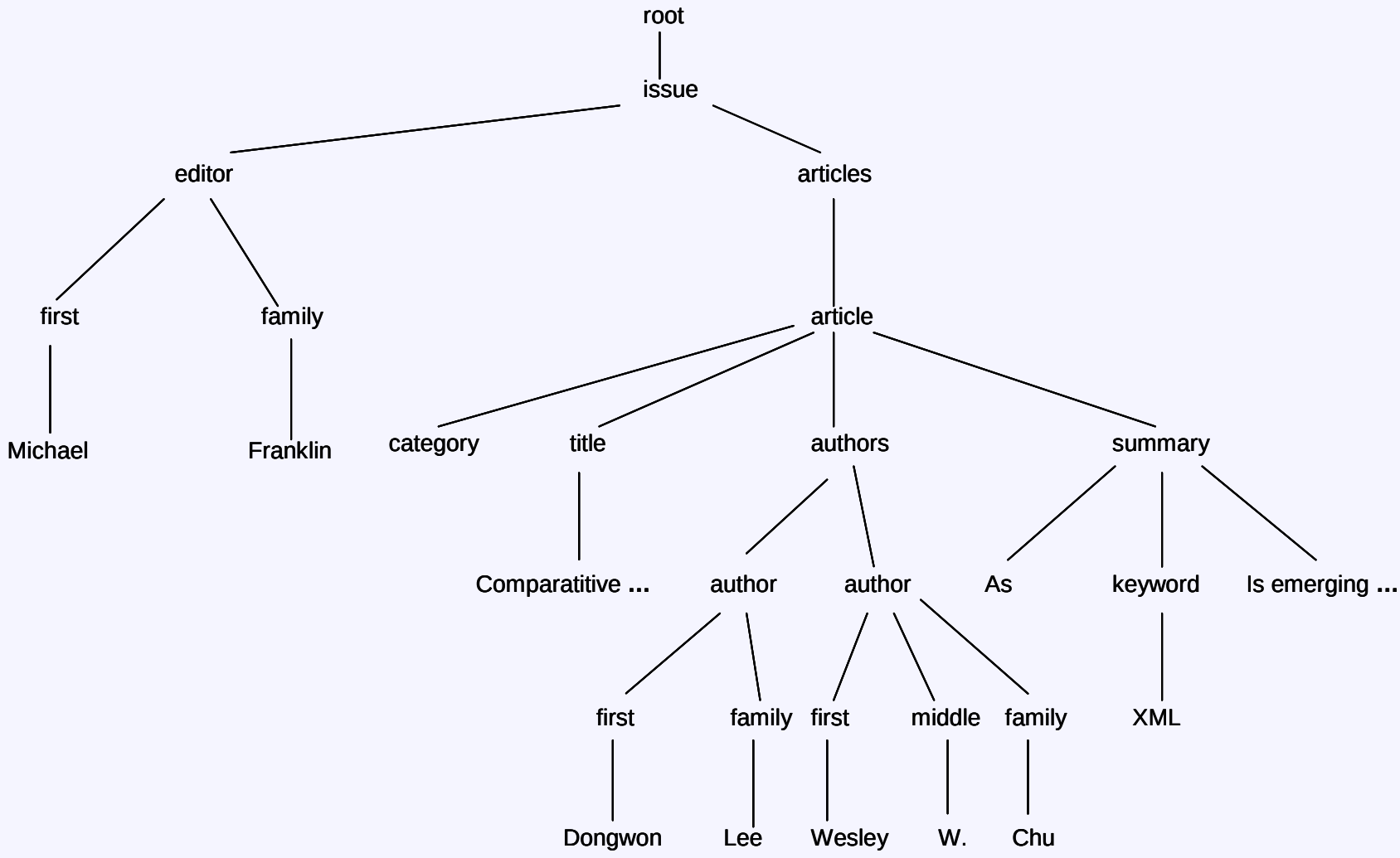
Speichern von XML

- generische Speicherung: finde eine Abbildung, die für beliebige XML-Dokumente anwendbar ist.
- schema-basierte Speicherung: gegeben ein XML-Schema (DTD), finde ein möglichst geeignetes relationales Schema.

Generische Speicherung:

Beispiel

```
<issue>
 <editor>
  <first>Michael</first>
  <family>Franklin</family>
 </editor>
 <articles>
  <article category="Research surveys">
   <title>Comparative Analysis of Six XML Schema Languages</title>
   <authors>
    <author>
     <first>Dongwon</first>
     <family>Lee</family>
    </author>
    <author>
     <first>Wesley</first>
     <middle>W.</middle>
     <family>Chu</family>
     </author>
   </authors>
   <summary>As <keyword>XML</keyword> is emerging ... </summary>
  </article>
 </articles>>
</issue>
```


root
issue
editor
articles
first
family
article
Michael
Franklin
category
title
authors
summary
Comparatitive ...
author
author
As
keyword
Is emerging ...
first
family
first
middle
family
XML
Dongwon
Lee
Wesley
W.
Chu

direkte Baum-Darstellung

Es werden die Kanten des XML-Baumes repräsentiert.

$$Edge(node, predecessor, ordinal, name, value)$$

- *node, predecessor*: Knoten-IDs,
- *ordinal*: Nummerierung bzgl. direktem Vorgängerknoten,
- *name*: Attribut/Element-Name des Knotens,
- *value*.

Beantwortung von Anfragen, z.B in XPath, verlangt Rekursion!

node	predecessor	ordinal	name	value
1		1	root	
2	1	1	issue	
3	2	1	editor	
4	3	1	first	
5	4	1		Michael
6	3	2	family	
7	6	1		Franklin
8	2	2	articles	
9	8	1	article	
10	9	1	category	research surveys
11	9	2	title	
12	11	1		Comparative Analysis ...
...				

intervallbasierte Baum-Darstellung

M. Yoshikawa et al. *XRel: A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases*. ACM ToIT, Vol 1, No 1, 2001.

In jeweils einer eigenen Relation werden Element-, Attribut- und Textknoten repräsentiert, wobei im Wesentlichen zu jedem Knoten der Pfad zu ihm identifiziert wird und zusätzlich die *Region* des Dokumentes angegeben wird, die er belegt. Zusätzlich werden alle *einfachen* XPath-Ausdrücke in einer Relation gespeichert.

Region eines Knotens:

- Die Region eines Text- und Elementknotens ist ein Paar von Zahlen, die der Start- und Endposition des Knotens im Dokument entspricht.
- Die Region eines Attributknotens ist gegeben durch zwei identische Zahlen, die gleich der Startposition des Vorgängerelementes + 1 sind. Hat ein Element mehrere Attribute, so ist keine Ordnung auf diesen Attributen impliziert.

Syntax einfacher XPath-Ausdruck:

```
SimplePathExpr ::= '#/' Step | SimplePathExpr '#/' Step
Step           ::= NameTest | '@' NameTest
NameTest       ::= QName
```

Element

pathID	start	end	ordinal
1	0	729	1
2	7	70	1
3	15	36	1
4	37	61	1
5	71	721	1
6	81	710	1
8	118	180	1
9	181	335	1
10	190	248	1
11	198	219	1
12	220	239	1
10	249	325	2
11	257	277	1
13	278	296	1
12	297	316	1
14	336	700	1
15	348	369	1

Attribute

pathID	start	end	value
7	82	82	research surveys

Text

pathID	start	end	value
3	22	28	Michael
4	45	52	Franklin
8	125	172	Comparative Analysis ...
11	205	211	Dongwon
. . .			

Path

pathID	pathExp
1	#/issue
2	#/issue#/editor
3	#/issue#/editor#/first
4	#/issue#/editor#/family
5	#/issue#/articles
6	#/issue#/articles#/article
7	#/issue#/articles#/article#/@category
8	#/issue#/articles#/article#/title
9	#/issue#/articles#/article#/authors
10	#/issue#/articles#/article#/authors#/author
11	#/issue#/articles#/article#/authors#/author#/first
12	#/issue#/articles#/article#/authors#/author#/family
13	#/issue#/articles#/article#/authors#/author#/middle
14	#/issue#/articles#/article#/summary
15	#/issue#/articles#/article#/summary#/keyword

XPath:

```
/issue//family
```

SQL:

```
SELECT e1.start, e1.end
FROM Element e1, Path p1
WHERE p1.pathExp
 LIKE '#/issue#%/family'
AND e1.pathID = p1.pathID
ORDER BY e1.start, e1.end
```

XPath:

```
//article[summary/keyword='XML']//author/family
```

SQL:

```
SELECT e5.start, e5.end
FROM Path p1, Path p2, Path p5,
     Element e1, Element e5, Text t3
WHERE p1.pathExp LIKE '#%/article'
AND p3.pathExp LIKE '#%/article#/summary#/keyword'
AND p5.pathExp LIKE '#%/article#%/author#/family'
AND e1.pathID = p1.pathID
AND e5.pathID = p5.pathID
AND t3.pathID = p3.pathID
AND e1.start < t3.start
AND e1.end > t3.end
AND e1.start < e5.start
AND e1.end > e5.end
AND t3.value = 'XML'
ORDER BY e5.start, e5.end
```

DTD-basierter Ansatz: Inlining

J. Shanmugasundaram, et al. *Relational Databases for Querying XML Documents: Limitations and Opportunities*, 25th VLDB Conference, 1999.

Gegeben eine DTD D.

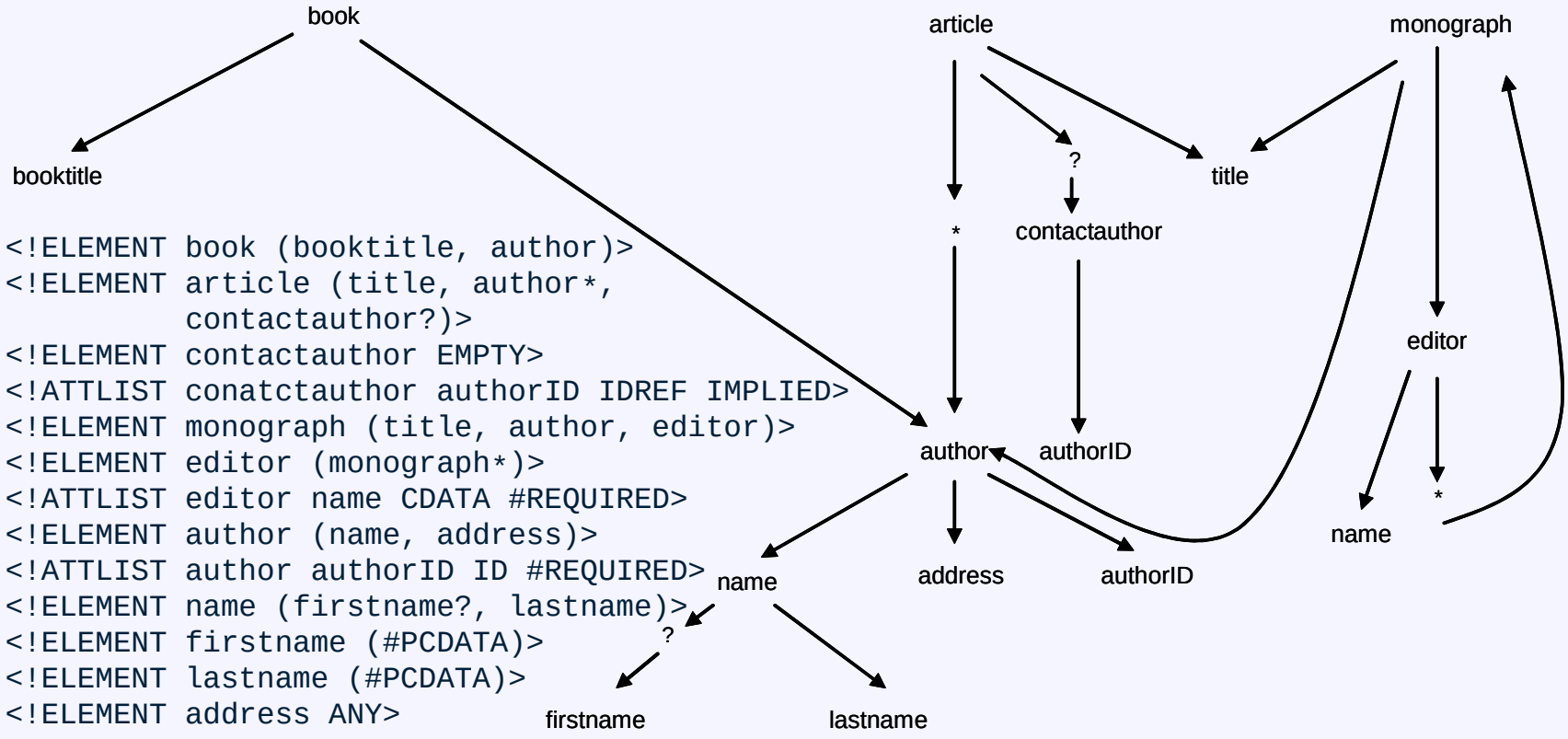
- Vereinfache D zu D'.
- Konstruiere zu D' einen DTD-Graphen G.
- Für jedes Element E in G bilde den Element-Graphen $G(E)$.
- Basierend auf $G(E)$ konstruiere zu E ein Relationsschema $R(E)$.

Vereinfachen einer DTD:

- $(E_1, E_2)* \rightarrow E_1*, E_2*,$
- $(E_1, E_2)? \rightarrow E_1?, E_2?,$
- $(E_1|E_2)* \rightarrow E_1?, E_2?,$
- $E_1 * * \rightarrow E_1*,$
- $E_1*? \rightarrow E_1*,$
- $E_1?* \rightarrow E_1*,$
- $E_1?? \rightarrow E_1?,$
- $\ldots, aE*, \ldots, aE*, \ldots \rightarrow aE*, \ldots,$
- $\ldots, aE*, \ldots, aE?, \ldots \rightarrow aE*, \ldots,$
- $\ldots, aE?, \ldots, aE*, \ldots \rightarrow aE*, \ldots,$
- $\ldots, aE?, \ldots, aE?, \ldots \rightarrow aE*, \ldots,$
- $\ldots, aE, \ldots, aE, \ldots \rightarrow aE*, \ldots,$

Die Transformation zerstört Angaben in der DTD über die Anordnung von Elementen – jedoch betrachtet werden ja nur solche Dokumente, die die DTD erfüllen D und deren Ordnung man somit bei der Abspeicherung festhalten kann.

Beispiel:




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

Idee der Vorgehensweise:

- einem Element wird ein Relationsschema (mit Schlüssel) zugeordnet, wenn
  - es Wurzel im DTD-Graphen ist,
  - es direkt unter einem * liegt,
  - es Senke von mehr als einer Kante ist,
  - es auf einem Zyklus liegt.
- führe Elemente mit Schema ein *inlining* durch, d.h.
  - übernehme alle erreichbaren Elemente als Attribute in das relationale Schema, die selbst kein eigenes Schema zugeordnet bekommen haben,
  - hat ein Vorgänger-Element eine eigene Relation, so übernehme seinen Schlüssel als Fremdschlüssel.
- fasse, wenn möglich, Relationsschemata zusammen.