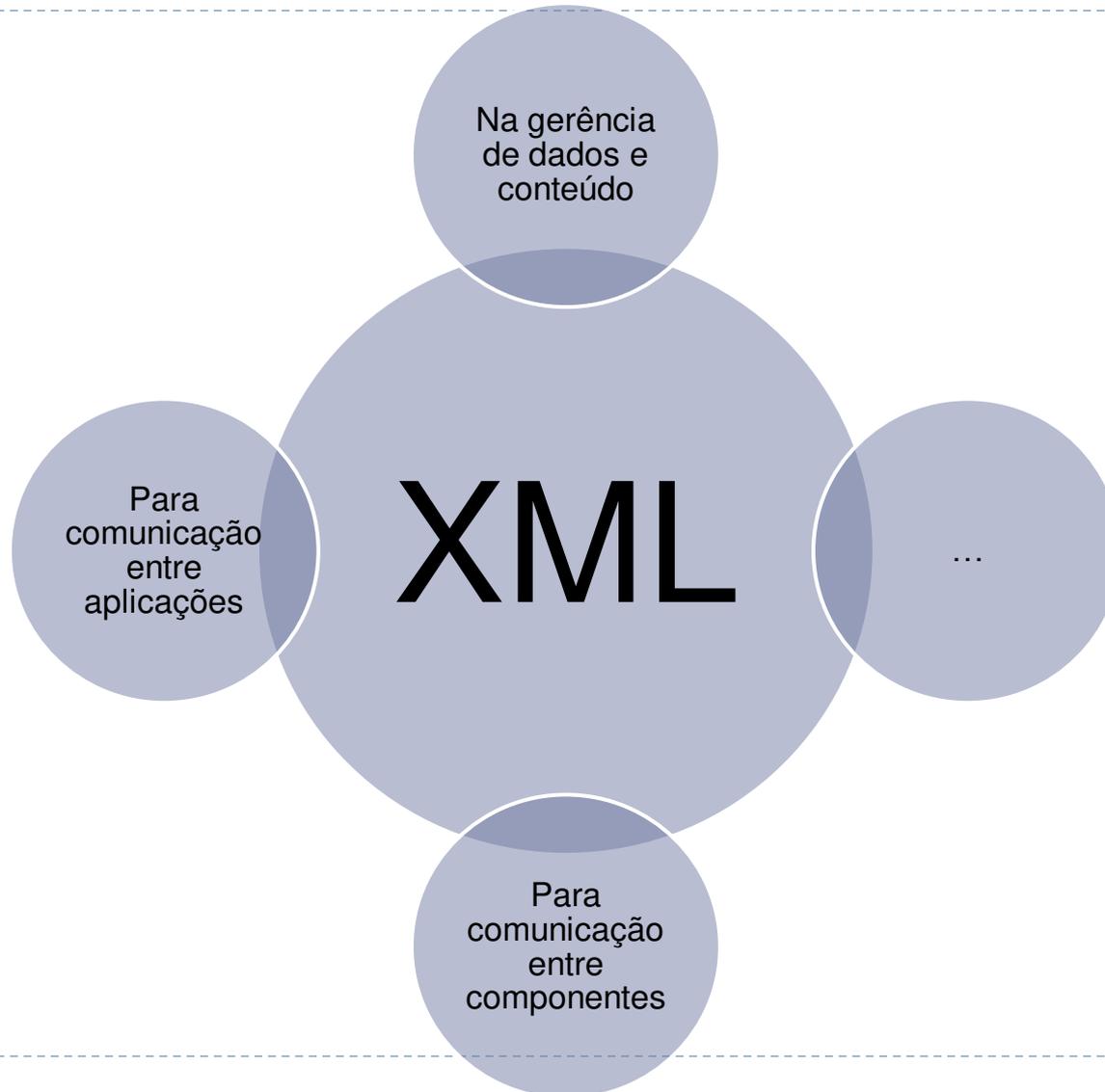


Armazenamento de Documentos XML em SGBDs Objeto-Relacionais

Vanessa Braganholo

XML em diferentes contextos



Como armazenar?

1. Sistema de arquivos
2. SGBD Relacional/Objeto Relacional
 1. Mapeamento “manual”
 2. Com suporte a XML
 1. Habilitado a XML
 2. Híbrido
3. Banco de Dados XML Nativo



1 – Sistema de arquivos

- ▶ Vantagens

- ▶ Flexível

- ▶ Desvantagens

- ▶ Segurança
- ▶ Impossibilidade de otimização de consultas (ausência de índices, etc.)



2 – Banco de Dados com suporte a XML

▶ Alternativas:

- ▶ Fazer mapeamento “na mão”
 - ▶ Genérico ou
 - ▶ Específico para uma DTD ou XML Schema
- ▶ Utilizar um banco de dados Habilitado a XML
 - ▶ SGBDs com extensões para transferir dados entre documentos XML e suas estruturas.
- ▶ Utilizar um banco de dados híbrido
 - ▶ SGBD relacional/objeto-relacional que possui suporte a armazenamento XML nativo



3 – Banco de Dados Nativo

- ▶ SGBDs que armazenam XML em sua forma nativa, geralmente como texto indexado ou como uma variante do DOM mapeado para uma estrutura proprietária



Fazendo o mapeamento “na mão”...

O princípio...

- ▶ Proposta publicada em 1993 (ABITEBOUL; CLUET; MILO; 1993) intitulada “**Querying and Updating the file**” sugeria usar a tecnologia relacional para consultar arquivos textuais
 - ▶ Para isso, seria necessário armazenar tais arquivos em BDs relacionais
 - ▶ Idéia: explorar a **estrutura intrínseca** de arquivos tais como arquivos SGML, código fonte, etc., para armazená-los no BD



XML

- ▶ XML possui tal estrutura intrínseca, e portanto poderia se beneficiar das idéias lançadas em 93
- ▶ Várias propostas específicas para armazenamento de XML surgiram ao longo dos anos:
 - ▶ (FLORESCU; KOSSMANN, 1999)
 - ▶ (DEUTSCH; FERNANDEZ; SUCIU, 1999)
 - ▶ (SHANMUGASUNDARAM et al., 1999)
 - ▶ (LEE; CHU, 2000)
 - ▶ (CHEN; DAVIDSON; ZHENG, 2002, 2003)



Consultas

- ▶ Mas não basta só armazenar os docs XML
- ▶ É necessário também poder consultá-los
- ▶ Propostas que exploram este problema são:
 - ▶ (SHANMUGASUNDARAM et al., 1999)
 - ▶ (MANOLESCU; FLORESCU;KOSSMANN, 2001)
 - ▶ (SHANMUGASUNDARAM et al., 2001)
 - ▶ (TATARINOV et al., 2002)
 - ▶ (DEHAAN et al., 2003)



Tipos de proposta

▶ Armazenamento:

- ▶ Técnicas que exploram a estrutura do XML (elementos, atributos, relação pai-filho)
 - ▶ (FLORESCU; KOSSMANN, 1999)
 - ▶ (DEHAAN et al., 2003)
 - ▶ (TATARINOV et al., 2002)
- ▶ Técnicas que exploram o esquema do doc. XML (DTD ou XML Schema)
 - ▶ (SHANMUGASUNDARAM et al., 1999)
- ▶ Técnicas que exploram alguma relação semântica entre os dados (ex. dependências funcionais)
 - ▶ (CHEN; DAVIDSON; ZHENG, 2002, 2003)
 - ▶ (LEE; CHU, 2000)



Técnicas que exploram a estrutura
do modelo XML

Documentos a serem armazenados

(FLORESCU; KOSSMANN, 1999)

```
<person id=1, age=55>
  <name>Peter</name>
  <address>4711 Fruitdale Ave.</address>
  <child>
    <person id=3, age=22>
      <name>John</name>
      <address>5361 Columbia Ave.</address>
      <hobby>swimming</hobby>
      <hobby>cycling</hobby>
    </person>
  </child>
  <child>
    <person id=4, age=7>
      <name>David</name>
      <address>4711 Fruitdale Ave.</address>
    </person>
  </child>
</person>
```

```
<person id=2, age=38, child=4>
  <name>Mary</name>
  <address>4711 Fruitdale Ave.</address>
  <hobby>painting</hobby>
</person>
```

FLORESCU; KOSSMANN, 1999

Proposta *Edge* (Aresta)

- ▶ Armazenar todos os documentos em uma única tabela chamada *Edge*

- ▶ Edge(source, ordinal, name, flag, target)

Nome do elemento
ou atributo

Numero para
preservar a ordem
entre os elementos
de um mesmo
documento

id que indica o
documento XML



FLORESCU; KOSSMANN, 1999

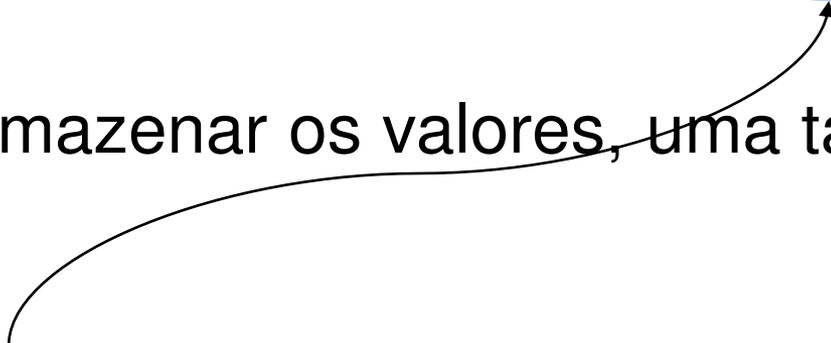
Proposta *Edge* (Aresta)

- ▶ Armazenar todos os documentos em uma única tabela chamada *Edge*
 - ▶ $\text{Edge}(\underline{\text{source}}, \underline{\text{ordinal}}, \text{name}, \text{flag}, \text{target})$
- ▶ Para armazenar os valores, uma tabela V para cada tipo:
 - ▶ $V_{\text{type}}(\underline{\text{vid}}, \text{value})$



FLORESCU; KOSSMANN, 1999

Proposta *Edge* (Aresta)

- ▶ Armazenar todos os documentos em uma única tabela chamada *Edge*
 - ▶ $\text{Edge}(\underline{\text{source}}, \underline{\text{ordinal}}, \text{name}, \text{flag}, \text{target})$
 - ▶ Para armazenar os valores, uma tabela V para cada tipo:
 - ▶ $V_{\text{type}}(\underline{\text{vid}}, \text{value})$
- 



FLORESCU; KOSSMANN, 1999

Edge

source	ordinal	name	flag	target
1	1	age	int	v1
1	2	name	string	v2
1	3	address	string	v3
1	4	child	ref	3
1	5	child	ref	4
2	1	age	int	v4
...

V_{int}

vid	value
v1	55
v4	38
v8	22
v13	7

V_{string}

vid	value
v2	Peter
v3	4711 Fruitdale Ave.
v5	Mary
v6	4711 Fruitdale Ave.
v7	painting
...	...
v15	4711 Fruitdale Ave.

```

(person id=1, age=55)
  (name)Peter(/name)
  (address)4711 Fruitdale Ave.(/address)
  (child)
    (person id=3, age=22)
      (name)John(/name)
      (address)5361 Columbia Ave.(/address)
      (hobby)swimming(/hobby)
      (hobby)cycling(/hobby)
    (/person)
  (/child)
  (child)
    (person id=4, age=7)
      (name)David(/name)
      (address)4711 Fruitdale Ave.(/address)
    (/person)
  (/child)
(/person)

(person id=2, age=38, child=4)
  (name)Mary(/name)
  (address)4711 Fruitdale Ave.(/address)
  (hobby)painting(/hobby)
(/person)

```



FLORESCU; KOSSMANN, 1999

Edge	source	ordinal	name	flag	target
	1	1	age	int	v1
	1	2	name	string	v2
	1	3	address	string	v3
	1	4	child	ref	3
	1	5	child	ref	4
	2	1	age	int	v4

V_{int}

vid	value
v1	55
v4	38
v8	22
v13	7

V_{string}

vid	value
v2	Peter
v3	4711 Fruitdale Ave.
v5	Mary
v6	4711 Fruitdale Ave.
v7	painting
...	...
v15	4711 Fruitdale Ave.

```

(person id=1, age=55)
  (name)Peter(/name)
  (address)4711 Fruitdale Ave./address)
  (child)
    (person id=3, age=22)
      (name)John(/name)
      (address)5361 Columbia Ave./address)
      (hobby)swimming(/hobby)
      (hobby)cycling(/hobby)
    (/person)
  (/child)
  (child)
    (person id=4, age=7)
      (name)David(/name)
      (address)4711 Fruitdale Ave./address)
    (/person)
  (/child)
(/person)

(person id=2, age=38, child=4)
  (name)Mary(/name)
  (address)4711 Fruitdale Ave./address)
  (hobby)painting(/hobby)
(/person)
  
```



FLORESCU; KOSSMANN, 1999

Elemento Complexo:
valor armazenado
na própria tabela
Edge

Edge				
source	ordinal	name	flag	target
1	1	age	int	v1
1	2	name	string	v2
1	3	address	string	v3
1	4	child	ref	3
1	5	child	ref	4
2	1	age	int	v4
...

```

<person id=1, age=55>
  <name>Peter</name>
  <address>4711 Fruitdale Ave.</address>
  <child>
    <person id=3, age=22>
      <name>John</name>
      <address>5361 Columbia Ave.</address>
      <hobby>swimming</hobby>
      <hobby>cycling</hobby>
    </person>
  </child>
  <child>
    <person id=4, age=7>
      <name>David</name>
      <address>4711 Fruitdale Ave.</address>
    </person>
  </child>
</person>

<person id=2, age=38, child=4>
  <name>Mary</name>
  <address>4711 Fruitdale Ave.</address>
  <hobby>painting</hobby>
</person>
  
```

V _{int}	
vid	value
v1	55
v4	38
v8	22
v13	7

V _{string}	
vid	value
v2	Peter
v3	4711 Fruitdale Ave.
v5	Mary
v6	4711 Fruitdale Ave.
v7	painting
...	...
v15	4711 Fruitdale Ave.



FLORESCU; KOSSMANN, 1999

- ▶ Propõem também várias variações
- ▶ A mais usada é chamada de **inlining...**
 - ▶ Armazenar tudo em uma única tabela
 - ▶ Duas possibilidades:
 - ▶ Uma coluna para cada tipo de valor
Edge (source, ordinal, name, v_{string} , v_{int} , ..., target)
 - ▶ Uma coluna única para todos os tipos de valores (todos os valores seriam convertidos para string)
Edge (source, ordinal, name, v, target)



FLORESCU; KOSSMANN, 1999

Processamento de consultas

▶ Reconstrução do documento:

Tabela Edge: junção com tabelas V, seleção pelo *source*, ordenar pelo *ordinal*

Tabela Inlinning: não há necessidade de junção, seleção pelo *source*, ordenar pelo *ordinal*



FLORESCU; KOSSMANN, 1999

Processamento de consultas

- ▶ Consultas com seleção (ex. selecionar todos os elementos `hobby="swimming"`)

Tabela Edge: junção com tabelas *V*, seleção pelo *source* e por *value="swimming"*

Tabela Inlining: não há necessidade de junção, seleção pelo *source* e por *value="swimming"*



Dynamic Interval

(DEHAAN et al., 2003)

- ▶ Relação muito simples que guarda o elemento e a codificação do intervalo que o engloba

```
<site>
<people>
  <person id="person0">
    <name>Jaak Tempesti</name>
    <emailaddress>mailto:Tempesti@labs.com</emailaddress>
    <phone>+0 (873) 14873867</phone>
    <homepage>http://www.labs.com/~Tempesti</homepage>
  </person>
  <person id="person1">
    <name>Cong Rosca</name>
    <emailaddress>mailto:Rosca@washington.edu</emailaddress>
    <phone>+0 (64) 27711230</phone>
    <homepage>http://www.washington.edu/~Rosca</homepage>
  </person>
  ...
</people>
<closed_auctions>
  <closed_auction>
    <seller person="person0" />
    <buyer person="person1" />
    <itemref item="item1" />
    <price>42.12</price>
    <date>08/22/1999</date>
    <quantity>1</quantity>
    <type>Regular</type>
  </closed_auction>
  ...
</closed_auctions>
  ...
</site>
```

Dynamic Interval

(DEHAAN et al., 2003)

```
0 <site>
  <people>
    <person id="person0">
      <name>Jaak Tempesti</name>
      <emailaddress>mailto:Tempesti@labs.com</emailaddress>
      <phone>+0 (873) 14873867</phone>
      <homepage>http://www.labs.com/~Tempesti</homepage>
    </person>
    <person id="person1">
      <name>Cong Rosca</name>
      <emailaddress>mailto:Rosca@washington.edu</emailaddress>
      <phone>+0 (64) 27711230</phone>
      <homepage>http://www.washington.edu/~Rosca</homepage>
    </person>
    ...
  </people>
  <closed_auctions>
    <closed_auction>
      <seller person="person0" />
      <buyer person="person1" />
      <itemref item="item1" />
      <price>42.12</price>
      <date>08/22/1999</date>
      <quantity>1</quantity>
      <type>Regular</type>
    </closed_auction>
    ...
  </closed_auctions>
  ...
85 </site>
```

s	l	r
<site>	0	85
<people>	1	46
<person>	2	23
@id	3	6
person0	4	5
<name>	7	10
Jaak Tempesti	8	9
.	.	.
.	.	.
.	.	.

Dynamic Interval

(DEHAAN et al., 2003)

- ▶ Tradução das consultas – operações matemáticas sobre os intervalos

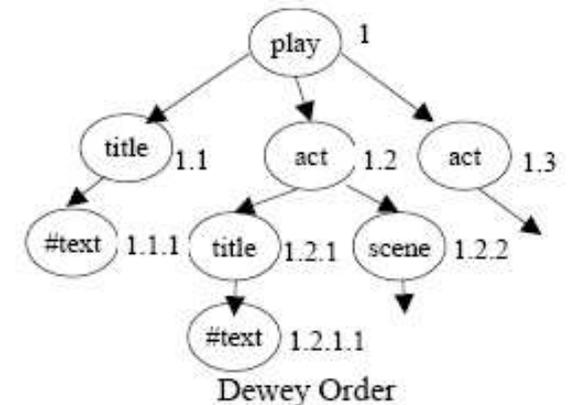
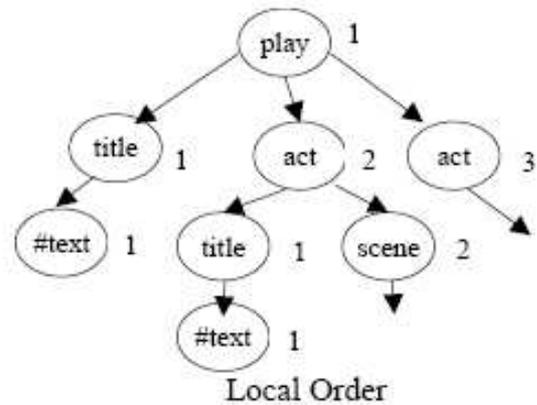
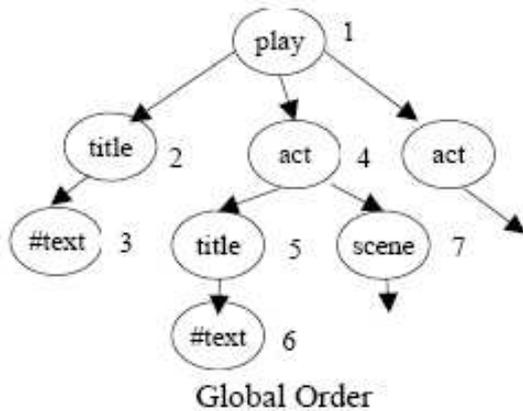
```
CREATE VIEW T_XNODE_item AS
  SELECT s, l+i*92 AS l, r+i*92 AS r
  FROM I,
    ( SELECT '<item>' AS s, 0 AS l, 91 AS r
      FROM UNIT
    UNION ALL
      SELECT s, l+i AS l, r+i AS r
      FROM
        ( SELECT s, l-i*90 AS l, r-i*90 AS r
          FROM T_e
          WHERE i*90<=l AND r<(i+1)*90
        )
    )
)
```



Opções para armazenar ordem

(TATARINOV et al., 2002)

- ▶ Global Order
- ▶ Local Order (irmãos)
- ▶ Dewey Order

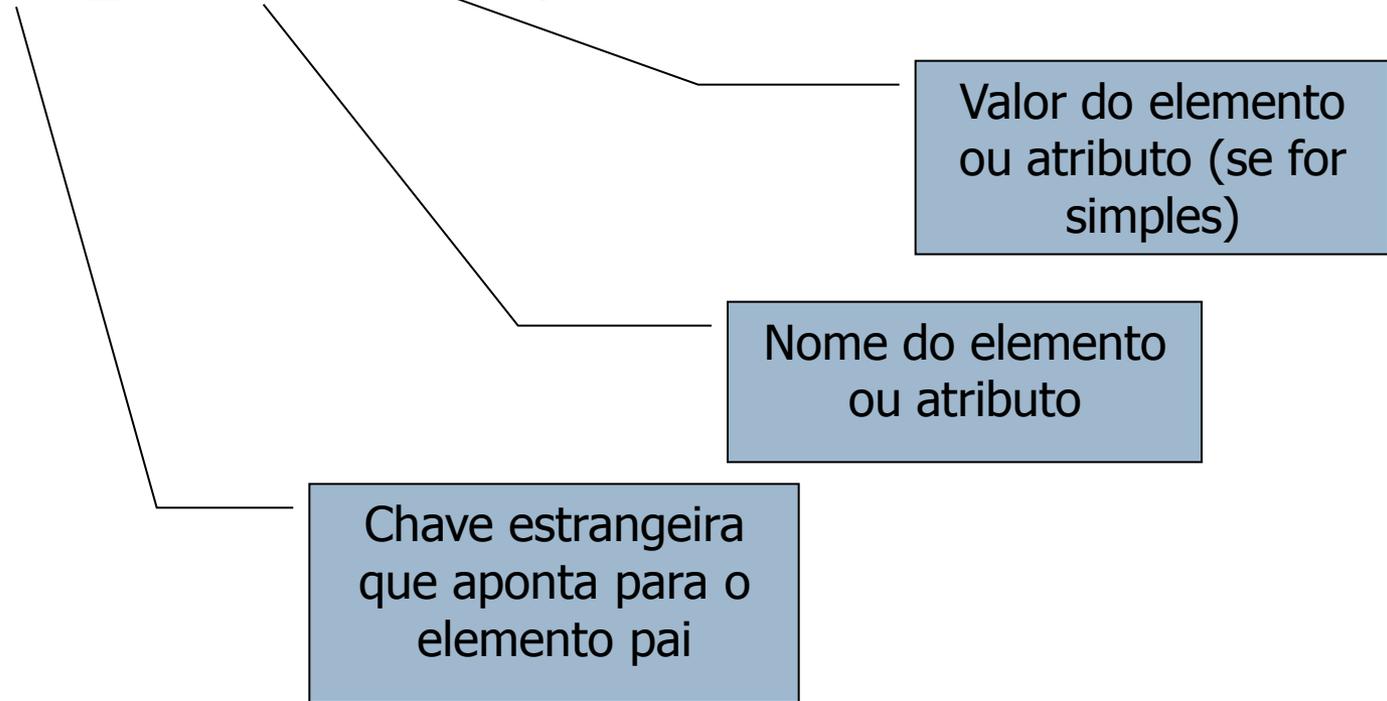


Proposta de Armazenamento

(TATARINOV et al., 2002)

- ▶ Variação do esquema Edge proposto por FLORESCU

Edge(id, parent_id, name, value)



Proposta de Armazenamento

(TATARINOV et al., 2002)

- ▶ Variação do esquema Edge proposto por FLORESCU

Edge(id, parent_id, name, value)

Ao invés do nome, o caminho do nodo pode ser armazenado (ex. /play/act ao invés de act)

Para poupar espaço, uma tabela Path pode ser usada:

Path(path_id, path)



Mas ainda falta a ordem...

(TATARINOV et al., 2002)

- ▶ **Global Order:**

Edge(id, parent_id, end_desc_id, path_id, value)

id – é o Global Order do nodo

end_desc_id – id do último descendente do nodo

- ▶ **Local Order:**

Edge(id, parent_id, sIndex, path_id, value)

id – um ID único (que não precisa seguir a ordem do doc.)

sIndex – Local Order do nodo

- ▶ **Dewey Order:**

Edge(dewey, path_id, value)



Atualização

(TATARINOV et al., 2002)

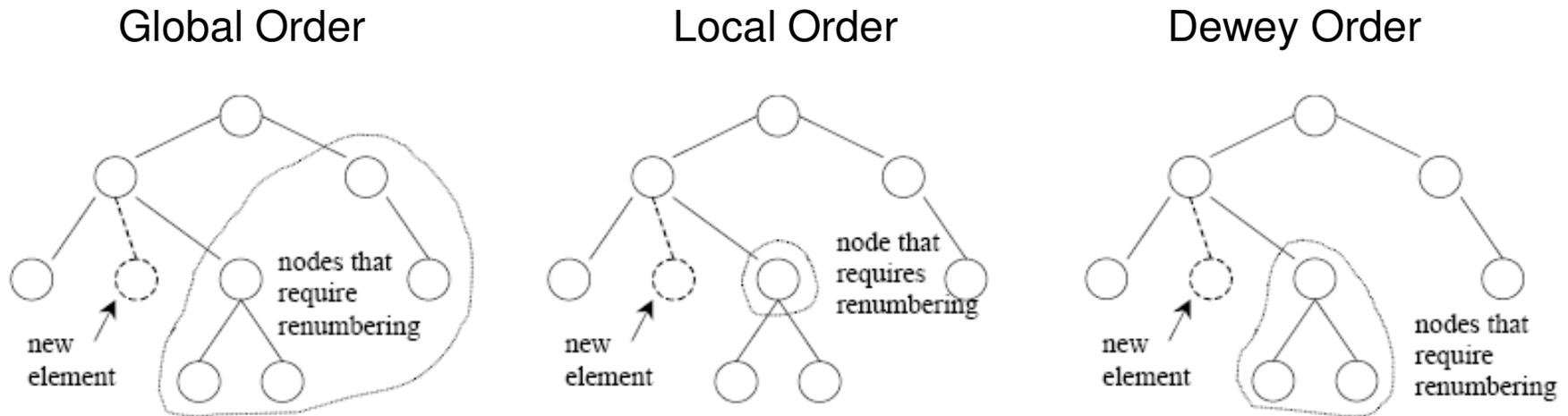


Figure 2. The worst case renumbering scenarios for Global, Local, and Dewey order encodings.



Consultas

(TATARINOV et al., 2002)

- ▶ Consultas suportadas: XPath





Técnicas que exploram o esquema
do documento XML



SHANMUGASUNDARAM et al., 1999

```
<!ELEMENT book (booktitle, author)
<!ELEMENT article (title, author*, contactauthor)>
<!ELEMENT contactauthor EMPTY>
<!ATTLIST contactauthor authorID IDREF IMPLIED>
<!ELEMENT monograph (title, author, editor)>
<!ELEMENT editor (monograph*)>
<!ATTLIST editor name CDATA #REQUIRED>
<!ELEMENT author (name, address)>
<!ATTLIST author id ID #REQUIRED>
<!ELEMENT name (firstname?, lastname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT address ANY>
```

```
<book>
  <booktitle> The Selfish Gene </booktitle>
  <author id = "dawkins">
    <name>
      <firstname> Richard </firstname>
      <lastname> Dawkins </lastname>
    </name>
    <address>
      <city> Timbuktu </city>
      <zip> 99999 </zip>
    </address>
  </author>
</book>
```



SHANMUGASUNDARAM et al., 1999

▶ Técnicas:

- ▶ Basic Inlining
- ▶ Shared Inlining
- ▶ Hybrid Inlining



Basic Inlining

SHANMUGASUNDARAM et al., 1999

▶ Basic Inlining

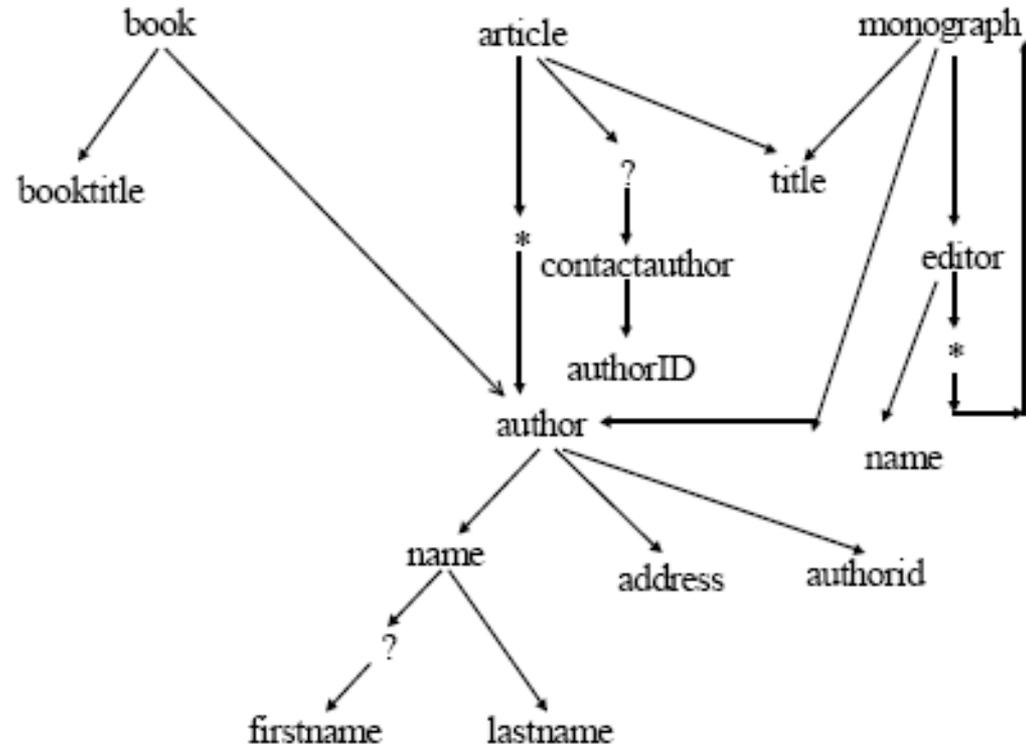
- ▶ Criar uma tabela para cada elemento da DTD, pq um documento XML pode usar como raiz qualquer um dos elementos de uma DTD (por isso declaramos a raiz em !DOCTYPE)
- ▶ Para lidar com elementos que se repetem (*), um grafo é construído a partir da DTD



Basic Inlining

SHANMUGASUNDARAM et al., 1999

```
<!ELEMENT book (booktitle, author)
<!ELEMENT article (title, author*, contactauthor)>
<!ELEMENT contactauthor EMPTY>
<!ATTLIST contactauthor authorID IDREF IMPLIED>
<!ELEMENT monograph (title, author, editor)>
<!ELEMENT editor (monograph*)>
<!ATTLIST editor name CDATA #REQUIRED>
<!ELEMENT author (name, address)
<!ATTLIST author id ID #REQUIRED>
<!ELEMENT name (firstname?, lastname)
<!ELEMENT firstname (#PCDATA)
<!ELEMENT lastname (#PCDATA)
<!ELEMENT address ANY>
```



Basic Inlining

SHANMUGASUNDARAM et al., 1999

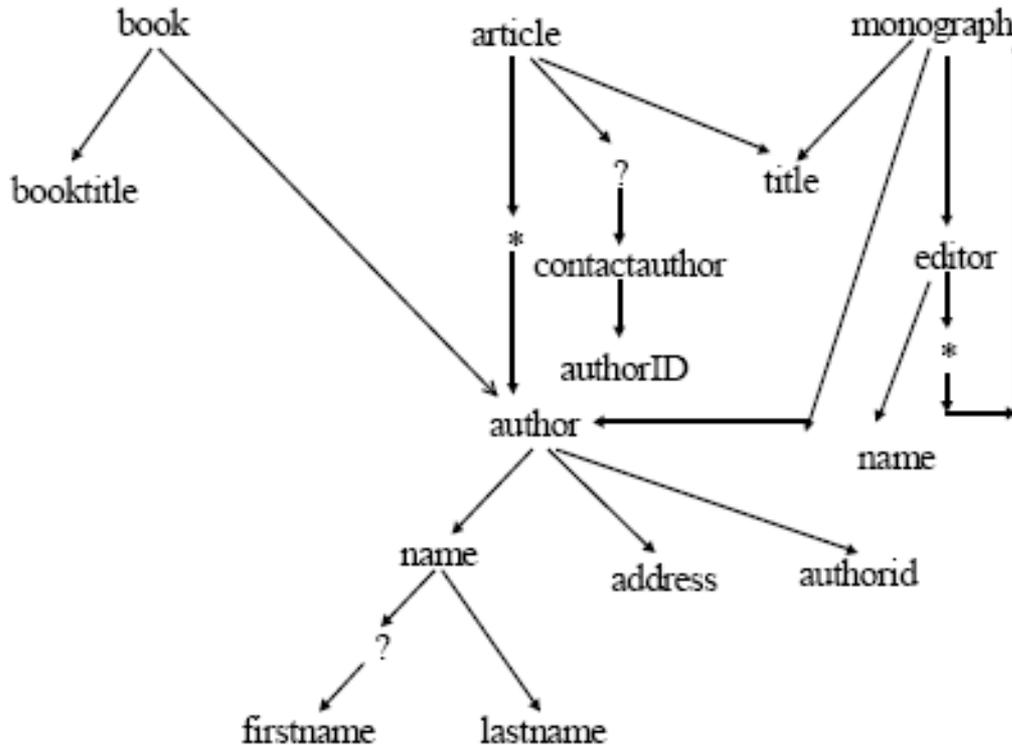
- ▶ O esquema para armazenar documentos que seguem uma DTD é a união dos conjuntos de relações criadas para cada elemento
- ▶ Para determinar o conjunto de relações necessário para armazenar um determinado elemento, construímos um grafo chamado “element graph”
 - ▶ Um elemento é escolhido para percorrer o grafo
 - ▶ Grafo vai sendo percorrido e cada elemento vai sendo marcado como visitado
 - ▶ Se o algoritmo tentar visitar um nodo já marcado, adicionar um “backpointer”
 - ▶ O resultado é uma árvore



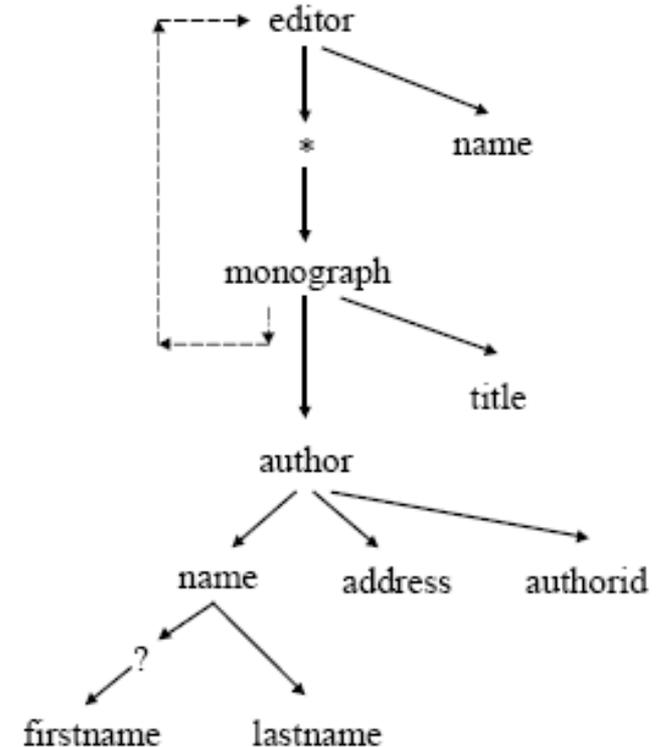
Basic Inlining

SHANMUGASUNDARAM et al., 1999

Grafo



Element Graph para elemento editor



Basic Inlining

SHANMUGASUNDARAM et al., 1999

- ▶ Dado um *element graph*, as relações são criadas como segue:
 - ▶ Uma relação é criada para o elemento raiz do *element graph*
 - ▶ Todos os descendentes são aninhados dentro desta relação, exceto:
 - ▶ Filhos de * são acomodados em relações separadas
 - ▶ Todo nodo que tem um backpointer é armazenado em uma relação separada (nova relação para lidar com recursão)
 - ▶ Ligações são feitas por chave estrangeira



Resultado

SHANMUGASUNDARAM et al., 1999

book (bookID: integer, book.booktitle : string, book.author.name.firstname: string, book.author.name.lastname: string, book.author.address: string, author.authorid: string)

booktitle (booktitleID: integer, booktitle: string)

article (articleID: integer, article.contactauthor.authorid: string, article.title: string)

article.author (article.authorID: integer, article.author.parentID: integer, article.author.name.firstname: string, article.author.name.lastname: string, article.author.address: string, article.author.authorid: string)

contactauthor (contactauthorID: integer, contactauthor.authorid: string)

title (titleID: integer, title: string)

monograph (monographID: integer, monograph.parentID: integer, monograph.title: string, monograph.editor.name: string, monograph.author.name.firstname: string, monograph.author.name.lastname: string, monograph.author.address: string, monograph.author.authorid: string)

editor (editorID: integer, editor.parentID: integer, editor.name: string)

editor.monograph (editor.monographID: integer, editor.monograph.parentID: integer, editor.monograph.title: string, editor.monograph.author.name.firstname: string, editor.monograph.author.name.lastname: string, editor.monograph.author.address: string, editor.monograph.author.authorid: string)

author (authorID: integer, author.name.firstname: string, author.name.lastname: string, author.address: string, author.authorid: string)

name (nameID: integer, name.firstname: string, name.lastname: string)

firstname (firstnamedID: integer, firstname: string)

lastname (lastnamedID: integer, lastname: string)

address (addressID: integer, address: string)

Resultado

SHANMUGASUNDARAM et al.,

book (bookID: integer, book.booktitle : string, book.author.name.firstname: string, book.author.address: string, author.authorid: string)

booktitle (booktitleID: integer, booktitle: string)

article (articleID: integer, article.contactauthor.authorid: string, article.title: string)

article.author (article.authorID: integer, article.author.parentID: integer, article.author.name.firstname: string, article.author.name.lastname: string, article.author.address: string, article.author.authorid: string)

contactauthor (contactauthorID: integer, contactauthor.authorid: string, contactauthor.name.firstname: string, contactauthor.name.lastname: string, contactauthor.address: string, contactauthor.authorid: string)

title (titleID: integer, title: string)

monograph (monographID: integer, monograph.parentID: integer, monograph.title: string, monograph.editor.name: string, monograph.author.name.firstname: string, monograph.author.name.lastname: string, monograph.author.address: string, monograph.author.authorid: string)

editor (editorID: integer, editor.parentID: integer, editor.name: string)

editor.monograph (editor.monographID: integer, editor.monograph.parentID: integer, editor.monograph.title: string, editor.monograph.author.name.firstname: string, editor.monograph.author.name.lastname: string, editor.monograph.author.address: string, editor.monograph.author.authorid: string)

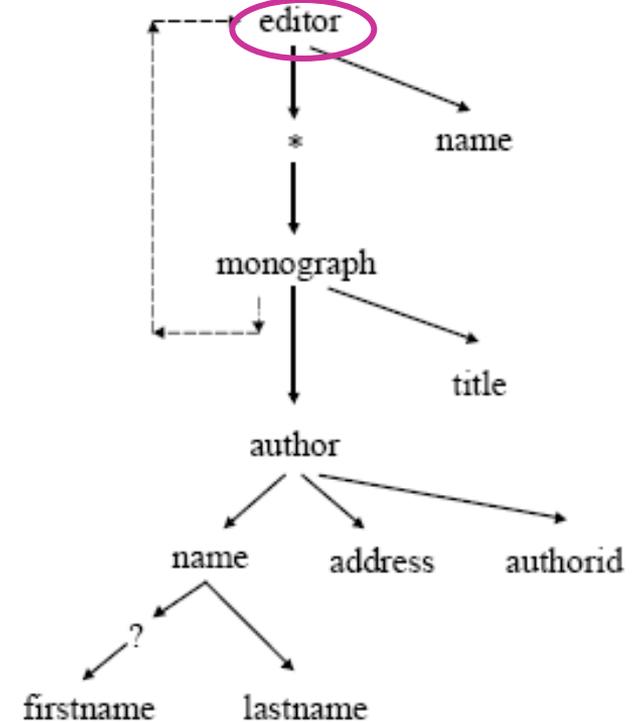
author (authorID: integer, author.name.firstname: string, author.name.lastname: string, author.address: string, author.authorid: string)

name (nameID: integer, name.firstname: string, name.lastname: string)

firstname (firstnameID: integer, firstname: string)

lastname (lastnameID: integer, lastname: string)

address (addressID: integer, address: string)



Resultado

SHANMUGASUNDARAM et al.,

book (bookID: integer, book.booktitle : string, book.author.name.firstname: string, book.author.address: string, author.authorid: string)

booktitle (booktitleID: integer, booktitle: string)

article (articleID: integer, article.contactauthor.authorid: string, article.title: string)

article.author (article.authorID: integer, article.author.parentID: integer, article.author.name.firstname: string, article.author.name.lastname: string, article.author.address: string, article.author.authorid: string)

contactauthor (contactauthorID: integer, contactauthor.authorid: string, contactauthor.address: string, contactauthor.authorid: string)

title (titleID: integer, title: string)

monograph (monographID: integer, monograph.parentID: integer, monograph.title: string, monograph.editor.name: string, monograph.author.name.firstname: string, monograph.author.name.lastname: string, monograph.author.address: string, monograph.author.authorid: string)

editor (editorID: integer, editor.parentID: integer, editor.name: string)

editor.monograph (editor.monographID: integer, editor.monograph.parentID: integer, editor.monograph.title: string, editor.monograph.author.name.firstname: string, editor.monograph.author.name.lastname: string, editor.monograph.author.address: string, editor.monograph.author.authorid: string)

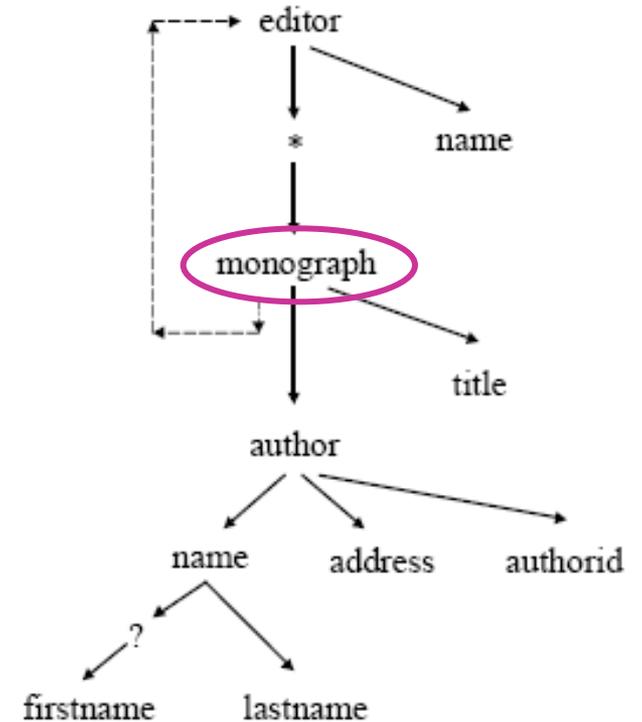
author (authorID: integer, author.name.firstname: string, author.name.lastname: string, author.address: string, author.authorid: string)

name (nameID: integer, name.firstname: string, name.lastname: string)

firstname (firstnameID: integer, firstname: string)

lastname (lastnameID: integer, lastname: string)

address (addressID: integer, address: string)



Resultado

SHANMUGASUNDARAM et al.,

book (bookID: integer, book.booktitle : string, book.author.name.firstname: string, book.author.address: string, author.authorid: string)

booktitle (booktitleID: integer, booktitle: string)

article (articleID: integer, article.contactauthor.authorid: string, article.title: string)

article.author (article.authorID: integer, article.author.parentID: integer, article.author.name.firstname: string, article.author.name.lastname: string, article.author.address: string, article.author.authorid: string)

contactauthor (contactauthorID: integer, contactauthor.authorid: string, contactauthor.address: string, contactauthor.authorid: string)

title (titleID: integer, title: string)

monograph (monographID: integer, monograph.parentID: integer, monograph.title: string, monograph.editor.name: string, monograph.author.name.firstname: string, monograph.author.name.lastname: string, monograph.author.address: string, monograph.author.authorid: string)

editor (editorID: integer, editor.parentID: integer, editor.name: string)

editor.monograph (editor.monographID: integer, editor.monograph.parentID: integer, editor.monograph.title: string, editor.monograph.author.name.firstname: string, editor.monograph.author.name.lastname: string, editor.monograph.author.address: string, editor.monograph.author.authorid: string)

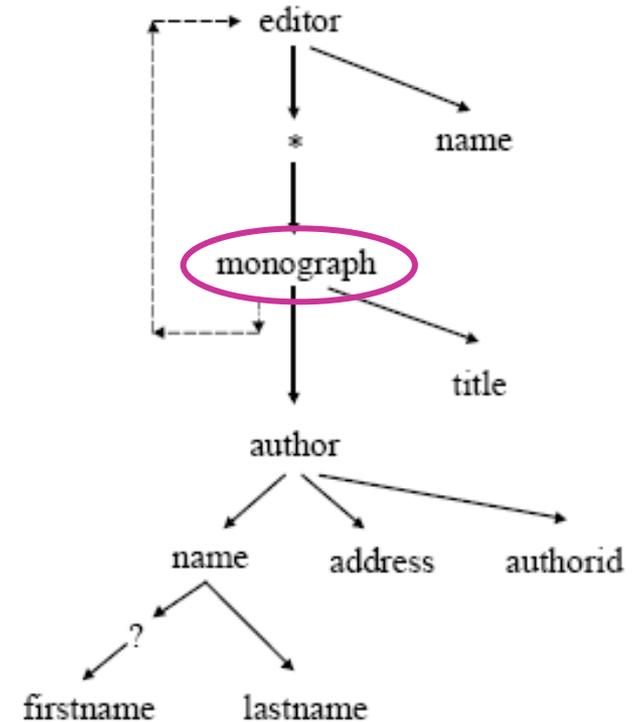
author (authorID: integer, author.name.firstname: string, author.name.lastname: string, author.address: string, author.authorid: string)

name (nameID: integer, name.firstname: string, name.lastname: string)

firstname (firstnamedID: integer, firstname: string)

lastname (lastnamedID: integer, lastname: string)

address (addressID: integer, address: string)



Basic Inlining

SHANMUGASUNDARAM et al., 1999

- ▶ Eficiente para consultas do tipo: me dê todos os autores dos livros
- ▶ Provavelmente muito ineficiente para outros tipos de consulta
 - ▶ Ex.: Liste todos os autores cujo primeiro nome é Jack
 - ▶ União de 5 consultas separadas
- ▶ Pensem nas atualizações! **Redundância** de informações! Péssima prática de modelagem!



Resultado

SHANMUGASUNDARAM et al., 1999

book (bookID: integer, book.booktitle : string, book.author.name.firstname: string, book.author.name.lastname: string, book.author.address: string, author.authorid: string)

booktitle (booktitleID: integer, booktitle: string)

article (articleID: integer, article.contactauthor.authorid: string, article.title: string)

article.author (article.authorID: integer, article.author.parentID: integer, article.author.name.firstname: string, article.author.name.lastname: string, article.author.address: string, article.author.authorid: string)

contactauthor (contactauthorID: integer, contactauthor.authorid: string)

title (titleID: integer, title: string)

monograph (monographID: integer, monograph.parentID: integer, monograph.title: string, monograph.editor.name: string, monograph.author.name.firstname: string, monograph.author.name.lastname: string, monograph.author.address: string, monograph.author.authorid: string)

editor (editorID: integer, editor.parentID: integer, editor.name: string)

editor.monograph (editor.monographID: integer, editor.monograph.parentID: integer, editor.monograph.title: string, editor.monograph.author.name.firstname: string, editor.monograph.author.name.lastname: string, editor.monograph.author.address: string, editor.monograph.author.authorid: string)

author (authorID: integer, author.name.firstname: string, author.name.lastname: string, author.address: string, author.authorid: string)

name (nameID: integer, name.firstname: string, name.lastname: string)

firstname (firstnamedID: integer, firstname: string)

lastname (lastnamedID: integer, lastname: string)

address (addressID: integer, address: string)

Shared Inlining

SHANMUGASUNDARAM et al., 1999

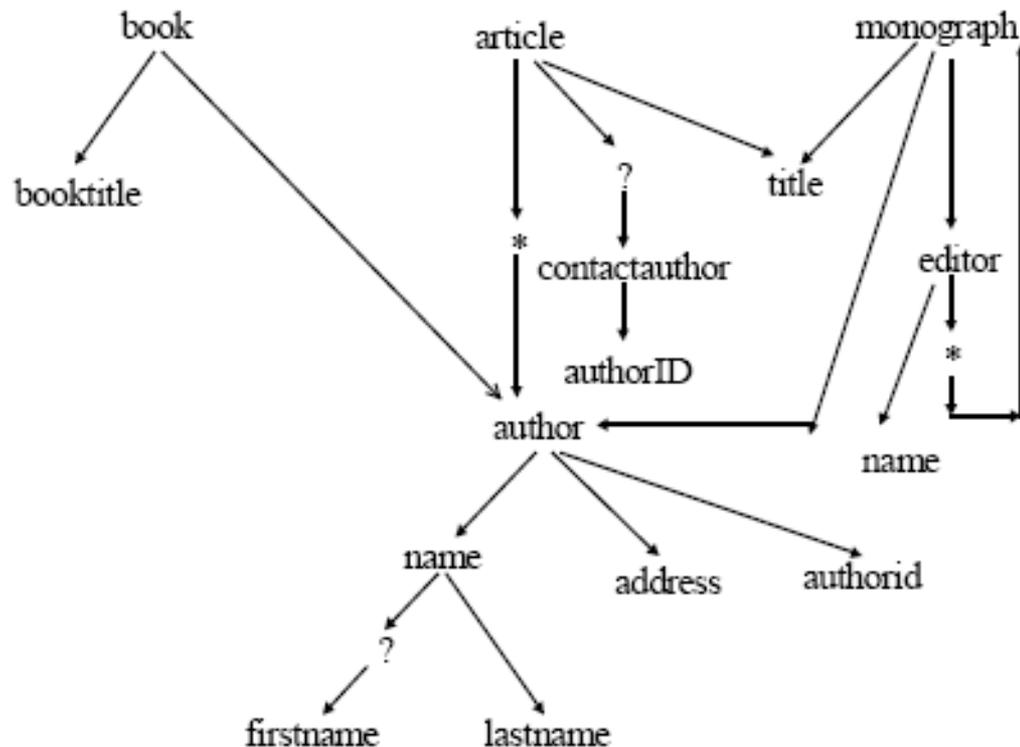
- ▶ Garante que cada elemento é representado em apenas uma tabela
- ▶ Para isso: Identificar quais nós são representados várias vezes



Shared Inlining

SHANMUGASUNDARAM et al., 1999

- ▶ Usar o grafo:
 - ▶ Nós que têm mais de uma aresta entrando são transformados em relações próprias
 - ▶ Nós restantes são colocados dentro de tabelas já existentes

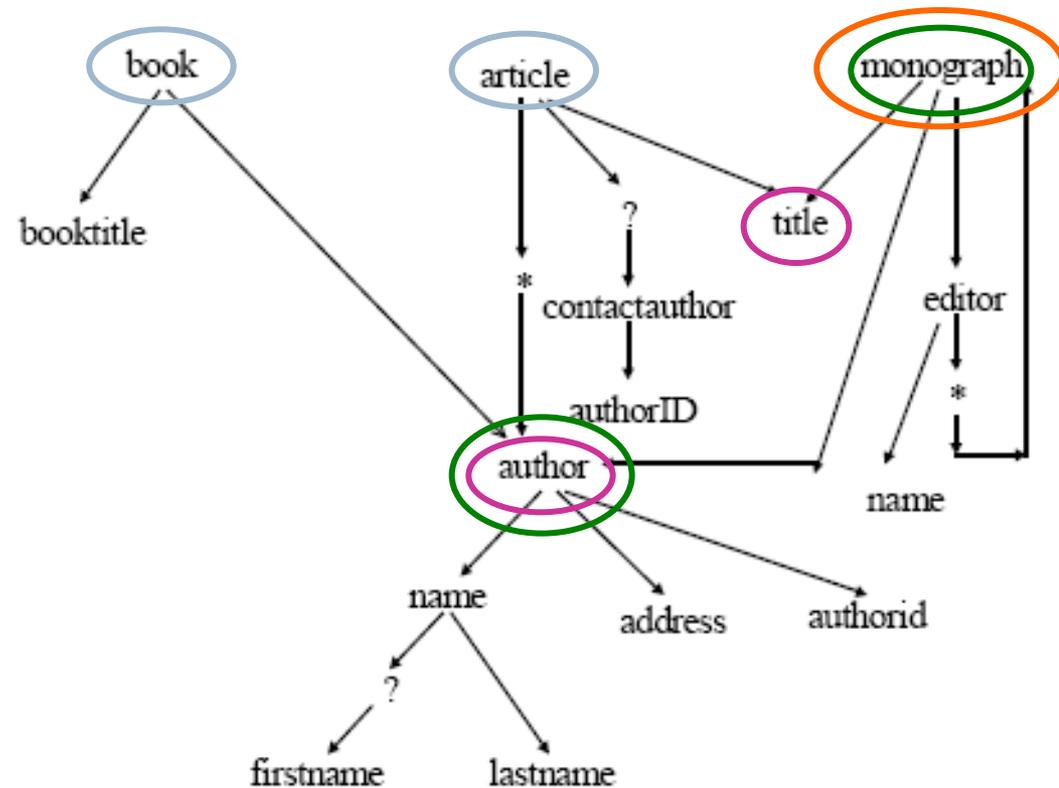


Shared Inlining

SHANMUGASUNDARAM et al., 1999

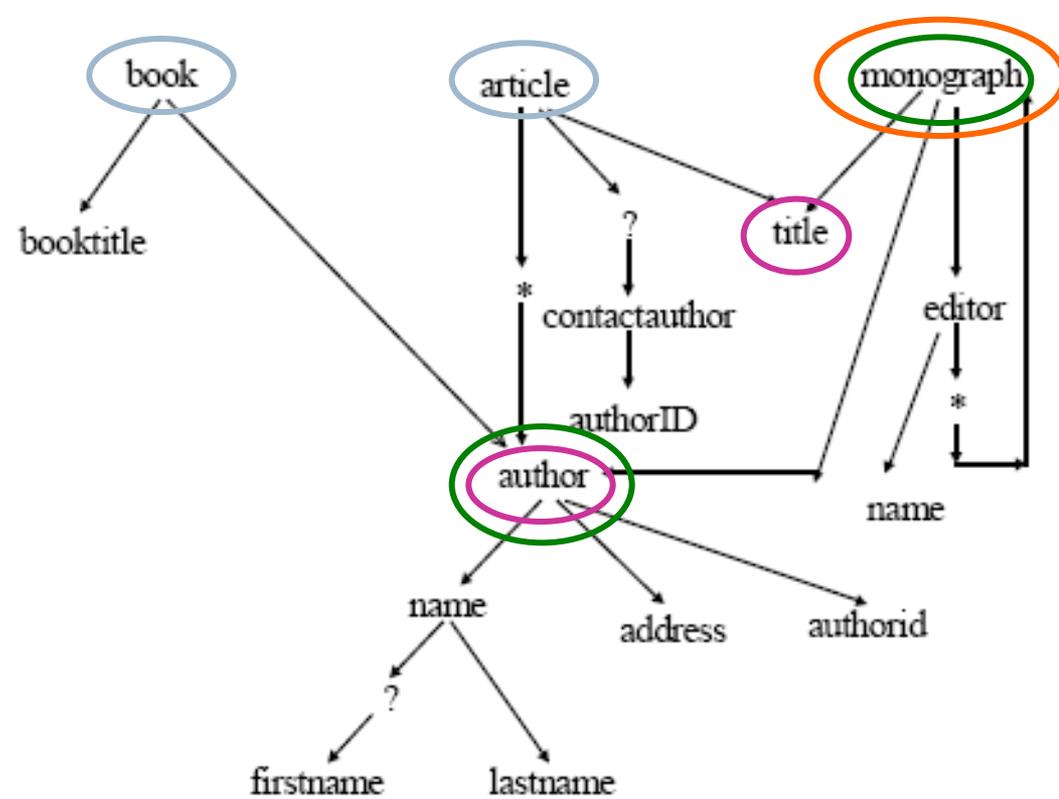
Usar o grafo:

- ▶ Relações próprias:
 - ▶ Nodos que têm **mais de uma aresta entrando**
 - ▶ Nodos com **zero arestas entrando** (pois eles não são acessíveis de nenhum outro nodo)
 - ▶ Elementos **depois de ***
 - ▶ Elementos **mutuamente recursivos** (elementos fortemente conectados – editor e monograph) – um deles é transformado em relação única
- ▶ Restantes são colocados dentro de tabelas já existentes



Shared Inlining

Resultado



book (bookID: integer, book.booktitle.isroot: boolean, book.booktitle : string)

article (articleID: integer, article.contactauthor.isroot: boolean, article.contactauthor.authorid: string)

monograph (monographID: integer, monograph.parentID: integer, monograph.parentCODE: integer, monograph.editor.isroot: boolean, monograph.editor.name: string)

title (titleID: integer, title.parentID: integer, title.parentCODE: integer, title: string)

author (authorID: integer, author.parentID: integer, author.parentCODE: integer, author.name.isroot: boolean, author.name.firstname.isroot: :boolean, author.name.firstname: string, author.name.lastname.isroot: boolean, author.name.lastname: string, author.address.isroot: boolean, author.address: string, author.authorid: string)

Shared Inlining

SHANMUGASUNDARAM et al., 1999

- ▶ **Considerações**

- ▶ Uniões não são mais necessárias
- ▶ No entanto, dependendo da consulta, junções são necessárias

- ▶ Abordagem que tenta balancear as vantagens das técnicas Basic e Shared: Hybrid

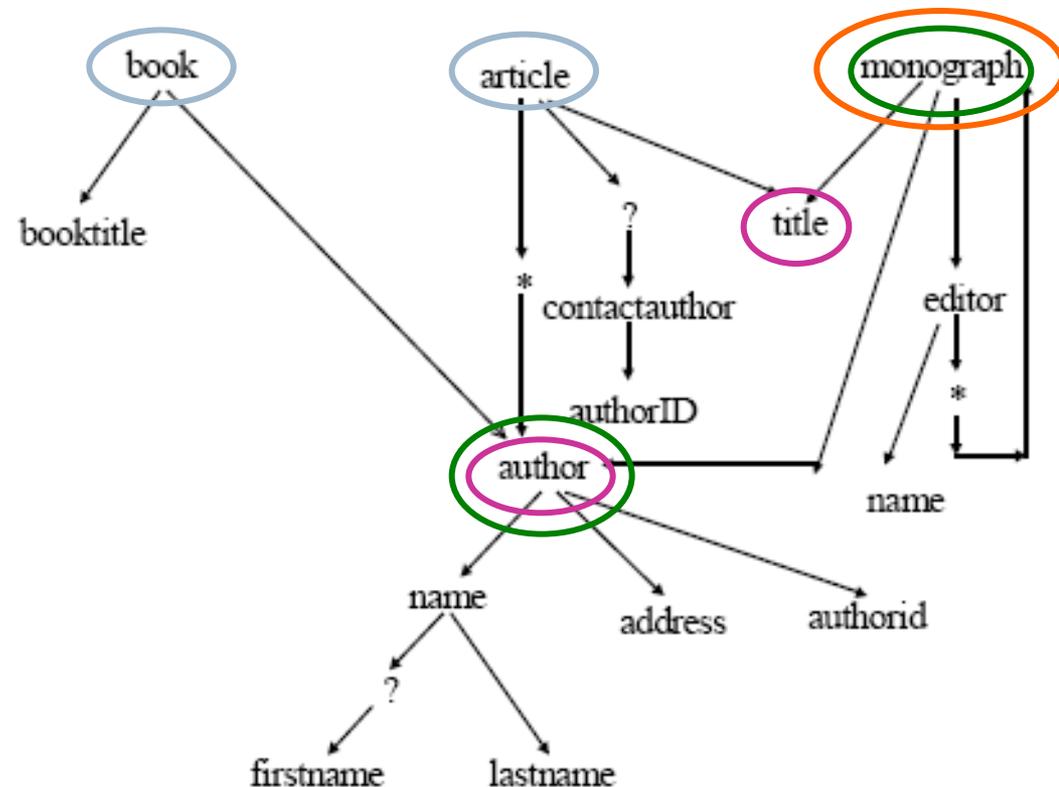


Hybrid Inlining

SHANMUGASUNDARAM et al., 1999

▶ Mesmo que Shared:

- ▶ Exceção: não cria relação separada para elementos que tem in-degree maior do que 1 e que não são recursivos e que não são filhos de *

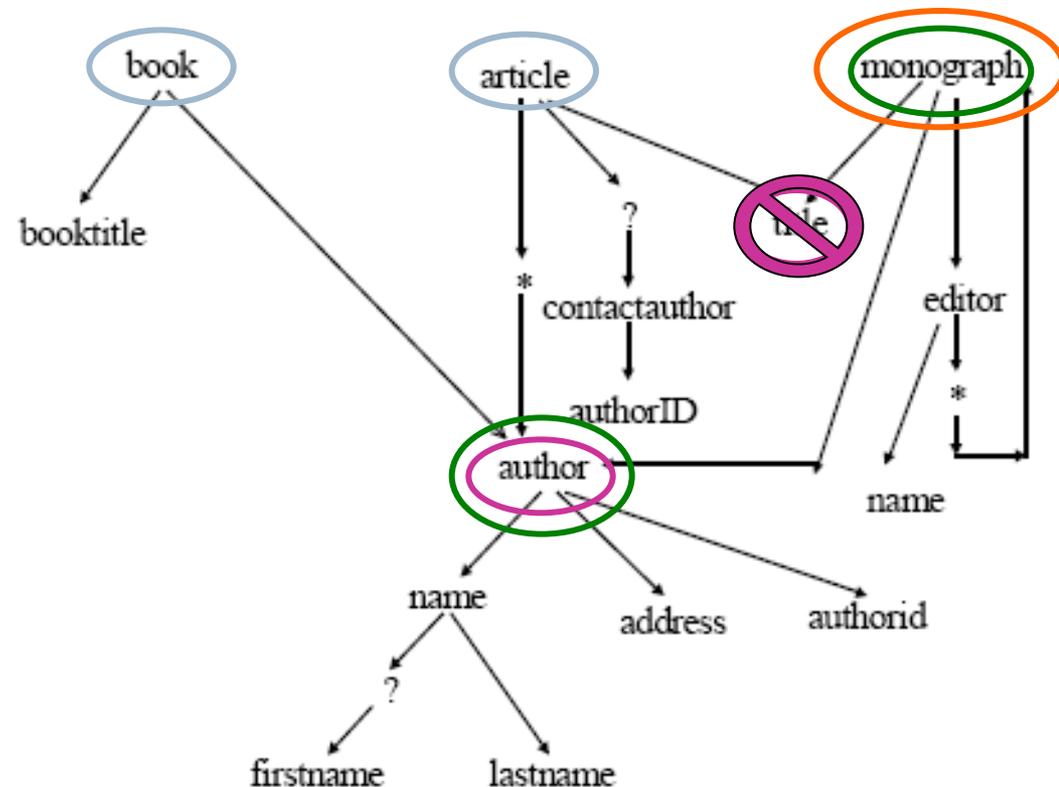


Hybrid Inlining

SHANMUGASUNDARAM et al., 1999

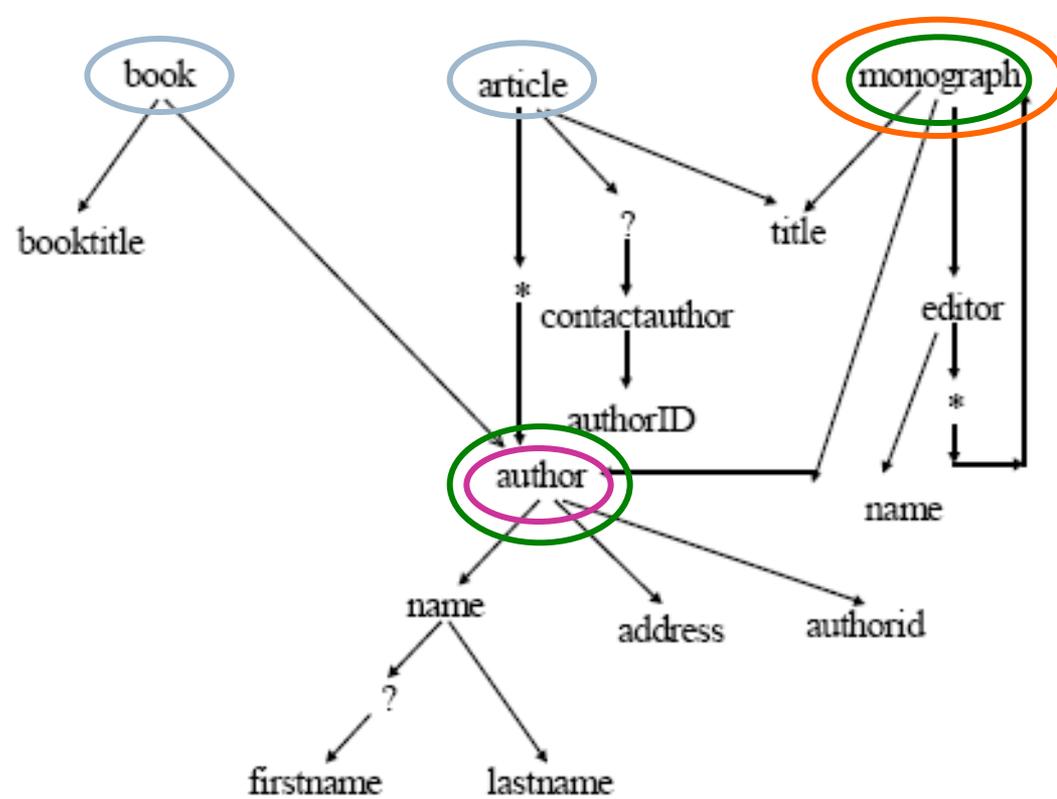
▶ Mesmo que Shared:

- ▶ Exceção: não cria relação separada para elementos que tem in-degree maior do que 1 e que não são recursivos e que não são filhos de *



Hybrid Inlining

Resultado



book (bookID: integer, book.booktitle.isroot: boolean, book.booktitle : string, author.name.firstname: string, author.name.lastname: string, author.address: string, author.authorid: string)

article (articleID: integer, article.contactauthor.isroot: boolean, article.contactauthor.authorid: string, article.title.isroot: boolean, article.title: string)

monograph (monographID: integer, monograph.parentID: integer, monograph.parentCODE: integer, monograph.title: string, monograph.editor.isroot: boolean, monograph.editor.name: string, author.name.firstname: string, author.name.lastname: string, author.address: string, author.authorid: string)

author (authorID: integer, author.parentID: integer, author.parentCODE: integer, author.name.isroot: boolean, author.name.firstname.isroot: boolean, author.name.firstname: string, author.name.lastname.isroot: boolean, author.name.lastname: string, author.address.isroot: boolean, author.address: string, author.authorid: string)

Referências

- ▶ CHEN, Y.; DAVIDSON, S. B.; ZHENG, Y. Constrain Preserving XML storage in Relations. In: INTERNATIONAL WORKSHOP ON THE WEB AND DATABASES, WEBDB, 2002, Madison, Wisconsin. Proceedings... [S.l.: s.n.], 2002. p.712.
 - ▶ CHEN, Y.; DAVIDSON, S. B.; ZHENG, Y. RRXS: redundancy reducing XML storage in relations. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 2003, Berlin, Germany. Proceedings... San Francisco:Morgan Kaufmann, 2003.
 - ▶ DEHAAN, D.; TOMAN, D.; CONSENS, M.; OZSU, M. T. A Comprehensive XQuery to SQL Translation using Dynamic Interval Encoding. In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, SIGMOD, 2003, San Diego, CA. Proceedings. . . [S.l.: s.n.], 2003.
 - ▶ DEUTSCH, A.; FERNANDEZ, M.; SUCIU, D. Storing semistructured data with STORED. In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, SIGMOD, 1999, Philadelphia, Pennsylvania. Proceedings... [S.l.: s.n.], 1999. p.431442.
-



Referências

- ▶ FLORESCU, D.; KOSSMANN, D. A performance evaluation of alternative mapping schemes for storing XML data in a relational database. France:INRIA, 1999. (Technical Report 3684).
 - ▶ LEE, D.; CHU, W. W. Constraints-Preserving Transformation from XML Document Type Denition to Relational Schema. In: INTERNATIONAL CONFERENCE ON ENTITY RELATIONSHIP, ER, 2000, Salt Lake City, Utah, USA. Proceedings... [S.l.: s.n.], 2000. p.323338.
 - ▶ MANOLESCU, I.; FLORESCU, D.; KOSSMANN, D. Pushing XML Queries inside Relational Databases. France: INRIA, 2001. (Technical Report 4112).
 - ▶ SHANMUGASUNDARAM, J.; TUFTE, K.; ZHANG, C.; HE, G.; DEWITT, D. J.; NAUGHTON, J. F. Relational Databases for Querying XML Documents: limitations and opportunities. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 1999, Edinburgh, Scotland, UK. Proceedings... San Francisco: Morgan Kaufmann, 1999. p.302314.
-



Referências

- ▶ SHANMUGASUNDARAM, J.; SHEKITA, E.; KIERNAN, J.; KRISHNAMURTHY, R.; VIGLAS, E.; NAUGHTON, J.; TATARINOV, I. A general technique for querying XML documents using a relational database system. Sigmod Record, [S.l.], v.30, n.3, p.2026, Sept. 2001.
- ▶ TATARINOV, I.; VIGLAS, E.; BEYER, K.; SHANMUGASUNDARAM, J.; SHEKITA, E. Storing and Querying Ordered XML Using a Relational Database System. In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, SIGMOD, 2002, Madison, Wisconsin. Proceedings... [S.l.: s.n.], 2002.



Banco de Datos Habilitado a XML/
Banco de Datos Híbrido

Motivação

- ▶ Empresas que já investiram enormes quantias em licenças de SGBDs relacionais/objeto relacionais
 - ▶ Não estão dispostas a adotar outros tipos de SGBDs para armazenar seus documentos XML
 - ▶ Necessidade de manter DBAs treinados nestes novos SGBDs implica em aumento de custos
- ▶ Empresas fabricantes dos principais SGBDs perceberam este filão de mercado e investiram para permitir armazenamento de docs. XML em seus SGBDs



Primeira geração: SGBDs Habilitados a XML

- ▶ Novo tipo de coluna (XML), que era capaz de armazenar um DOC XML; e/ou
- ▶ Uso de arquivos de mapeamento para “espalhar” o conteúdo dos docs XML em diversas tabelas do SGBD
- ▶ Suporte a consultas limitado



Nova geração: SGBDs híbridos

- ▶ Suporte a armazenamento de docs. XML em sua forma nativa, ao mesmo tempo em que mantém suporte a armazenamento de dados relacionais/objeto-relacionais

- ▶ Suporte a SQLX
 - ▶ Documentação do Oracle
http://download.oracle.com/docs/cd/B10500_01/appdev.920/a96616/arxml34.htm#1004572



Exemplo SQLX

SELECT

```
XMLElement("departments",  
  XMLElement("dept",  
    XMLElement("number", DEPTNO),  
    XMLElement("name", DNAME),  
    XMLElement("location", LOC)))  
FROM DEPT;
```



Exemplo SQLX

```
SELECT
  XMLElement("departments",
    XMLElement("dept",
      XMLElement("number", DEPTNO),
      XMLElement("name", DNAME),
      XMLElement("location", LOC)))
  FROM DEPT;
```

```
<departments>
  <dept>
    <number>10</number>
    <name>ACCOUNTING</name>
    <location>NEW YORK</location>
  </dept>
</departments>
<departments>
  <dept>
    <number>20</number>
    <name>RESEARCH</name>
    <location>DALLAS</location>
  </dept>
</departments >
```



Problema

- ▶ O resultado não é um XML bem formado
- ▶ Solução: usar a função **XMLAgg** para agregar tags iguais sobre o mesmo pai



Exemplo

▶ SELECT
XMLElement("departments", XMLAgg(
 XMLElement("dept",
 XMLElement("number", DEPTNO),
 XMLElement("name", DNAME),
 XMLElement("location", LOC))))
FROM DEPT;



Exemplo

```
<departments>
  <dept>
    <number>10</number>
    <name>ACCOUNTING</name>
    <location>NEW YORK</location>
  </dept>
  <dept>
    <number>20</number>
    <name>RESEARCH</name>
    <location>DALLAS</location>
  </dept>
</departments >
```

- ▶ SELECT
XMLElement("departments", XMLAgg(
 XMLElement("dept",
 XMLElement("number", DEPTNO),
 XMLElement("name", DNAME),
 XMLElement("location", LOC))))
FROM DEPT;
-



Outras funções

- ▶ **XMLElement()** Creates an XML Element.
- ▶ **XMLForest()** Creates an XML Fragment from passed-in components.
- ▶ **XMLColAttVal()** Creates an XML fragment and then expands the resulting XML so that each XML fragment has the name "column" with the attribute "name"
- ▶ **ExtractValue()** Takes as arguments an XMLType instance and an XPath expression and returns a scalar value of the resultant node.
- ▶ **XMLTransform()** Takes as arguments an XMLType instance and an XSL style sheet, which is itself a form of XMLType instance. It applies the style sheet to the instance and returns an XMLType.
- ▶ **XMLSequence()** Takes input and returns either a varray of the top-level nodes in the XMLType, or an XMLSequence type an XML document for each row of the cursor.
- ▶ **XMLConcat()** Takes as input a series of XMLType instances, concatenates the series of elements for each row, and returns the concatenated series.
- ▶ **UpdateXML()** Takes as arguments an XMLType instance and an XPath-value pair, and returns an XMLType instance with the updated value.

Tarefa

- ▶ Cada aluno pesquisa sobre um dos SGBDs habilitados a XML/híbridos (Oracle, DB2, SQL Server – ou outro que vcs encontrarem)
- ▶ Ver o suporte que o banco escolhido dá para XML
 - ▶ Como armazenar um DOC XML no banco?
 - ▶ Como recuperar o documento armazenado?
 - ▶ É possível gerar um doc. XML a partir de dados relacionais pré-existentes? Como?
- ▶ Cada aluno irá apresentar o que encontrou no final da aula

