UNICAMP

Maio 2015

Serialização

- Transformação do estado de um objeto em um formato de dados que possa ser armazenado ou transmitido
- Deserialização - processo inverso

Java

Interface `java.io.Serializable`

- Implementada por objetos que podem ser serializados
- Não define métodos
 - funciona como marcação
- Serialização padrão
 - feita na forma de reflexão
- Serialização customizada
 - Devem ser implementados métodos `writeObject`, `readObject` e `readObjectNoData`

Serializando e Deserializando Objetos

Formato Binário

- `ObjectOutputStream` → serialização
- `ObjectInputStream` → deserialização

The background of the slide is a close-up photograph of a green leaf with several clear water droplets. The leaf's veins are visible, and the droplets are in various stages of focus, some sharp and some blurred. The overall lighting is soft and natural, highlighting the texture of the leaf and the clarity of the water.

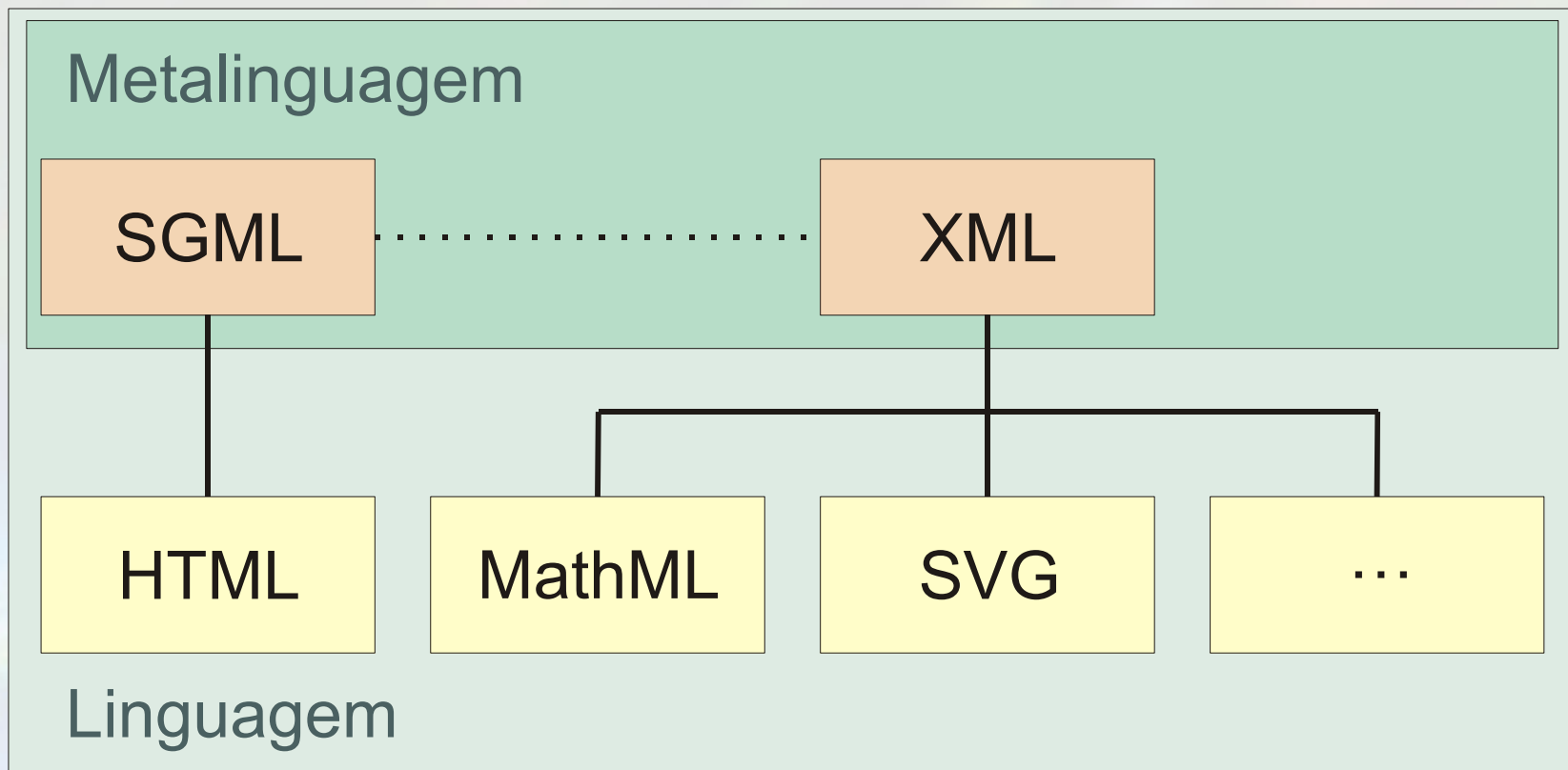
XML - eXtensible Markup Language

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.

XML e Objetos

- A estrutura hierárquica do XML combina com a estrutura hierárquica dos Objetos

Serializando e Deserializando Objetos Formato XML

- Formato XML
 - XMLEncoder → serialização
 - XMLDecoder → deserialização



JSON

JavaScript Object Notation

JSON

- Padrão aberto de intercâmbio de objetos
- Baseado na notação JavaScript
- Incorporado ao ECMAScript (Ecma, 2011)
- Adotado por diversas linguagens (<http://json.org/>)

The background of the slide is a close-up photograph of a green leaf with several clear water droplets. The image is slightly blurred and has a soft, ethereal quality. The text is centered over this background.

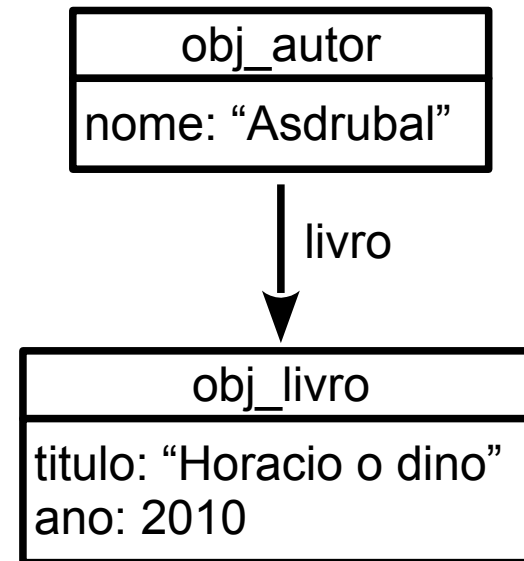
Notação Inline de Objetos JavaScript

Objetos JS

<pre>{ }</pre>	<table border="1"><tr><td>vazio</td></tr></table>	vazio					
vazio							
<pre>{ "nome": "Asdrubal", "idade": 25 }</pre>	<table border="1"><tr><td>obj_pessoa</td></tr><tr><td>nome: "Asdrubal" idade: 25</td></tr></table>	obj_pessoa	nome: "Asdrubal" idade: 25				
obj_pessoa							
nome: "Asdrubal" idade: 25							
<pre>{ "nome": "Unidos da Esquina", "vitorias": [1961, 1975, 1982] }</pre>	<table border="1"><tr><td>obj_time</td></tr><tr><td>nome: "Unidos da Esquina"</td></tr></table> <p style="text-align: center;">↓ vitorias</p> <table border="1"><tr><td>obj_vitorias: Array</td></tr><tr><td>0: 1961</td></tr><tr><td>1: 1975</td></tr><tr><td>2: 1982</td></tr></table>	obj_time	nome: "Unidos da Esquina"	obj_vitorias: Array	0: 1961	1: 1975	2: 1982
obj_time							
nome: "Unidos da Esquina"							
obj_vitorias: Array							
0: 1961							
1: 1975							
2: 1982							

Objetos JS

```
{  
  "nome": "Asdrubal",  
  "livro": {  
    "titulo": "Horacio o dino",  
    "ano": 2010  
  }  
}
```



Stringify

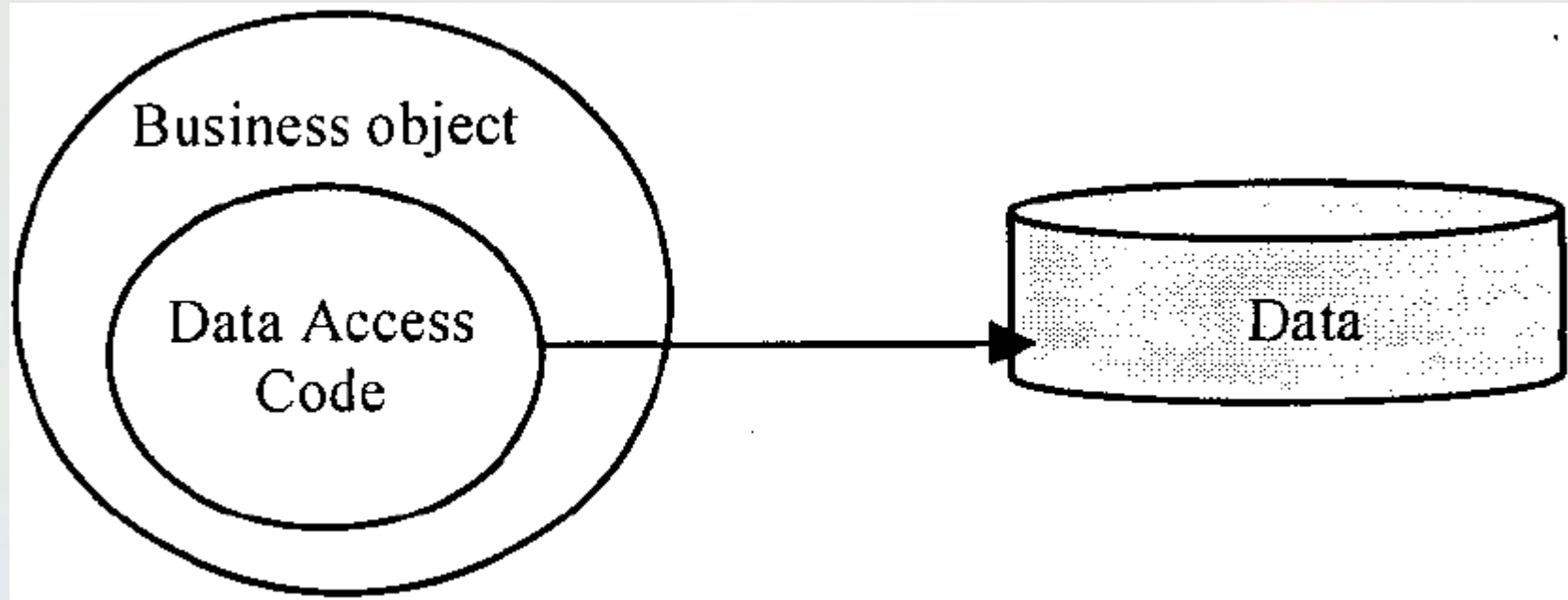
- Serializando

```
var pessoa = {  
  "nome": "Asdrubal",  
  "idade": 25  
};  
var pessoaStr = JSON.stringify(pessoa);
```

- Deserializando

```
var pessoa2 = JSON.parse(pessoaStr);
```

Data Access Object (DAO) Pattern



(Matic, 2004)



Armazenando em Bancos de Dados

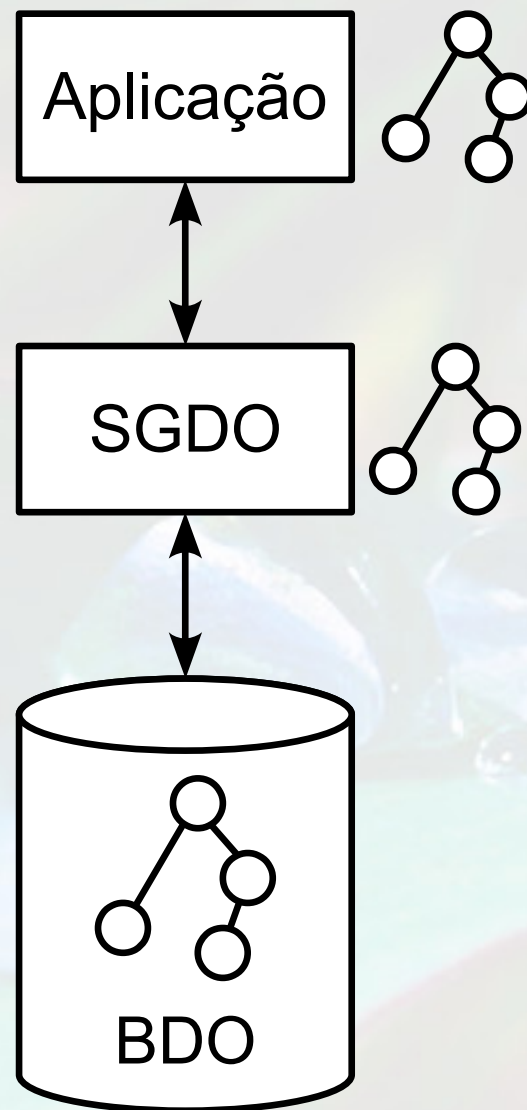
BDO

Bancos de Dados de Objeto

- Anteriormente conhecidos como BDOO
- “Pode estender a existência de objetos de modo que eles sejam armazenados permanentemente em um banco de dados, e, portanto, os objetos se tornam objetos persistentes...”

(Elmasri, 2011)

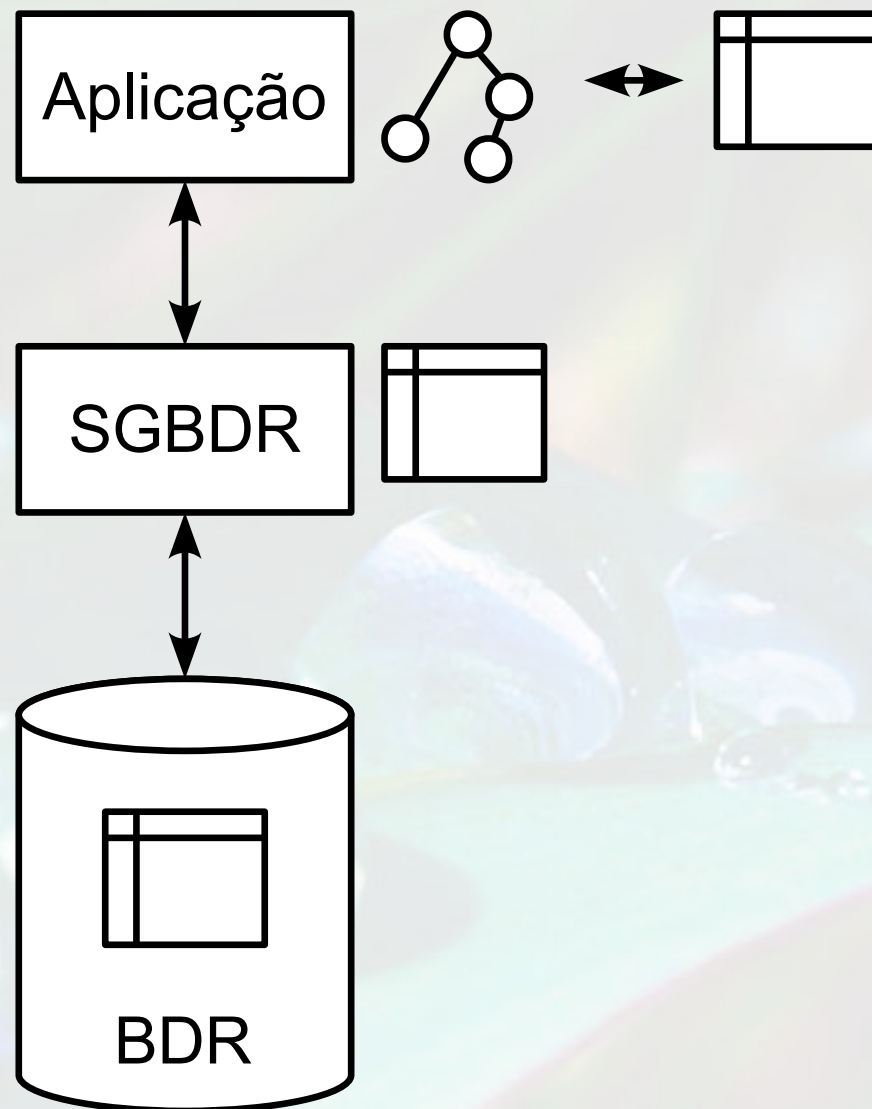
SGDO & BDO



SGDOs

- 02 - clássico BDO
- db4objects (<http://www.db4o.com>) - Versant
- Objectivity/DB (<http://www.objectivity.com>)

Aplicações OO x BD Relacionais



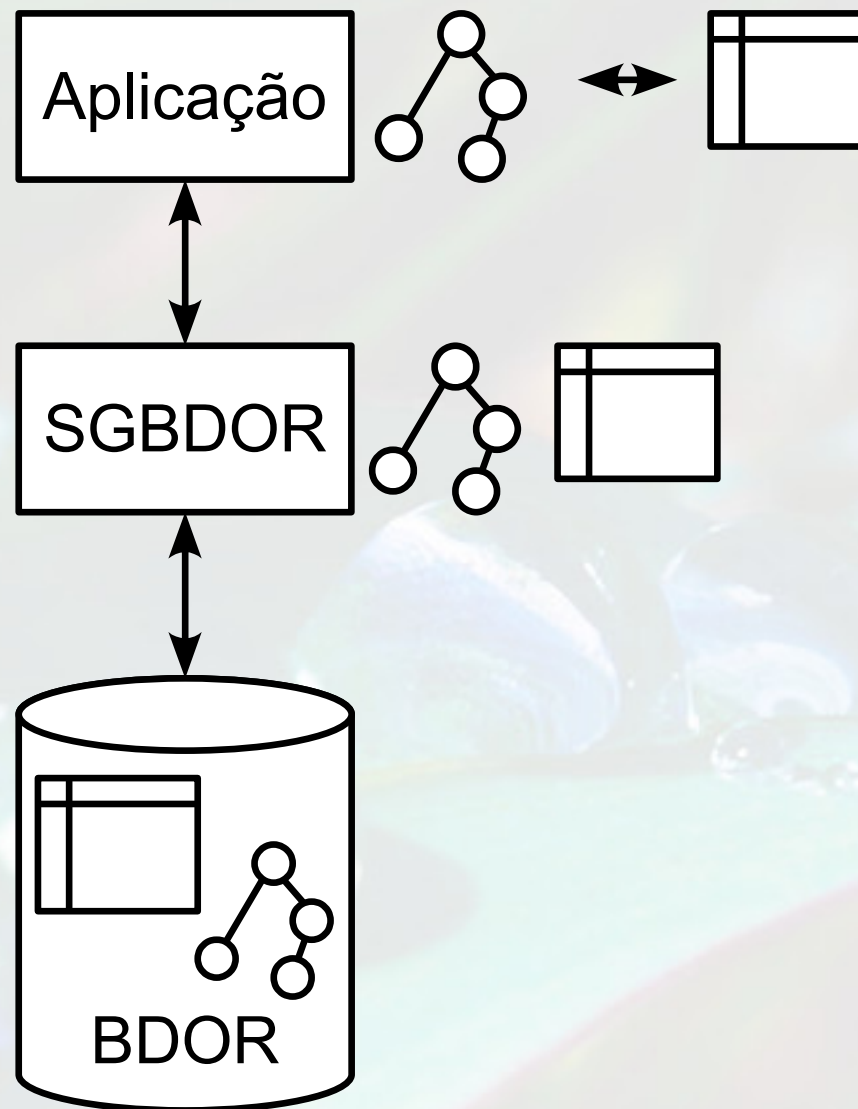
SGBDOR

SGBD Objeto-Relacional

- Extensão em SGBDRs para suportar objetos
- Extensão do SQL para objetos
 - Originalmente introduzida no SQL:1999
 - Atualizados no SQL:2008

(Elmasri, 2011)

SGBDOR & BDOR



Document Databases

- XML-based
 - BaseX (<http://basex.org>)
- JSON
 - CouchDB (<http://couchdb.apache.org>)
 - Mongo DB (<http://www.mongodb.org>)



Key-value

Web Storage



- Cookies
 - tem sido o principal mecanismo de armazenamento
- W3C Web Storage
 - modelo “mínimo” de armazenamento
 - baseado em (chave, valor)

(Hickson, 2013)

Web Storage API

<code>setItem(chave, valor)</code>	adiciona/atualiza par chave-valor
<code>getItem(chave)</code>	recupera o valor associado à chave
<code>key(n)</code>	recupera a enésima chave
<code>removeItem(chave)</code>	remove o par que possui a chave
<code>length</code>	indica o número de pares chave-valor
<code>clear()</code>	remove todos os dados do repositório

Implementações da API

- `sessionStorage`
 - persistência apenas durante uma sessão
- `localStorage`
 - persistência a longo prazo

Exemplo

- Gravando o campo HTML:

Nome: `<input type="text" id="nome"></input>`

- Funções de leitura/gravação

```
function ler() {  
    var nomeLido = localStorage.getItem("nome_db");  
    if (nomeLido != null)  
        document.querySelector("#nome").value = nomeLido;  
}
```

```
function gravar() {  
    var nomeGravar = document.querySelector("#nome").value;  
    localStorage.setItem("nome_db", nomeGravar);  
}
```



Amazon DynamoDB

Referências

- Ecma International (2011). ECMAScript Language Specification - Standard ECMA-262 (5.1 ed.).
- Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. 2008. Bigtable: A Distributed Storage System for Structured Data. ACM Trans. Comput. Syst. 26, 2, Article 4 (June 2008).
- Hickson, I. (2011). HTML Microdata -- W3C Working Draft 13 January 2011. W3C. Retrieved from <http://www.w3.org/TR/2011/WD-microdata-20110113/>

André Santanchè

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

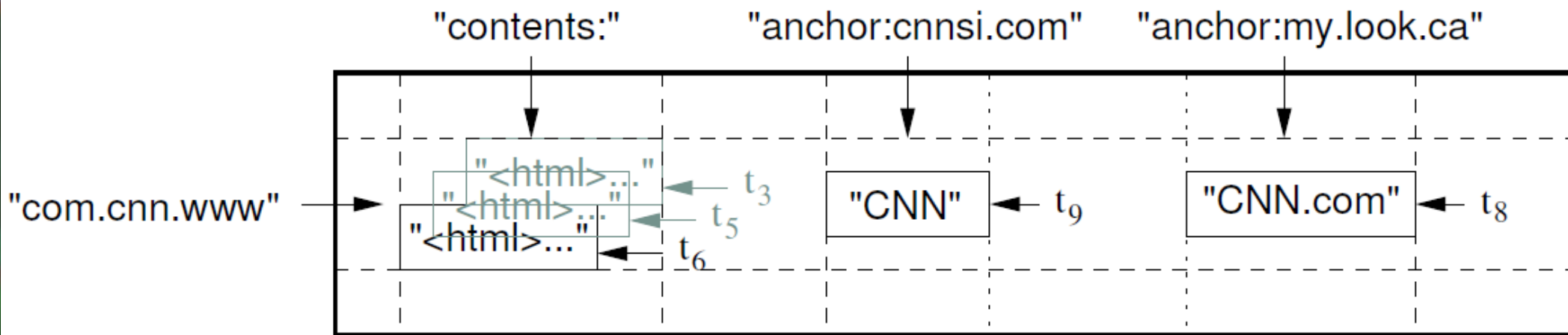
Licença

- Estes slides são concedidos sob uma Licença Creative Commons. Sob as seguintes condições: Atribuição, Uso Não-Comercial e Compartilhamento pela mesma Licença.
- Mais detalhes sobre a referida licença Creative Commons veja no link:
<http://creativecommons.org/licenses/by-nc-sa/3.0/>
- Agradecimento a Moyan Brenn [http://www.flickr.com/photos/aigle_dore/] por sua fotografia “Dew drops” usada na capa e nos fundos, disponível em [http://www.flickr.com/photos/aigle_dore/6225536653/] vide licença específica da fotografia.



Google Bigtable

Bigtable Model



(Fay et al., 2008)

Tablets & Hierarchy

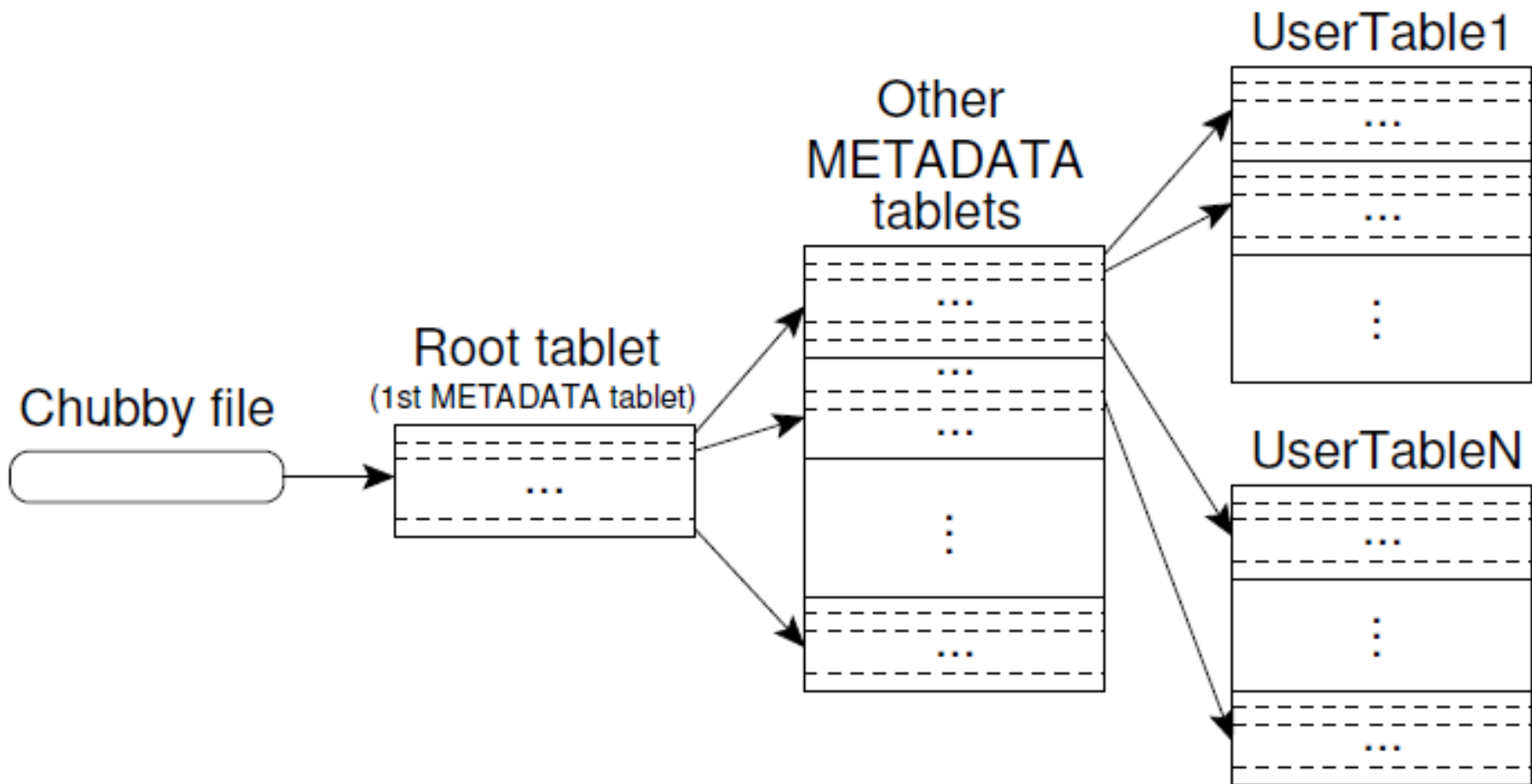


Figure 4: Tablet location hierarchy.

(Fay et al., 2008)