Non-Standard-Datenbanken

Semistrukturierte Datenbanken

Prof. Dr. Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

Integration verschiedener Datenquellen

Semistrukturierte Datenbanken: Überblick

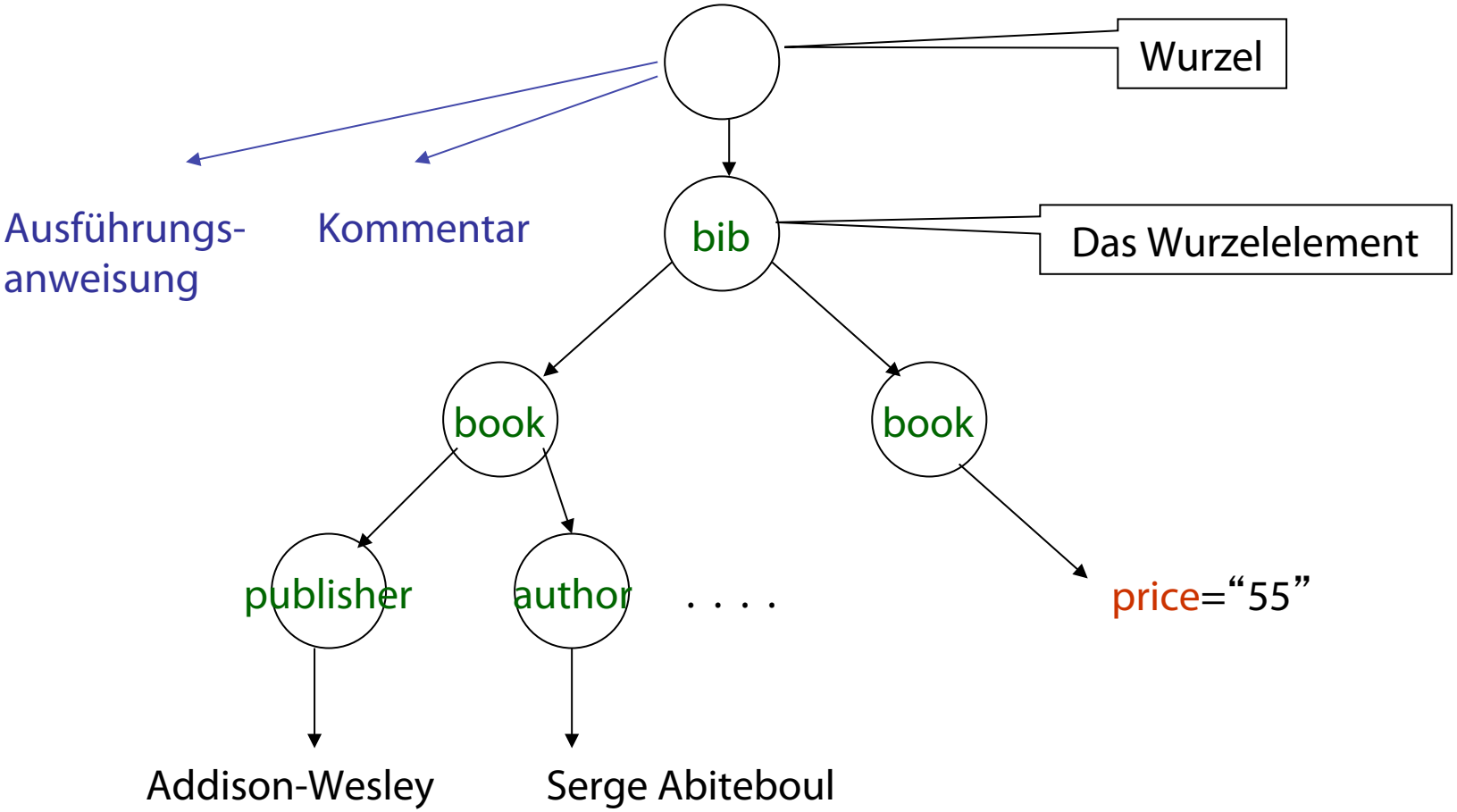
- **Datenspezifikationssprache XML**
  - Gedacht zur Datenkommunikation und –integration
  - Begriff des XML-“Dokuments“
  - Strukturprüfung möglich (DTD, XML-Schema)
  - Repräsentation von XML-Daten in SQL möglich?
- **Anfragesprache XPath**
  - Beispiele
  - Übersetzung nach SQL möglich?
- **Anfragesprache XQuery**
  - Beispiele
  - Einige Aspekte der optimierten Anfragebeantwortung

Acknowledgment: XPath-Präsentationen basieren auf Darstellungen von Dan Suciu, Univ. Washington
http://www.cs.washington.edu/homes/suciu/COURSES/590DS
Anpassungen für XQuery 3.0 sind erfolgt

Beispiel für ein Dokument

```
<bib>
    <book>  <publisher> Addison-Wesley </publisher>
            <author> Serge Abiteboul </author>
            <author> <first-name> Rick </first-name>
                    <last-name> Hull </last-name>
            </author>
            <author> Victor Vianu </author>
            <title> Foundations of Databases </title>
            <year> 1995 </year>
    </book>
    <book price="55">
            <publisher> Freeman </publisher>
            <author> Jeffrey D. Ullman </author>
            <title> Principles of Database and Knowledge Base Systems </title>
            <year> 1998 </year>
    </book>
</bib>
```

Datenmodell für XPath



XPath

- W3C-Standard: http://www.w3.org/TR/xpath (11/99)
- Wird in anderen W3C-Standards verwendet
  - XSL Transformations (XSLT, XML→HTML)
  - XML Link (XLink, Verweise in XML Dokumenten)
  - XML Pointer (XPointer, XPath in URIs)
  - XML Query (XQuery)
- Ursprünglich Teil von XSLT

XPath: Einfache Ausdrücke

/bib/book/year

Ergebnis: <year> 1995 </year>
<year> 1998 </year>

/bib/paper/year

Ergebnis: empty (keine Papiere in diesem Jahr)

XPath: Hülloperator //

//author

Ergebnis:<author> Serge Abiteboul </author>
<author> <first-name> Rick </first-name>
<last-name> Hull </last-name>
</author>
<author> Victor Vianu </author>
<author> Jeffrey D. Ullman </author>

/bib//first-name
Ergebnis: <first-name> Rick </first-name>

XPath: Platzhalter

//author/*

Ergebnis: <first-name> Rick </first-name>
<last-name> Hull </last-name>

* passt auf jedes Element

XPath: Attributknoten

/bib/book/@price

Ergebnis: “55”

@price steht für eine Referenz auf den Attributwert

XPath: Qualifizierung

/bib/book/author[first-name]

Ergebnis: <author> <first-name> Rick </first-name>
<last-name> Hull </last-name>
</author>

XPath: Funktionen

/bib/book/author/text()

Ergebnis: Serge Abiteboul
Victor Vianu
Jeffrey D. Ullman

Rick Hull tritt nicht auf, da firstname, lastname vorhanden

/bib/book/*[name() != author]

Zugriff auf Namen eines Knotens mittels Funktion name()

XPath: Weitere Qualifizierungen

/bib/book/author[firstname][address[//zip][city]]/lastname

Ergebnis: <lastname> … </lastname>
<lastname> … </lastname>

XPath: Weitere Qualifizierungen

/bib/book[@price < “60”]

/bib/book[author/@age < “25”]

/bib/book[author/text()]

XPath: Zusammenfassung

bib	passt auf ein bib Element
*	passt auf beliebiges Element
/	passt auf das root Element
/bib	passt auf bib Element unter root
bib/paper	passt auf paper in bib
bib//paper	passt auf paper in bib, in jeder Tiefe
//paper	passt auf paper in jeder Tiefe
paper|book	passt auf paper oder book
@price	passt auf price Attribut
bib/book/@price	passt auf price Attribute in book, in bib
bib/book/[@price<“55”]/author/lastname	passt auf …

XPath: Weitere Details

- Ein Xpath-Ausdruck, p, beschreibt eine Beziehung zwischen:
  - Einem *Kontextknoten* (Ausgangspunkt), und
  - Eine Knoten in der Antwortmenge (bzw. Antwortsequenz)
- Anders ausgedrückt: p denotiert eine Funktion:
  - S[p] : Knoten -> 2^{Knoten}
- Beispiele:
  - author/firstname
  - . = self
  - .. = parent
  - part/*/*/subpart/../name = part/*/*[subpart]/name

Das Wurzeldokument und die Wurzel

- <bib> <paper> 1 </paper> <paper> 2 </paper> </bib>
- bib ist das "***Wurzeldokument***"
- Die Wurzel ist über bib

- /bib = ergibt das Wurzeldokument
- / = ergibt die Wurzel

- Warum? Es gibt Elemente neben <bib> (Kommentar usw.)

XPath: Weitere Details

- Navigation entlang 13 Achsen möglich:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - parent ..
  - self .
  - preceding
  - preceding-sibling
  - namespace

Wir haben bislang diesen Teil betrachtet.

XPath: Weitere Details

- Beispiele:
  - child::author/child:lastname = author/lastname
  - child::author/descendant::zip = author//zip
  - child::author/parent::* = author/..
  - child::author/attribute::age = author/@age
- Was bedeuten folgende Ausdrücke?
  - paper/publisher/parent::*/author
  - /bib//address[ancestor::book]
  - /bib//author/ancestor::*//zip

ancestor:: denotiert nächsten Vorgänger

XPath: Und noch weitere Details

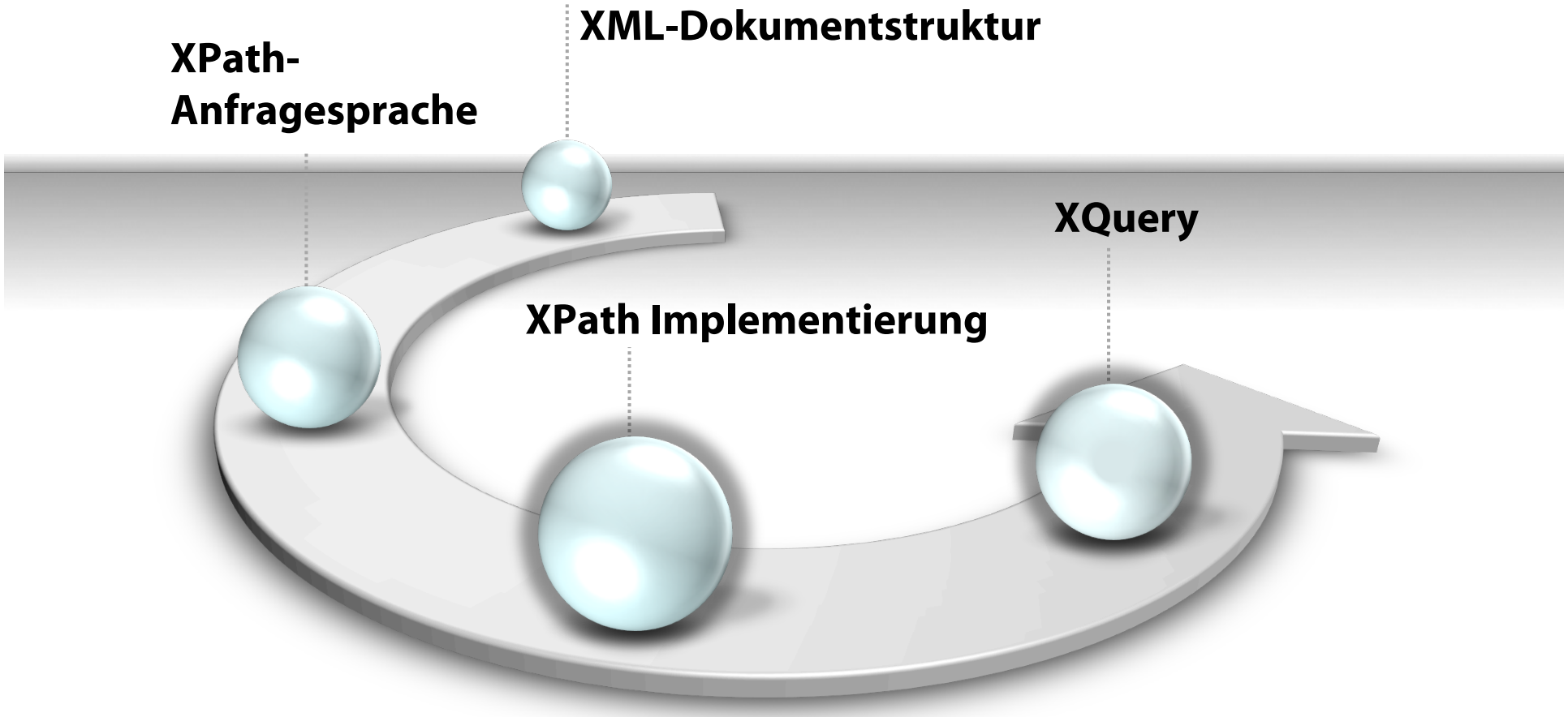
- name() = Name des aktuellen Knotens
  - /bib//*[name()=book] identisch zu /bib//book

- Was bedeutet dieser Ausdruck?
  /bib//*[ancestor::*[name()!=book]]

- Navigationsachsen erhöhen die Ausdruckskraft!

Non-Standard-Datenbanken

Semistrukturierte Datenbanken



Im nächsten Teil werden wir uns die letzten 2 Themen erarbeiten

Unser Beispieldokument noch einmal

```
<bib>
    <book>  <publisher> Addison-Wesley </publisher>
            <author> Serge Abiteboul </author>
            <author> <first-name> Rick </first-name>
                    <last-name> Hull </last-name>
            </author>
            <author> Victor Vianu </author>
            <title> Foundations of Databases </title>
            <year> 1995 </year>
    </book>
    <book price="55">
            <publisher> Freeman </publisher>
            <author> Jeffrey D. Ullman </author>
            <title> Principles of Database and Knowledge Base Systems </title>
            <year> 1998 </year>
    </book>
</bib>
```

Speicher- und Zugriffstechniken für XML

1. Verwendung **bestehender Techniken**
   - Abbildung auf relationale Datenbanken
     - Verwendung des physischen Datenmodells
     - Verwendung der Zugriffsoperatoren und deren Optimierungstechniken
   - Abbildung des XML-Datenmodells auf relationale Schemata notwendig
2. Entwicklung **neuer Techniken**
   - Neues physisches Datenmodell
     - Ausnutzung neuer Hardware-Strukturen

XML-Daten in Relationalen Datenbanken

- Verwendung eines generischen Schemas

  D. Florescu, D. Kossmann, Storing and Querying XML Data using an RDBMS,
  Bulletin of the Technical Committee on Data Engineering, 22(3):27-34, September **1999**.

- Verwendung von DTDs zur Herleitung eines Schemas

  J. Shanmugasundaram, K. Tufte, C. Zhang, H. Gang, DJ. DeWitt, and JF. Naughton
  Relational databases for querying xml documents: Limitations and opportunities.
  Proceedings of the 25th International Conference on Very Large Data Bases, **1999**.

- Herleitung eines Schemas aus gegebenen Daten

  A. Deutsch, M. Fernandez, D. Suciu, Storing semistructured data with STORED,
  ACM SIGMOD Record 28 (2), 431-442, **1999**.

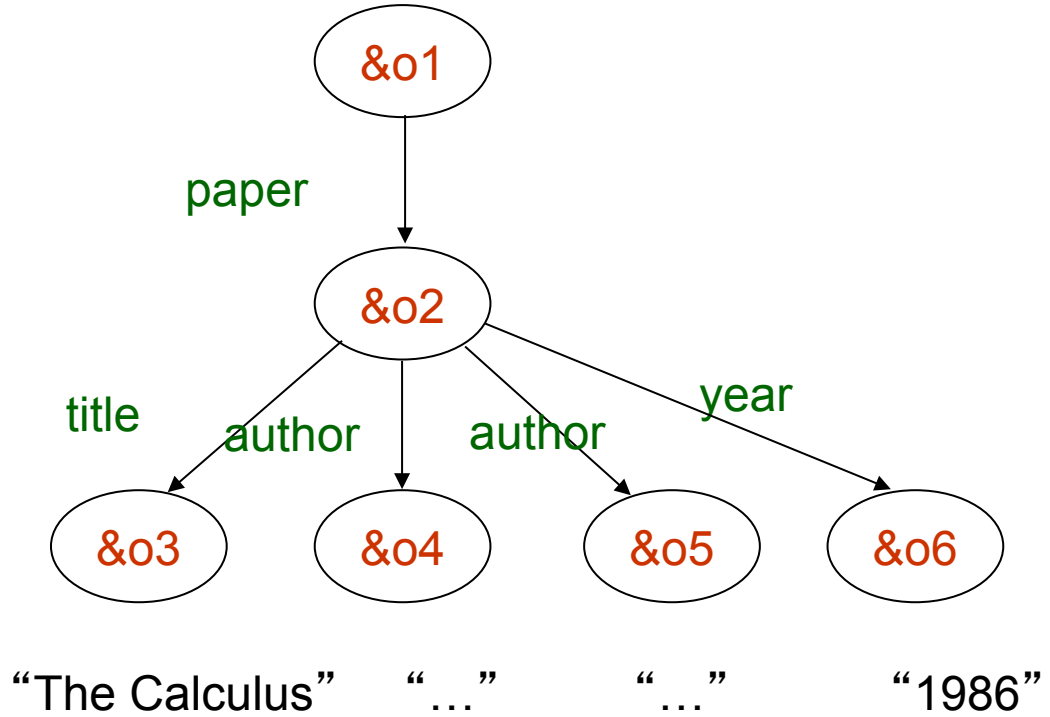
- Verwendung einer sog. Pfad-Relation

  M. Yoshikawa, T. Amagasa, S. Takeyuki, S. Uemura,
  XRel: A Path-Based Approach to Storage and Retrieval of XML Documents
  using Relational Databases, ACM TOIT Volume 1 (1), 110-141, **2001**.

Generisches Schema: Ternäre Relation

Ref(Source, Label, Dest)

Val(Node, Value)



Ref

Source	Label	Dest
&o1	paper	&o2
&o2	title	&o3
&o2	author	&o4
&o2	author	&o5
&o2	year	&o6

Val

Node	Value
&o3	The Calculus
&o4	…
&o5	…
&o6	1986

[Florescu, Kossman, 1999]

XML in ternären Relationen: Aufgabe

- Schema für SQL:

Ref(Source, Label, Dest)
Val(Node, Value)

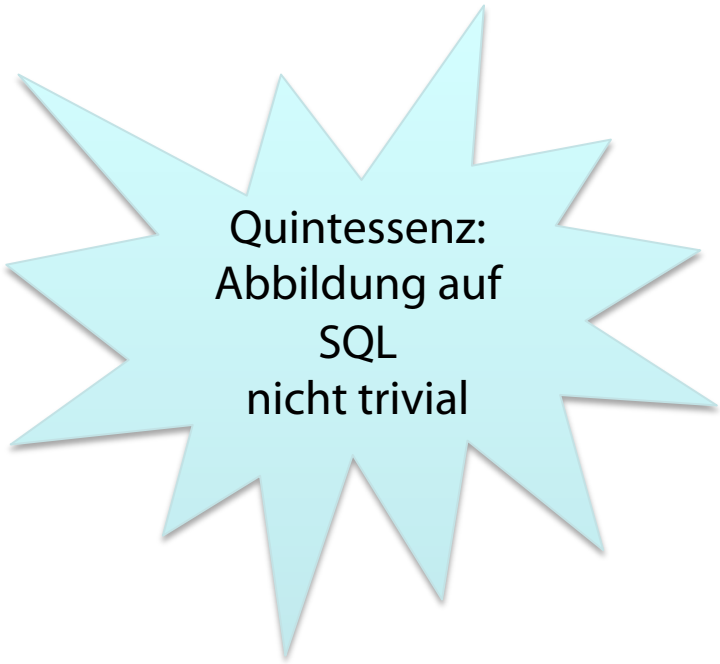
- XPath: /paper[year="1986"]/author

- SQL:

Generisches Schema: Ternäre Relation?

- In der Praxis werden mehrere Tabellen benötigt:
  - Sonst Tabellen zu groß
  - Sonst Datentypen nicht unterstützt

RefTag1(Source, Dest)
RefTag2(Source, Dest)
…
IntVal(Node, IntVal)
RealVal(Node, RealVal)
…



DTDs zur Herleitung eines Schemas

- DTD (Kontextfreie Grammatik)

```
<!ELEMENT paper (title, author*, year?)>
<!ELEMENT author (firstName, lastName)>
```

- Relationales Schema:

Paper(PID, Title, Year)
PaperAuthor(PID, AID)
Author(AID, FirstName, LastName)

[Shanmugasundaram et al. 1999 siehe oben]

Siehe auch: V. Christophides, S. Abiteboul, S. Cluet, M. Scholl,
From structured documents to novel query facilities,
ACM SIGMOD Record 23 (2), 313-324, **1994**

Aus DTD hergeleitetes Schema: Aufgabe

- Schema für SQL:

Paper(PID, Title, Year)
PaperAuthor(PID, AID)
Author(AID, FirstName, LastName)

- XPath:

```
/paper[year="1986"]/author
```

- SQL:

Aus Daten hergeleitetes Schema

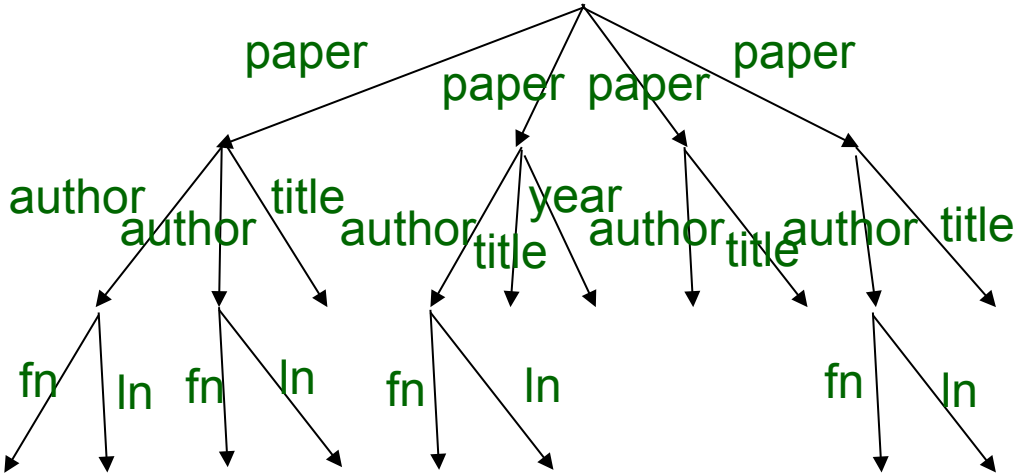
- (Große) XML Dokumente
- Kein Schema bzw. DTD

- Problem: Finde ein “gutes” relationales Schema

- NB: Selbst wenn DTD gegeben ist, kann die Ausdrucksstärke zu gering sein:
  - Z.B. wenn eine Person 1-3 Telefonnummer hat, steht trotzdem: phone*

[Deutsch, Fernandez, Suciu 1999]

Aus Daten hergeleitetes Schema



Paper1

author	fn1	ln1	fn2	ln2	title	year
X	X	X	X	X	X	-
X	X	X	-	-	X	X
X	X	X	-	-	X	-

Paper2

author	title
X	X

Aus Daten hergeleitetes Schema: Aufgabe

- Schema für SQL:

Paper1(fn1, ln1, fn2, ln2, title, year)
Paper2(author, title)

- XPath:

/paper[year="1986"]/author

- SQL:

Pfad-Relations-Methode

- Speicherung von Pfaden als Zeichenketten
- Xpath-Ausdrücke werden durch SQL **like** umgesetzt (vgl. auch **contains**)
- Das Prozentzeichen '%' steht für eine beliebige Zeichenkette mit 0 oder mehr Zeichen

```
SELECT * FROM Versicherungsnehmer
 WHERE Ort LIKE '%alt%';
```

- Der Unterstrich '_' steht für ein beliebiges einzelnes Zeichen, das an der betreffenden Stelle vorkommen soll.

[M. Yoshikawa, T.Amagasa, T.Shimura, S.Uemura 2001]

Pfad-Relations-Methode

Path

pathID	Pathexpr
1	/bib
2	/bib/paper
3	/bib/paper/author
4	/bib/paper/title
5	/bib/paper/year
6	/bib/book/author
7	/bib/book/title
8	/bib/book/publisher

Ein Eintrag für jeden vorkommenden Pfad
Annahme: Nicht zu viele verschiedene Pfadbezeichner notwendig

Pfad-Relations-Methode

Element

NodeID	pathID	ParentID
1	1	-
2	2	1
3	3	2
4	3	2
5	3	2
6	3	2
7	4	2
8	2	1
. . .		

Eine Eintrag für jeden Knoten in der Datenbasis
Recht große Tabelle (Baum der Höhe h hat max. 2^h Blätter)

Pfad-Relations-Methode

Val
text()

NodeID	Val
3	Smith
4	Vance
5	Tim
6	Wallace
7	The Best Cooking Book Ever
6	3
7	4
8	2
. . .	

Ein Eintrag für jedes Blatt in der Datenbasis
und jedes Attribut
Recht große Tabelle

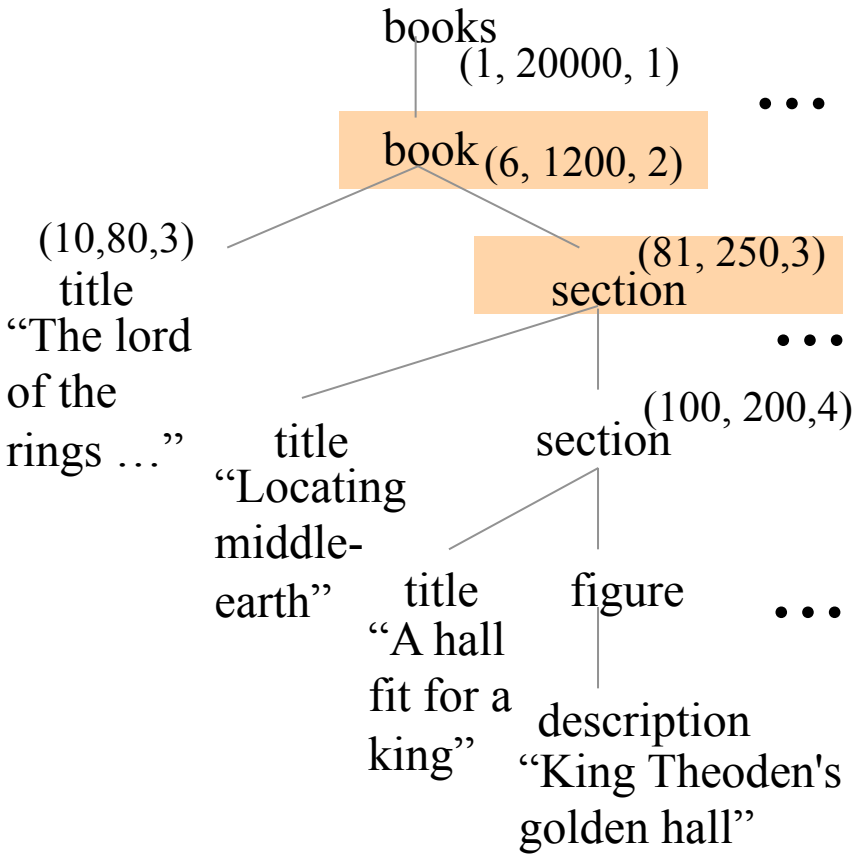
Pfad-Relations-Methode: Aufgabe

- Schema wie oben vereinbart
- XPath: /bib/paper[year="1986"]//figure
- SQL:

Motivation

- Können wir // mit relationalen Techniken besser unterstützen?
  - Bereichsindex (D-Labeling)
  - Pfadindex (P-Labeling)
- Können wir die Anzahl der IO-Operationen reduzieren?
- Können wir Join-Operationen optimieren?

D-Beschriftung: Dynamische Intervallkodierung



- Beschriftung (Start, Ende, Ebene) kann verwendet werden, um Vorgänger-Nachfolger-Beziehungen in einem Baum zu entdecken

D. DeHaan, D. Toman, M. Consens, and M.T. Ozsu.
A comprehensive XQuery to SQL translation using dynamic interval encoding. In *Proceedings of SIGMOD*, **2001**

J. Celko. Trees, Databases and SQL. DBMS, 7(10):48–57, **1994**

D-Beschriftung: Aufgabe

- Schema:

Book(BID, Title, Year, Start, Ende, Ebene
Author(AID, FirstName, LastName
Start, Ende, Ebene)

- XPath:

/book//author

- SQL:

Aufgabe

Book(BID, Title, Year, Start, Ende, Ebene)
Author(AID, FirstName, LastName
Start, Ende, Ebene)

Wozu dient die Ebenen-Angabe?

Aufgabe

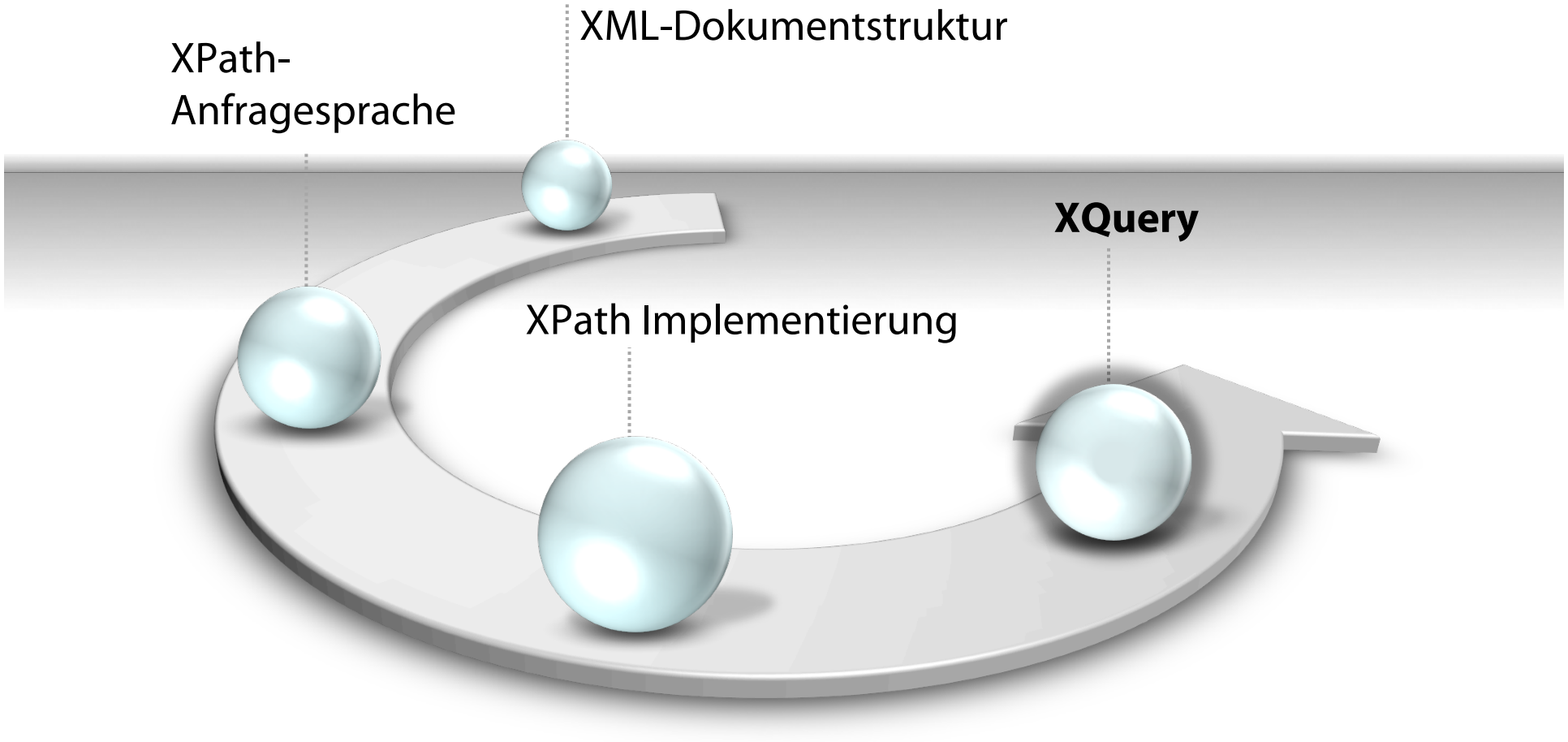
Book(BID, Title, Year, Start, Ende, Ebene)
Author(AID, FirstName, LastName
Start, Ende, Ebene)

Wozu dient die Ebenen-Angabe?

Finden von direkten Nachfahren
("Child"-Relation)

/book/author

Non-Standard-Datenbanken



XQuery: FLWOR (“Flower”) Ausdrücke

for... let... for... let...

where…

order by…

return...

Unser Beispieldokument noch einmal

```
<bib>
    <book>  <publisher> Addison-Wesley </publisher>
            <author> Serge Abiteboul </author>
            <author> <first-name> Rick </first-name>
                    <last-name> Hull </last-name>
            </author>
            <author> Victor Vianu </author>
            <title> Foundations of Databases </title>
            <year> 1995 </year>
    </book>
    <book price="55">
            <publisher> Freeman </publisher>
            <author> Jeffrey D. Ullman </author>
            <title> Principles of Database and Knowledge Base Systems </title>
            <year> 1998 </year>
    </book>
</bib>
```

XQuery: Erstes Beispiel

- Finde alle Buchtitel die nach 1995 publiziert wurden:

```
for $x in doc("bib.xml")/bib/book
where $x/year > 1995
return $x/title
```

- Ergebnis ist eine Liste von XML-Tags (i.a. Bäumen)

```
<title> abc </title>
<title> def </title>
<title> ghi </title>
```

XQuery: RETURN konstruiert Ergebnisliste

- Für jeden Author eines Buches bei “Morgan Kaufmann”, liste alle veröffentlichten Bücher auf

```
for $a in distinct-values (doc("bib.xml")
                          /bib/book[publisher="Morgan Kaufmann"]/author)
return <result>
        {
          $a,
          for $t in /bib/book[author=$a]/title
          return $t
        }
       </result>
```

distinct-values = Duplikateliminierung, dargestellt über eine Funktion

XQuery: Ergebnis ist eine sog. Liste

Ergebnis:

```
<result>
    <author>Jones</author>
    <title> abc </title>
    <title> def </title>
</result>
<result>
    <author> Smith </author>
    <title> ghi </title>
</result>
```

distinct-values

Examples

```
let $in-xml := <in-xml>
                 <a>3</a>
                 <b>5</b>
                 <b>3</b>
               </in-xml>

return
```

XQuery Example	Results
`distinct-values( ('a', 'b', 'a') )`	`('a', 'b')`
`distinct-values( (1, 2, 3) )`	`(1, 2, 3)`
`distinct-values( ('a', 2, 3) )`	`('a', 2, 3)`
`distinct-values( (xs:integer('1'), xs:decimal('1.0'), xs:float('1.0E0') ) )`	`1`
`distinct-values($in-xml/*)`	`(3, 5)`
`distinct-values( () )`	`()`

XQuery: for und let

- for $x in expr -- bindet $x mit jedem Element in der Liste expr

- let $x = expr -- bindet $x an die ganze Liste expr
  - Nützlich für gemeinsame Teilausdrücke oder für Aggregationen

XQuery: Aggregation

```
<big_publishers>
   {
      for $p in distinct-values (doc("bib.xml")//publisher)
      let $b := doc("bib.xml")/bib/book[publisher = $p]
      where count($b) > 100
      return $p
   }
</big_publishers>
```

count = eine Aggregatfunktion zur Bestimmung der Anzahl von Elementen einer Liste

XQuery: where

- Finde Bücher, deren Preis größer als der Durchschnitt ist:

```
let $a=avg(doc("bib.xml")/bib/book/@price)
for $b in doc("bib.xml")/bib/book
where $b/@price > $a
return $b
```

for vs. let

for

- Bindet *Knotenvariablen* → Iteration

let

- Bindet *Listenvariablen* → ein Wert

Zugriff auf Listenelement über Index [.]

let $b := /bib/book

$b[2]

for vs. let

```
for $x in doc("bib.xml")/bib/book
return <result> $x </result>
```

Ergebnis:

```
<result> <book>...</book></result>
<result> <book>...</book></result>
<result> <book>...</book></result>
...
```

```
let $x := doc("bib.xml")/bib/book
return <result> $x </result>
```

Ergebnis:

```
<result> <book>...</book>
         <book>...</book>
         <book>...</book>
         ...
</result>
```

Listen in XQuery

- Geordnete und ungeordnete Listen
  - /bib/book/author = geordnete Liste
  - distinct-values(/bib/book/author) = ungeordnete Liste
- let $a = /bib/book → $a ist eine Liste
- $b/author → eine Liste (mehrer Autoren ...)

```
return <result> { $b/author } </result>
```

Ergebnis:

```
<result> <author>...</author>
         <author>...</author>
         <author>...</author>
         ...
</result>
```

Listen in XQuery

Was ist mit Listen in Ausdrücken?

Let $b = /bib/book

- $b/@price → Liste von n Preisen
- $b/@price * 0.7 → Liste von n Zahlen

- $b/@price * $b/@quantity → Liste von n x n Zahlen??

Konsequenz wäre:

- $b/@price * ($b/@quant1 + $b/@quant2) ≠
  $b/@price * $b/@quant1 + $b/@price * $b/@quant2 !!

Semantik von arithmetischen Operationen zwischen Sequenzen nicht geklärt

Sortierung in XQuery

```
<publisher_list>
{
   for $p in distinct-values (doc("bib.xml")//publisher)
   order by $p/text()
   return <publisher>
                <name>  $p/text()  </name>
               {
                for $b in document("bib.xml")//book[publisher = $p]
                order by $b/@price descending
                return<book>
                             $b/title ,
                             $b/@price
                          </book>

               }
             </publisher>
}
</publisher_list>
```

Fallunterscheidungen

```
for $h in //holding
order by $h/title
return <holding>
          {
             $h/title,

             if ( $h/@type = "Journal" )

             then $h/editor

             else $h/author
          }
          </holding>
```

Existenzquantoren

```
for $b in //book
where some $p in $b//para satisfies
  contains($p, "sailing")
  and contains($p, "windsurfing")
return $b/title
```

Allquantoren

```
for $b in //book
where every $p in $b//para satisfies
   contains($p, "sailing")
return $b/title
```

Funktionen in XQuery

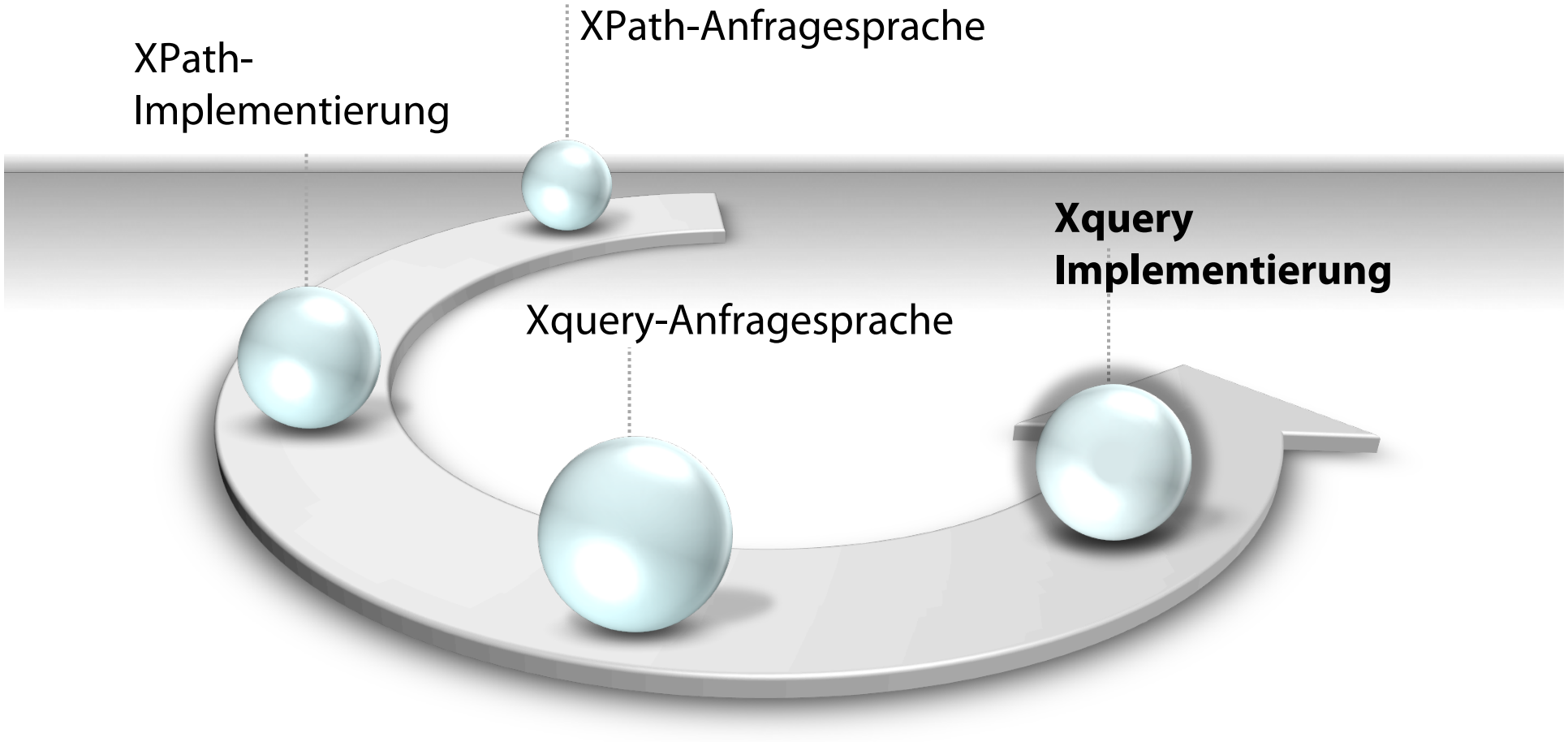
```
declare function reverse ($items) {
  let $count := count($items)
  for $i in 0 to $count
  return $items[$count - $i]
}
```

Beachte: Zugriff auf Listenelemente über Index [..]

XQuery ist Turing-vollständig, also eine Programmiersprache

Non-Standard-Datenbanken

Semistrukturierte Datenbanken



BLAS: An Efficient XPath Processing System

- **B**i-**LA**beling based XPath processing **S**ystem
  - D-Beschriftung (Literatur siehe oben)
    - <Start, End, Ebene> (minimale Angaben)
    - Aufbau eines B-Baums zur Unterstützung von Vorgänger/Nachfolger-Anfragen
  - P-Beschriftung
    - basiert auf XPRESS → eine XML Datenkompressionstechnik, die eine arithmetische Kodierung für Pfadbezeichner verwendet
  - Anfragebeantwortung über komprimierten Dokumenten unter Verwendung der D- und P-Beschriftungen

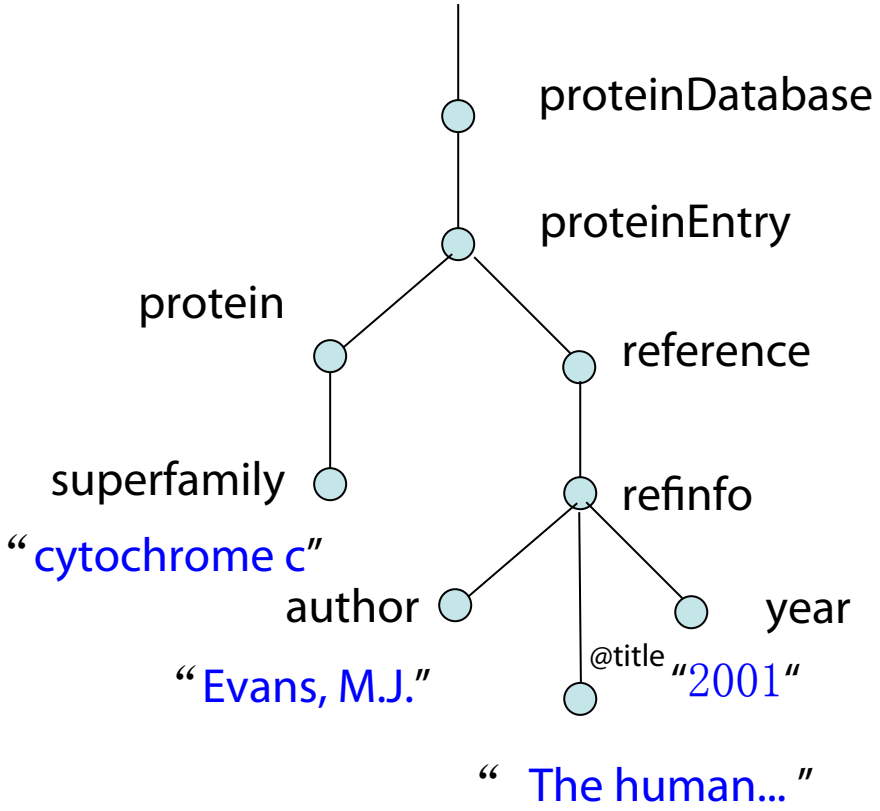
J.-K. Min, M.-J. Park, and C.-W. Chung. XPRESS: A queryable compression for XML data. In Proceedings of SIGMOD, **2003**.

BLAS: An efficient XPath processing system, Y. Chen, S.B. Davidson, Y. Zheng, In Proceedings SIGMOD '04 Proceedings of the 2004 ACM SIGMOD international conference on Management of data, 47-58, **2004**

Baustrukturen: Ein Beispieldokument

```
<proteinDatabase>
   <proteinEntry>
      <protein>
         <Name> cytochrome c [validated]</name>
         <classification>
            <superfamily>cytochrome c</superfamily>
         </classification>…
      </protein>
      <reference>
         <refinfo title="The human somatic cytochrome c gene">
             <authors>
                <author>Evans, M.J.</author>…
             </authors>
             <year>2001</year>
              …
         </refinfo>
         …
      </reference>
      …
   </proteinEntry>
   …
</proteinDatabase>
```

Ein Beispieldokument: Graphische Darstellung



Definitionen

- Die Auswertung eines Pfadausdrucks P (geschrieben [P]) gibt eine Menge von Knoten in einem XML-Baum T zurück, die über P von der Wurzel des Baums T erreichbar sind.
- Pfadausdruck, Pfad und Anfrage werden synonym verwendet
- $P \sqsubseteq Q$ gdw. $[P] \subseteq [Q]$ (Enthaltensein, Containment)
- Disjoint(P, Q) gdw. $[P] \cap [Q] = \varnothing$ (Disjunktheit)

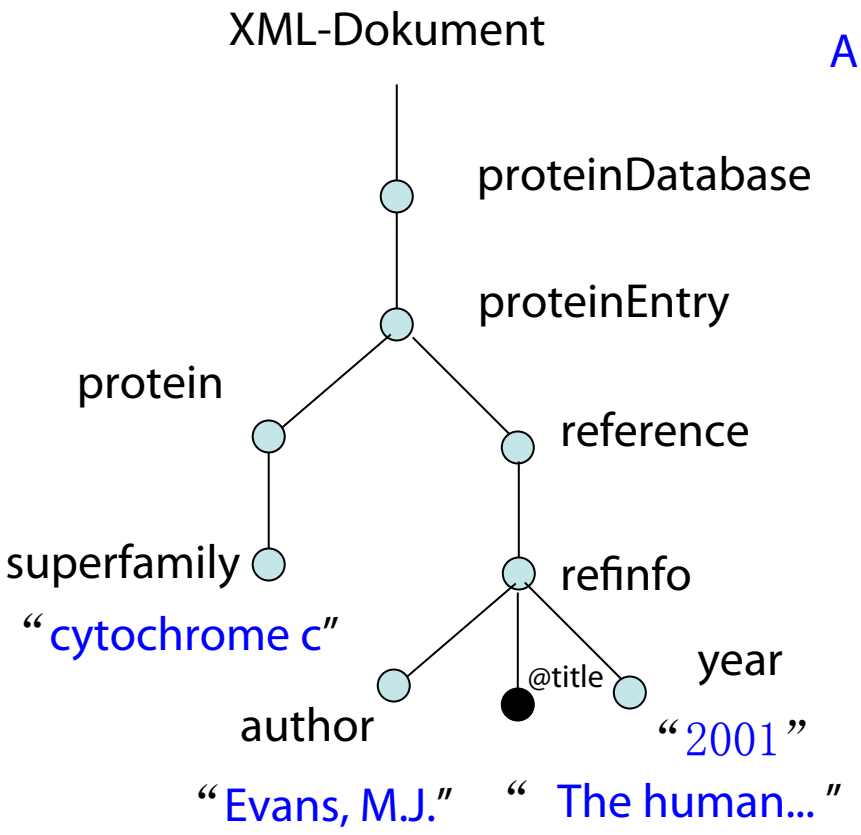
Definitionen (Forts.)

- Suffix-Pfadausdruck: Ein Pfadausdruck P mit einer Folge von Kind-Schritten (/), ggf. mit einem Nachfolger-Schritt (//) am Anfang
- Beispiele:
  - //protein/name
  - /proteinDatabase/proteinEntry/protein/name
- SP(n) : der eindeutige einfache Pfad P von der Wurzel bis zum Knoten n (SP = simple path)
- Die Auswertung eines Suffix-Pfadausdrucks Q ist die Bestimmung aller Knoten n, so dass SP(n) $\sqsubseteq$ Q

D-Beschriftungen

- Tripel $<d_1,d_2,d_3>$ wird für jeden XML-Knoten n vergeben ($n.d_1 \leq n.d_2$)
  - m ist Nachfolger von n gdw. $n.d_1 < m.d_1$ und $m.d_2 < n.d_2$
  - m ist Kind von n gdw.
    m ist Nachfolger von n und $n.d_3+1 = m.d_3$

Beispielanfrage



Anfrage: //proteinDatabase//refinfo/@title

Suche alle Knoten refinfo und proteinDatabase

Seien pDB und refinfo zwei Relationen, die die jeweiligen Knoten repräsentieren, dann D-join ausführen

Select refinfo.title

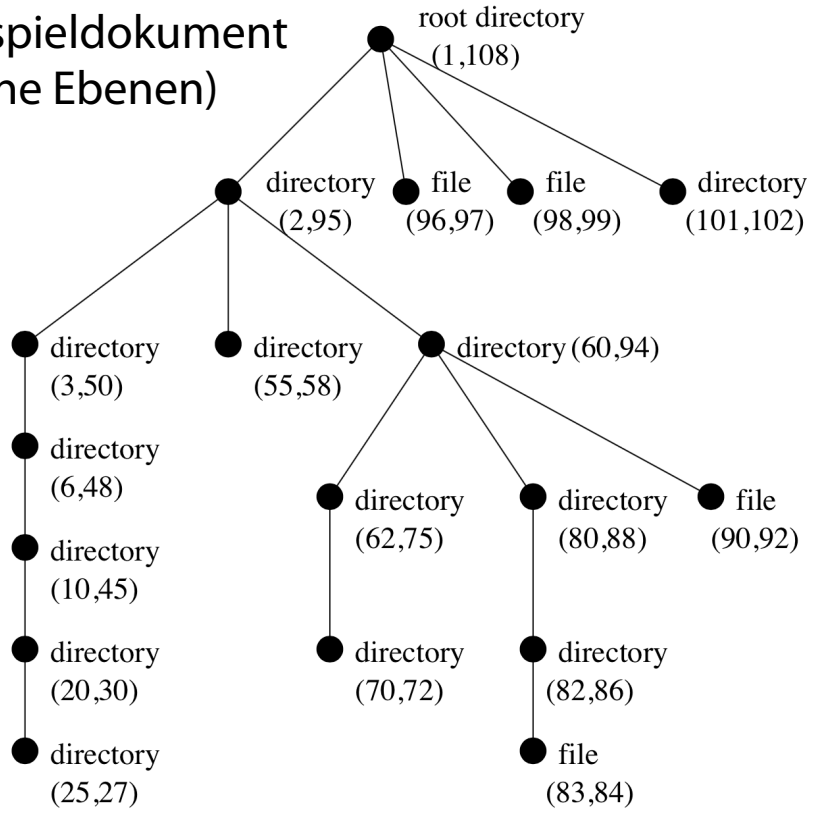
from pDB, refinfo

where pDB.start < refinfo.start and
refinfo.end < pDB.end

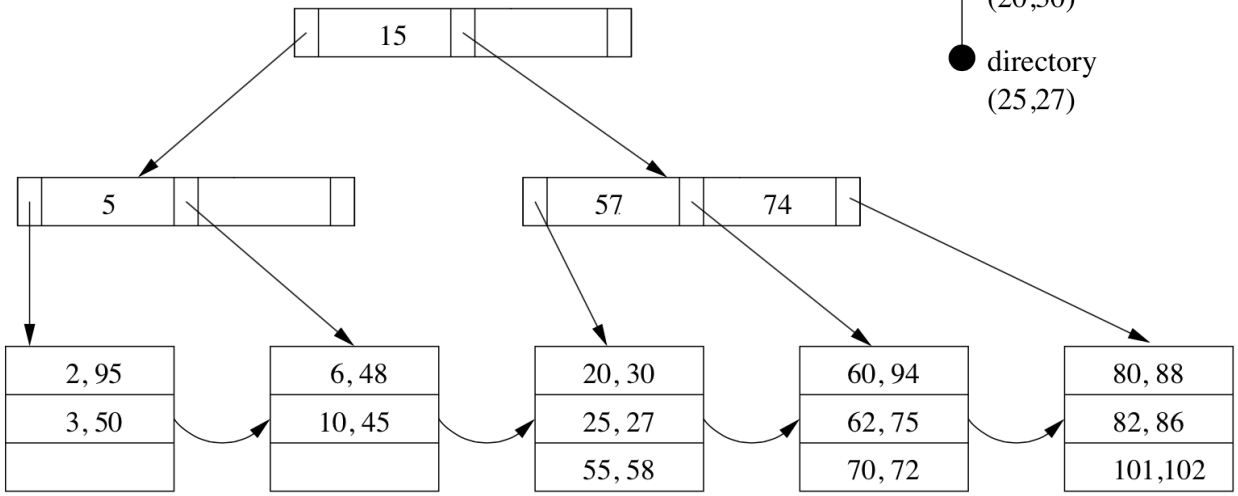
D-Beschriftungen

- B-Bäume als Index für Erreichbarkeits-anfragen
- Nachfolger zuerst?

Beispieldokument (ohne Ebenen)



B-Baum für das Beispieldokument



Vergleiche hierzu: An Evaluation of XML Indexes for Structural Join, H. Li. M.L. Lee. W. Hsu. Ch. Chen, SIGMODRecord, 33(3), Sept. **2004**

Zugriff über B-Baum: Clustered Index

- Blattknoten sind normalerweise nicht in sequentieller Reihenfolge auf der Festplatte gespeichert
- Dieses muss explizit angefordert werden (→ clustered index)

Ning Zhang, Varun Kacholia & M. Tamer Özsu. A Succinct Physical Storage Scheme for Efficient Evaluation of Path Queries in XML. In Proc. ICDE, pages 54–63, **2004**

R. Bayer and E. M. McCreight, Organization and Maintenance of Large Ordered Indexes, Acta Informatica, vol. 1, no. 3, **1972**

P-Beschriftungen

- Kind-Navigation sollte ebenfalls effizient implementiert werden
  - Beispiel /proteinDatabase/proteinEntry/protein/name
- Aufgabe: Verbesserung der "/"-Auswertung für Pfade
- Fokus auf Suffix-Pfadanfragen:
  - Beispiel: //protein/name

P-Beschriftungen

- Weise jedem Knoten n eine Zahl p zu und jedem Suffix-Pfad ein Intervall $[p_1,p_2]$, so dass für Suffix-Pfade Q_1 and Q_2 gilt:
  - $Q_1 \sqsubseteq Q_2$ (Q_1 in Q_2 enthalten) falls $Q_2.p_1 \leq Q_1.p_1$ und $Q_1.p_2 \leq Q_2.p_2$
  - Ein Knoten n ist in einem Suffix-Pfad Q enthalten falls $Q.p_1 \leq SP(n).p_1 \leq Q.p_2$
  - Sei Q ein Suffix-Pfad, dann gilt

    $[Q] = \{n \mid Q.p_1 \leq n.p \leq Q.p_2\}$ wenn $n.p=SP(n).p_1$

P-Beschriftung Beispiel

- Annahme: Längster Pfad: 6
- Wähle maximalen p-Wert m = 10^{12}
- Sei die Maximalanzahl der Auszeichner (tags) auf 99 festgelegt
- Jedem Auszeichner (Tags) wird ein Bereich r zugewiesen: $r_i = 0.01$ (bei 99 Auszeichnern also ausreichend viele)
- P-Beschriftung für jeden Suffix-Pfad bestimmen
- Beispiel P = /ProteinDatabase/ProteinEntry/protein/name

Path expression	P-label
//name	$< 4 \times 10^{10}, 5 \times 10^{10} - 1 >$
//protein/name	$< 4.03 \times 10^{10}, 4.04 \times 10^{10} - 1 >$
//ProteinEntry/protein/name	$< 4.0302 \times 10^{10}, 4.0303 \times 10^{10} - 1 >$
//ProteinDatabase/ProteinEntry/protein/name	$< 4.030201 \times 10^{10}, 4.030202 \times 10^{10} - 1 >$
/ProteinDatabase/ProteinEntry/protein/name	$< 4.030201 \times 10^{10}, 4.03020101 \times 10^{10} - 1 >$

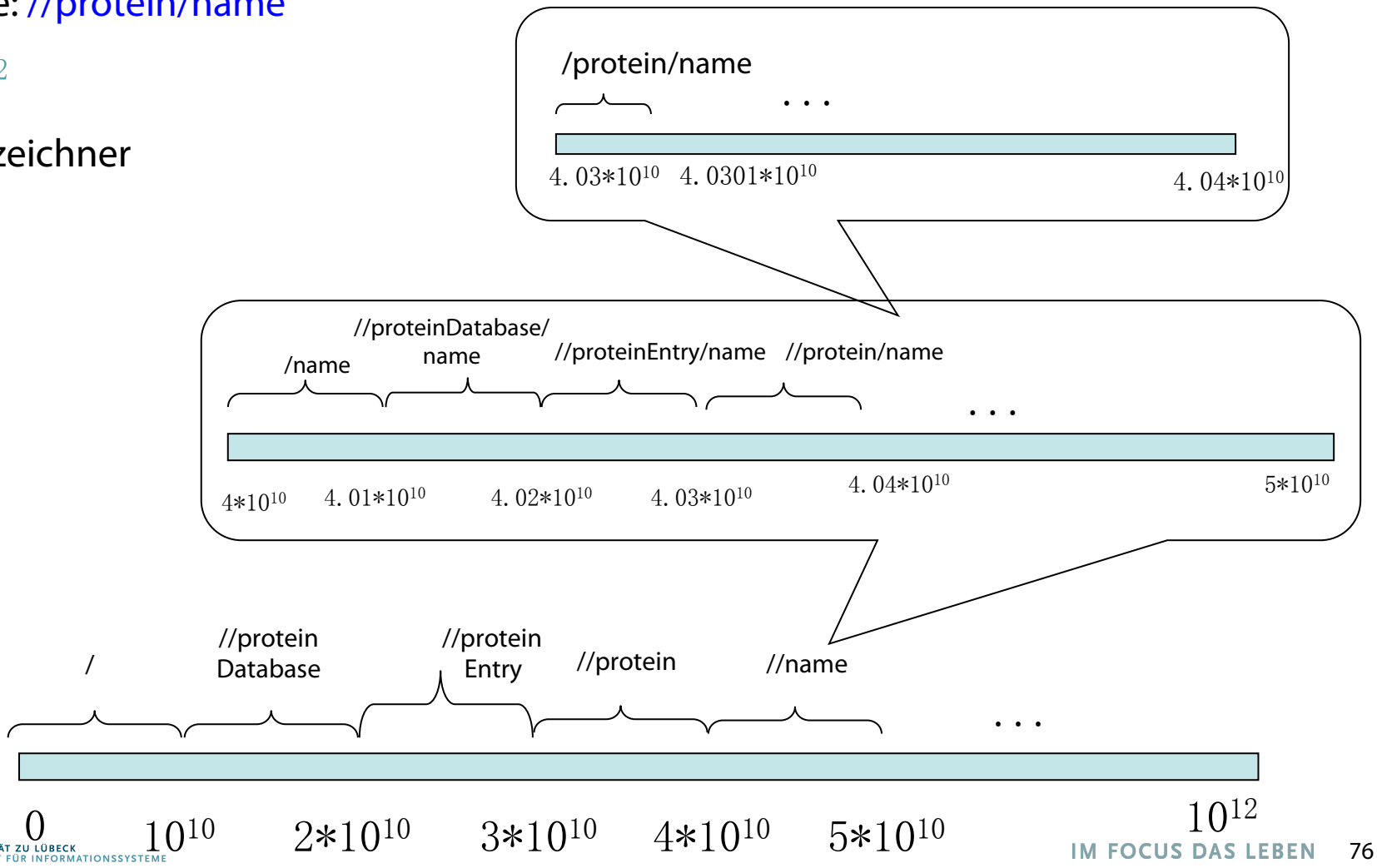
P-Beschriftungen (Beispiel)

Anfrage: //protein/name

m=10^{12}

99 Auszeichner

$r_i = 0.01$

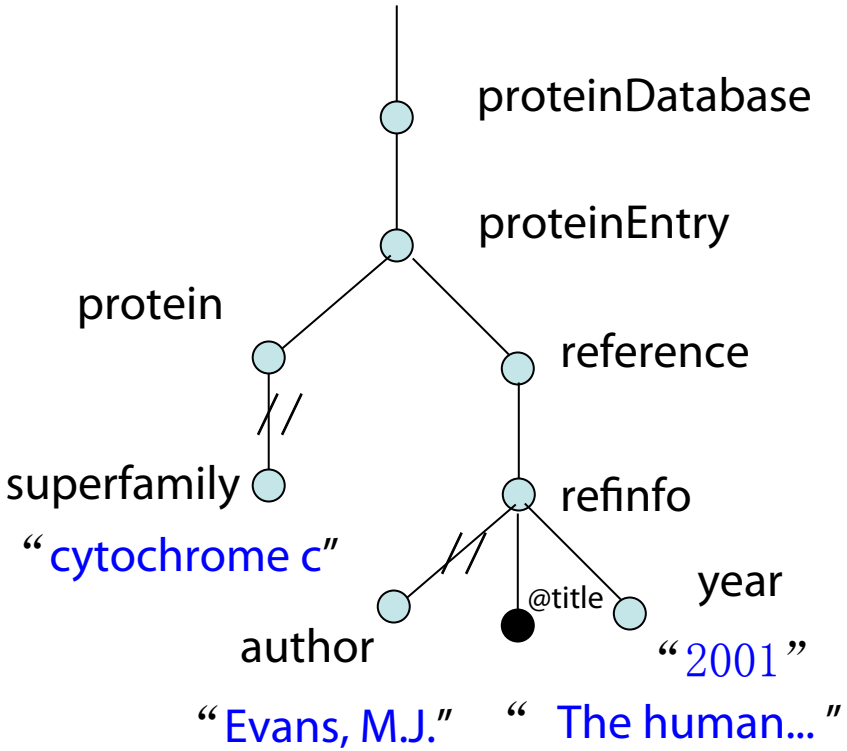


Zum Nachvollziehen zuhause:

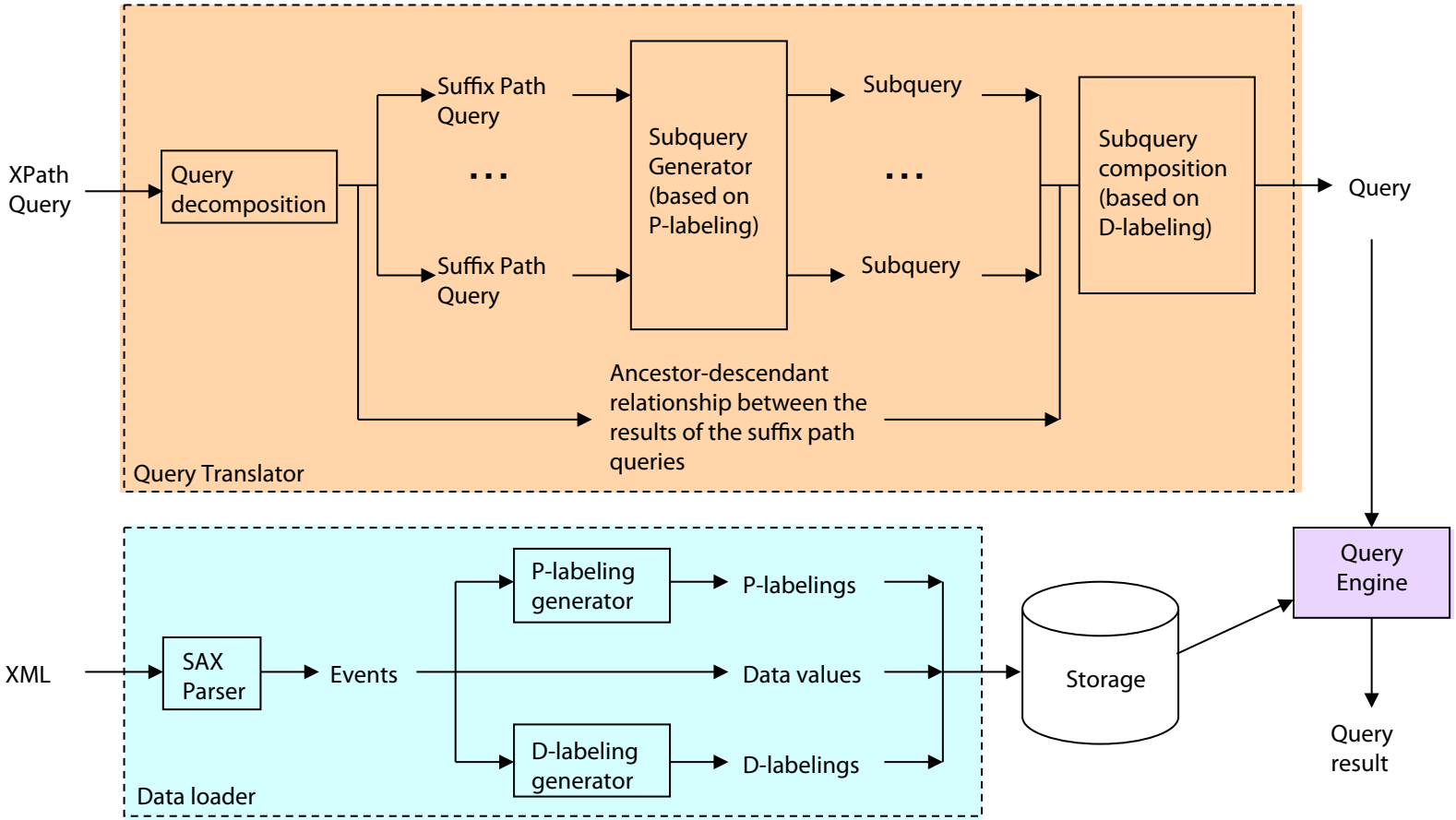
- Seien n Auszeichner gegeben $(t_1,t_2,\ldots.,t_n)$.
- Weise "/" einen Wert r_0 und jedem Auszeichner ti einen Wert r_i zu, so dass $r_0+r_1+r_2+\ldots\ldots+r_n = 1$.
- Setze $r_i = 1/(n+1)$.
- Definiere den Wertebereich der Zahlen in P-Beschriftungen als Integer in [0, m-1], m wird gewählt, so dass $m \geq (n+1)^h$, wobei h der längste Pfad im XML-Baum ist
- P-Beschriftung wie folgt:
  - Pfad // ist ein Intervall (P-label) von <0, m-1> zugeordnet
  - Partitionierung des Intervalls <0, m-1> in der Ordnung der Auszeichnungen proportional zum Abschnitt r_i von t_i, für jeden Pfad $//t_i$ und jeden Abschnitt des Kind-Nachfolgers r_0.
  - Wir allozieren das Intervall $<0, m*r_0 -1>$ für "/" and $<p_i, p_{i+1}>$ für jedes t_i, so dass $(p_{i+1} - p_i)/m=r_i$ und $p_1/m = r_0$

Zweiganfragen (TWIG-Patterns)

Beispielanfrage (ähnlich zu vorigen Daten, beachte aber //)



BLAS Architektur

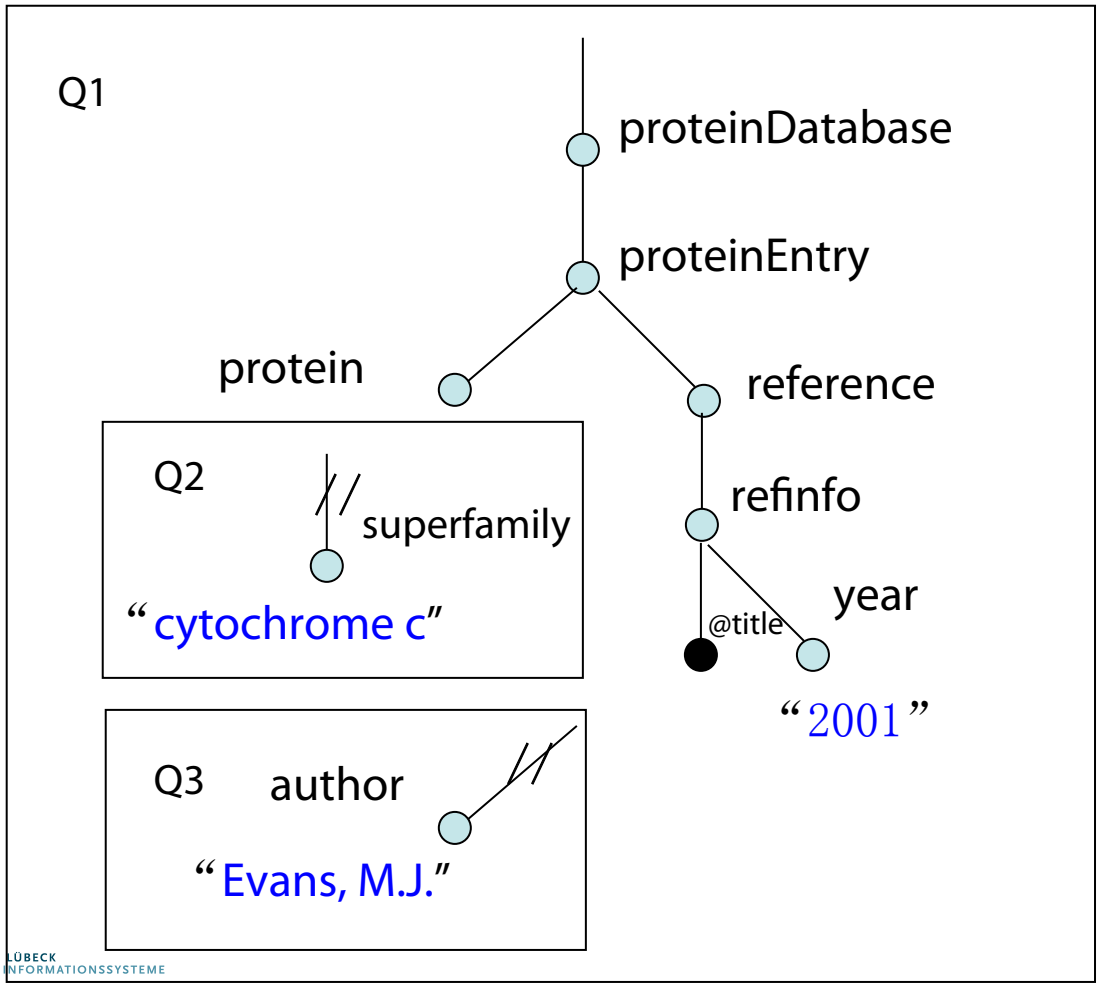


Aufspaltung von Anfragen

- Übersetzung einer XQuery-Anfrage nach SQL
  - **Anfragedekomposition**
    - Aufspaltung einer Anfrage in eine Menge von Suffix-Pfad-Anfragen unter Speicherung von Vorgänger-Nachfolger-Beziehungen
  - **SQL-Generierung**
    - Bestimmung der P-Beschriftung der Anfrage zur Verwendung in einer SQL-Unteranfrage (Bereichsanfrage)
  - **SQL-Komposition**
    - Komposition der Teilanfragen auf Basis der D-Beschriftungen und der Vorgänger-Nachfolger-Beziehung

Ein Beispiel

- Aufspaltung:
  - Verzweigungselimination (B-Eliminierung)



p//q → p and //q

Tiefensuche

Spalte p//q in p and //q

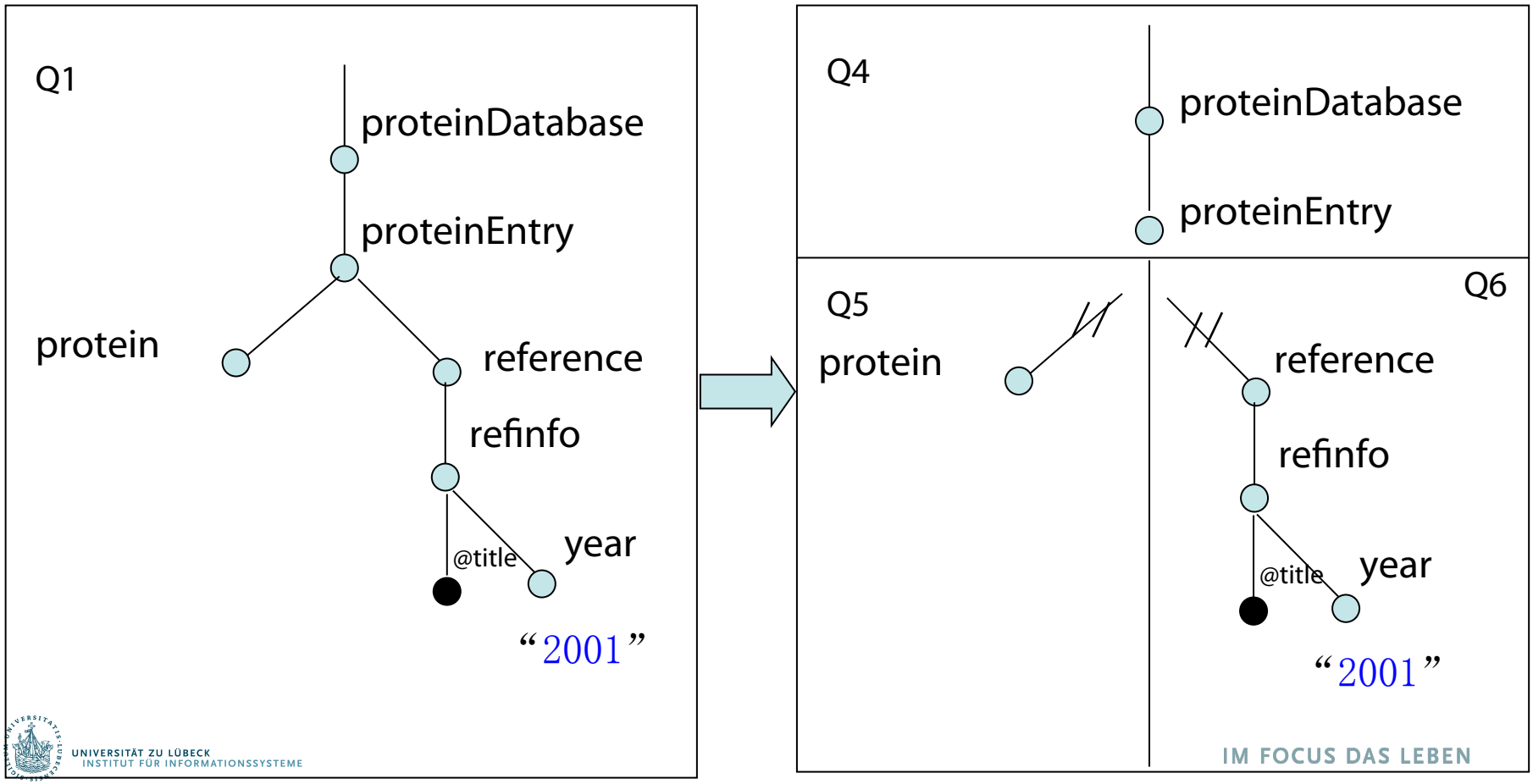
Eliminiere Verzweigung falls vorhanden, sonst evaluiere Q mit P-Beschriftungen

Join für Zwischenresultate unter Verwendung der D-Beschriftungen

Ein Beispiel

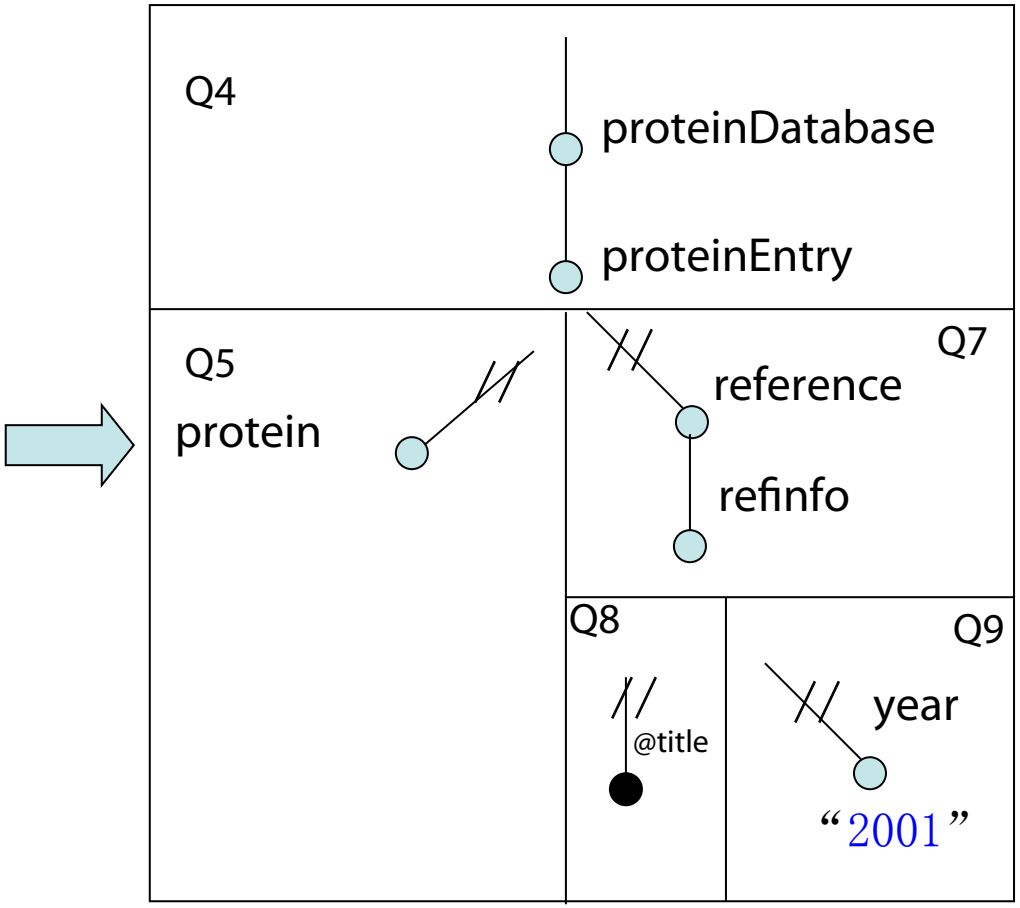
– Verzweigungselimination

P[q1,q2….qi]/r → p, //q1, //q2,…..,//qi, //r



Ein Beispiel

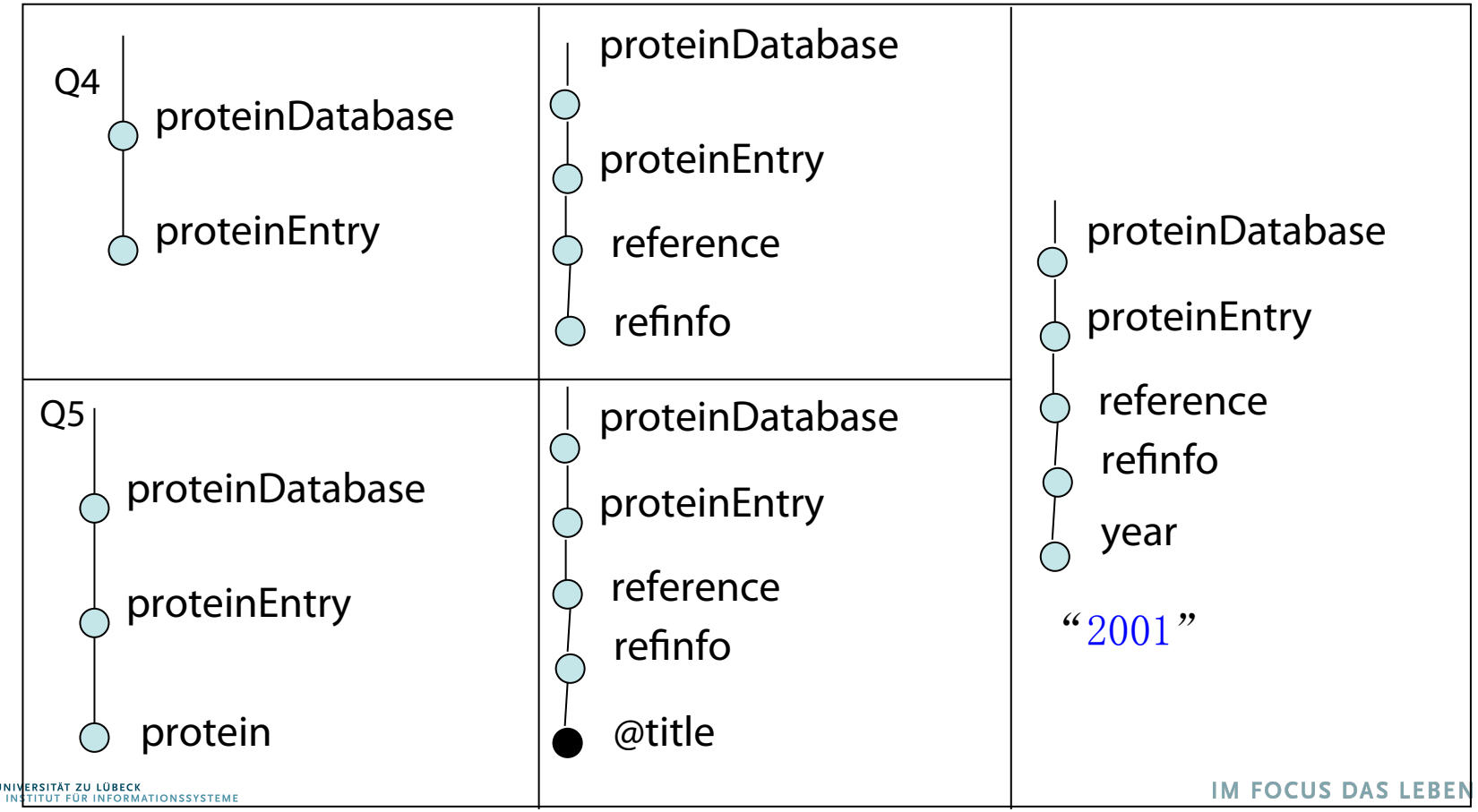
Verzweigungselimination



BLAS: Verfeinerung 1

Push up-Algorithmus: Optimierung der Verzweigungselimination

Da p/qi und p/r spezieller als //qi und //r,
zerlege P[q1,q2,….,qi]/r → p, p/q1, p/q2, …..p/qi, p/r



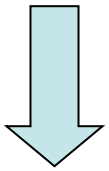
BLAS: Verfeinerung 2

Entfaltungsalgorithmus als Nachfolger-Eliminierung (D-Eliminierung)

p//q → p/r1/q, p/r2/q, ….., p/ri/q

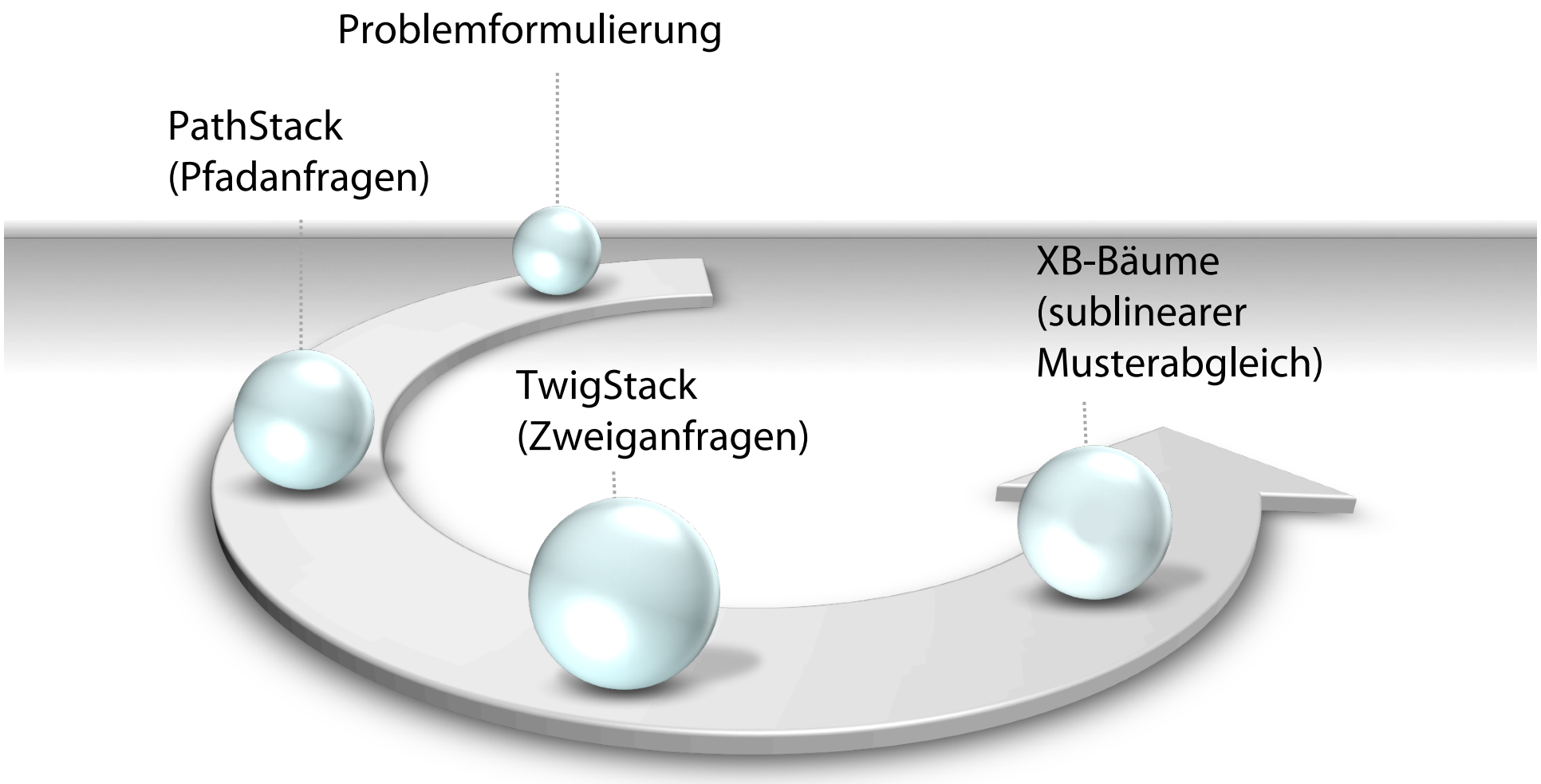
Am Beispiel:

Q2=/ProteinDatabase/ProteinEntry/protein//
superfamily="cytochrome c"



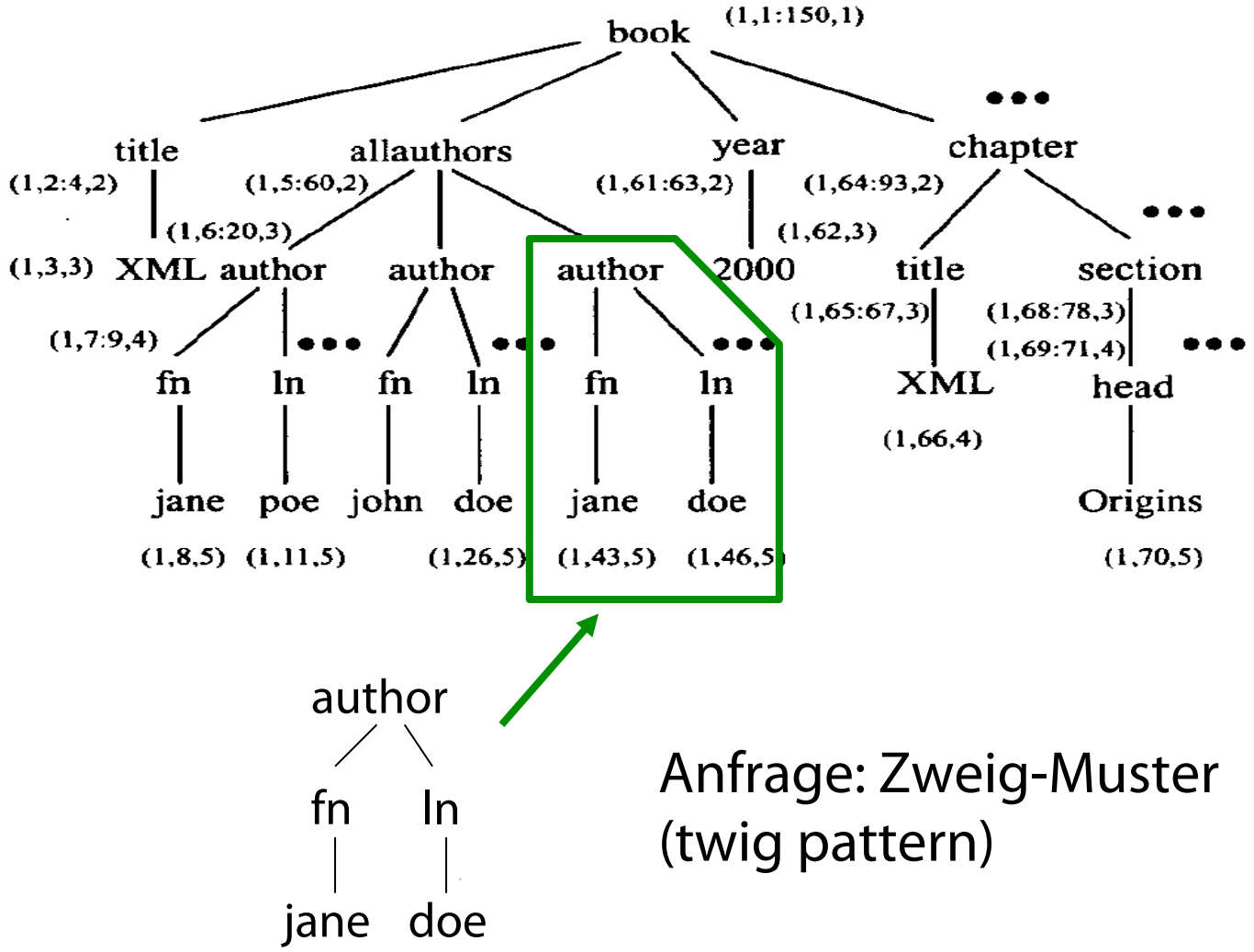
Q21 = /ProteinDatabase/ProteinEntry/protein/classification/
superfamily="cytochrome c" ,

Zweig-basierte Anfragen direkt implementiert



Nicolas Bruno, Nick Koudas, Divesh Srivastava, Holistic twig joins: optimal XML pattern matching, In Proceedings of the 2002 ACM SIGMOD international conference on Management of data (SIGMOD '02), pp. 310-321, **2002**

Beispiel



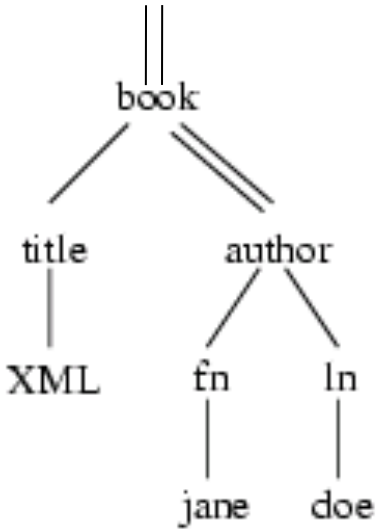
Anfrage: Zweig-Muster (twig pattern)

Zweig-basierte Anfragen direkt implementiert

Beispiel:

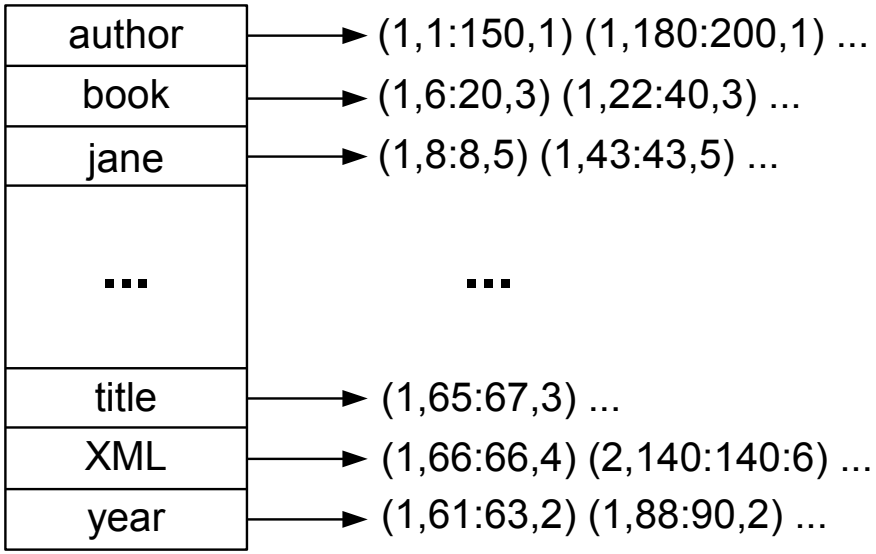
Finde das Publikationsjahr aller Bücher über “XML” geschrieben von “Jane Doe”.

```
for $b in doc(“books.xml”)//book
   $a in $b//author
where contains($b/title, ‘XML’) and
     $a/fn = ‘jane’ and
     $a/ln = ‘doe’
return
   <pubyear> $b/year <pubyear/>
```



Indexierung (D-Beschriftung)

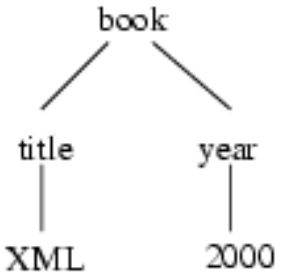
- Elementpositionen durch Tupel repräsentiert (**DocID, Links:Rechts, Ebene**), Sortierung in Überlaufliste nach **Links**
- Kind und Nachfolger-Beziehung bestimmbar



Frühere Ansätze

[Zhang 01] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, G. Lohman, On Supporting Containment Queries in Relational Database Management Systems, In SIGMOD 2001, pp. 425-436, **2001**

- Basierend auf binären Joins [Zhang 01, Al-Khalifa 02].
  - Dekomponieren einer Anfrage in binäre Join-Operationen
  - Bestimme binäre Joins in Bezug auf XML-DB
  - Kombination dieser Basis über D-Join
- Umfassende SQL-Optimierung nötig
  - Zwischenresultate können groß sein

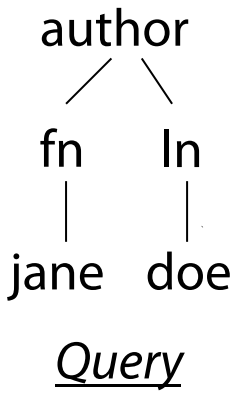


- ((book ⋈ title) ⋈ XML) ⋈ (year ⋈ 2000)
- (((book ⋈ year) ⋈ 2000) ⋈ title) ⋈ XML

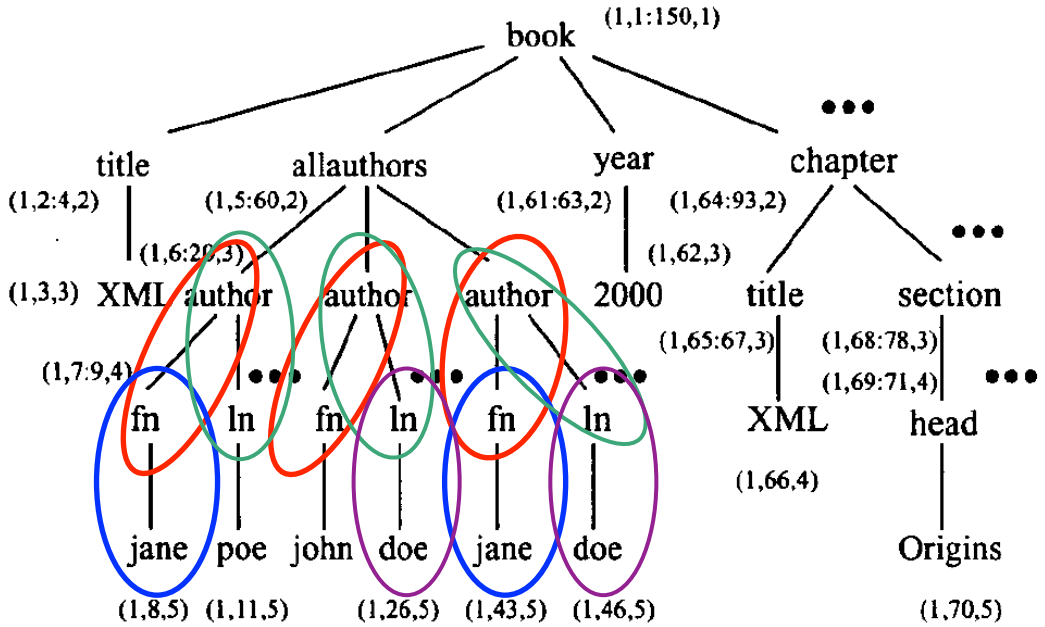
Viele andere Möglichkeiten…

[Al-Khalifa 02] Shurug Al-khalifa, Jignesh M. Patel, H. V. Jagadish, Divesh Srivastava, Nick Koudas & Yuqing Wu. Structural joins: A Primitive for Efficient XML Query Pattern Matching. In Proc. ICDE, pages 141–152, **2002**

Binäre strukturelle Joins



Query



Document

Decomposition

author-fn → **3**

fn-jane → **2**

author-ln → **3**

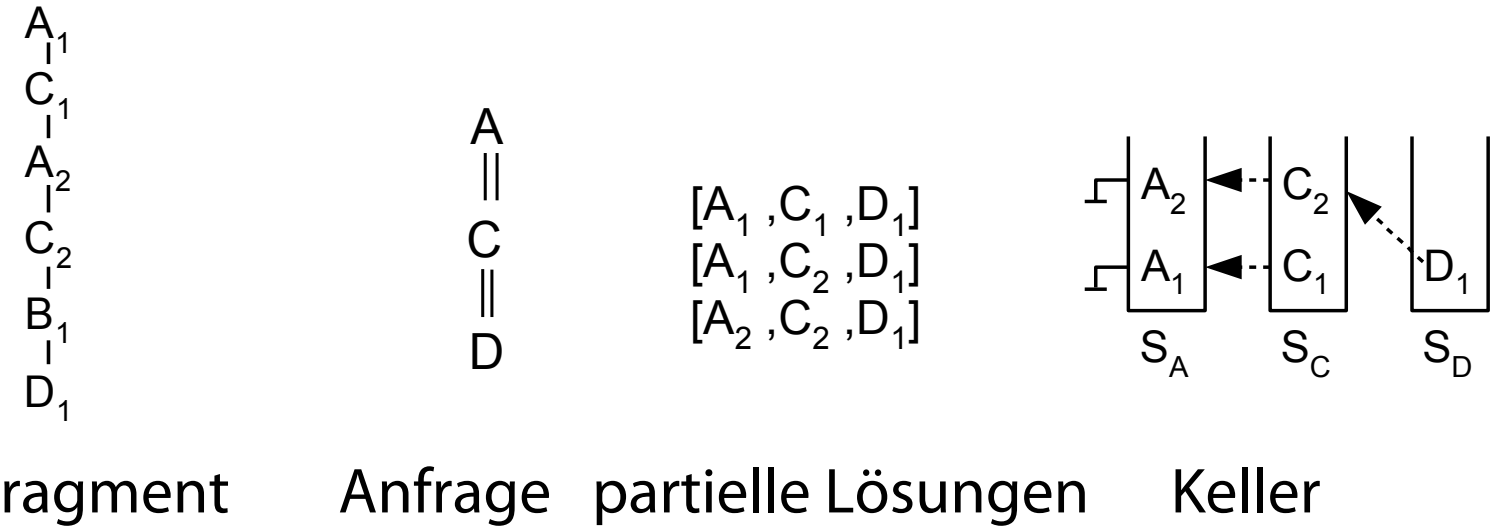
ln-doe → **2**

Neuer Ansatz: Holistische Joins

- Bearbeitung der gesamten Anfrage in 2 Schritten:
  1. Produziere “garantierte” Teilergebnisse in 1. Schritt
  2. Kombiniere (merge join) Teilergebnisse
- Ziel: Teilergebnisse kleiner als Endergebnis.
- Wunsch: Nutze Indexe

Datenstrukturen

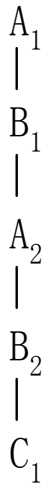
- Assoziiere mit jedem Knoten q in einer Anfrage
  - Einen **Strom** T_q mit den Positionen der Elemente, die zu q korrespondieren, in aufsteigener Links-Ordnung
  - Einen **Keller** S_q mit einer kompakten Kodierung von partiellen Lösungen (Keller sind "verkettet")

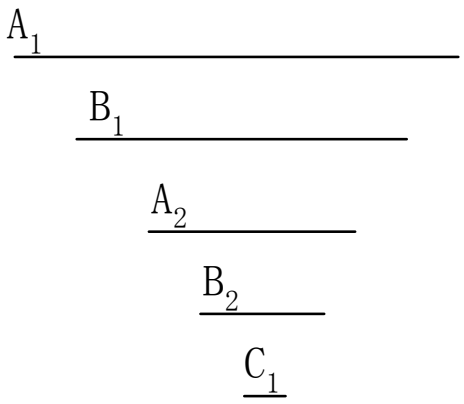


XML-Fragment Anfrage partielle Lösungen Keller

PathStack Beispiel 1

Dokument





Anfrage

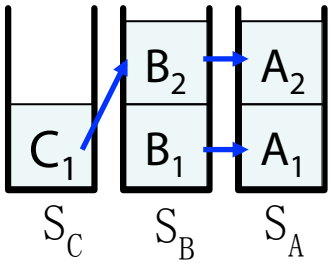
Ströme

T_A: A_1, A_2

T_B: B_1, B_2

T_C: C_1

Keller



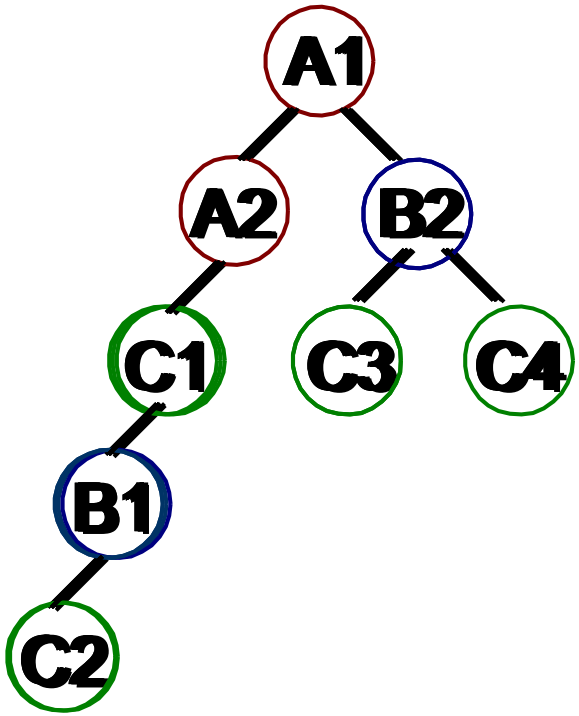
Ausgabe

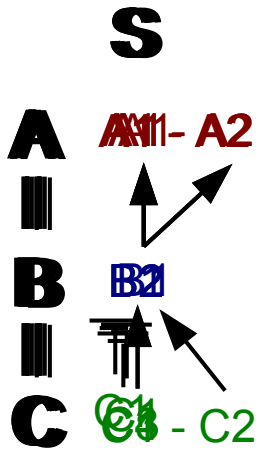
C_1 B_2 A_2

C_1 B_2 A_1

C_1 B_1 A_1

PathStack Beispiel 2



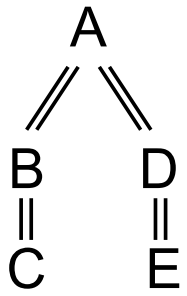


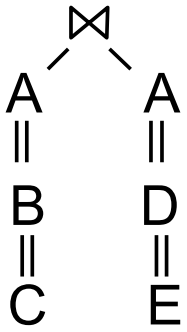
A1,B1,C2
A2,B1,C2
A2,B1,C3
A2,B2,C3
A1,B2,C3
A1,B2,C4

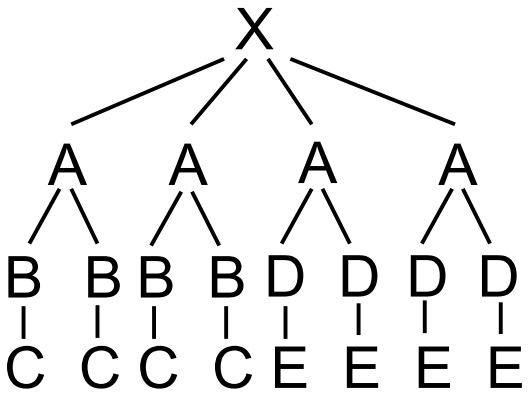
Theorem: PathStack bestimmt Anfrageergebnisse korrekt in O(|input|+|output|) I/O- und CPU-Schritten.

Zweiganfragen

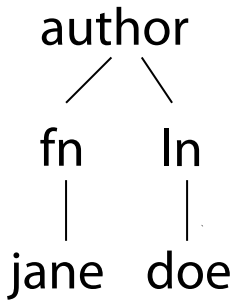
- Naïve Adaptation von PathStack.
  - Bearbeite jeden Pfad isoliert
  - Kombiniere Zwischenergebnisse (merge join)
- Problem: Viele Zwischenergebnisse sind nicht Teil der endgültigen Antwort



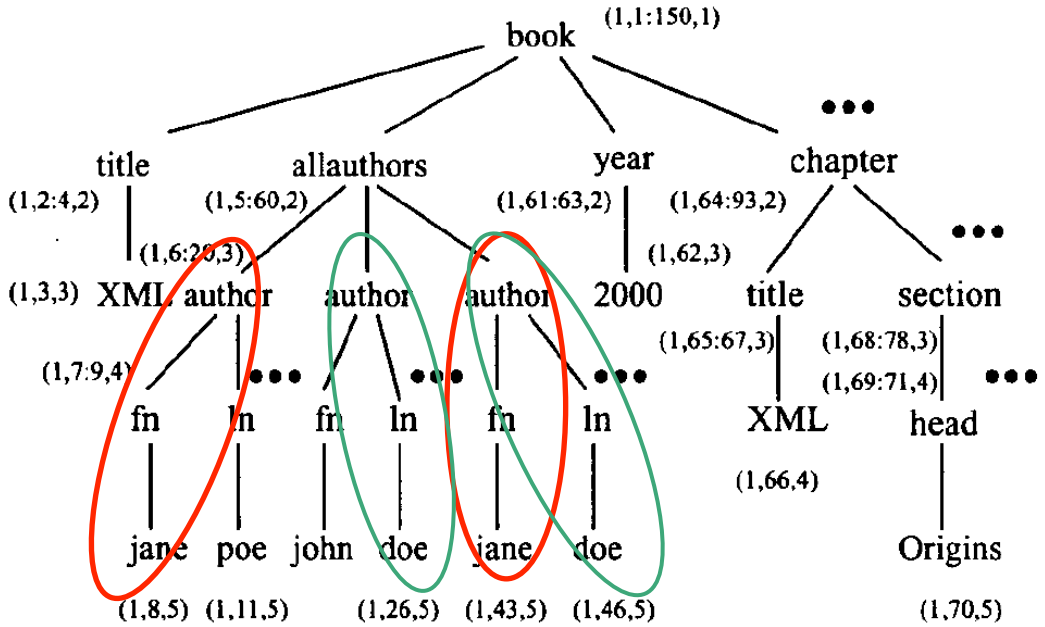




PathStack



Anfrage



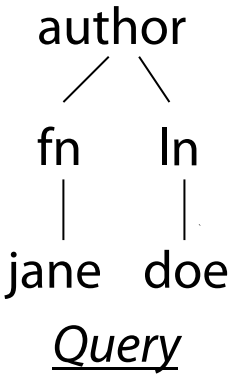
Dokument

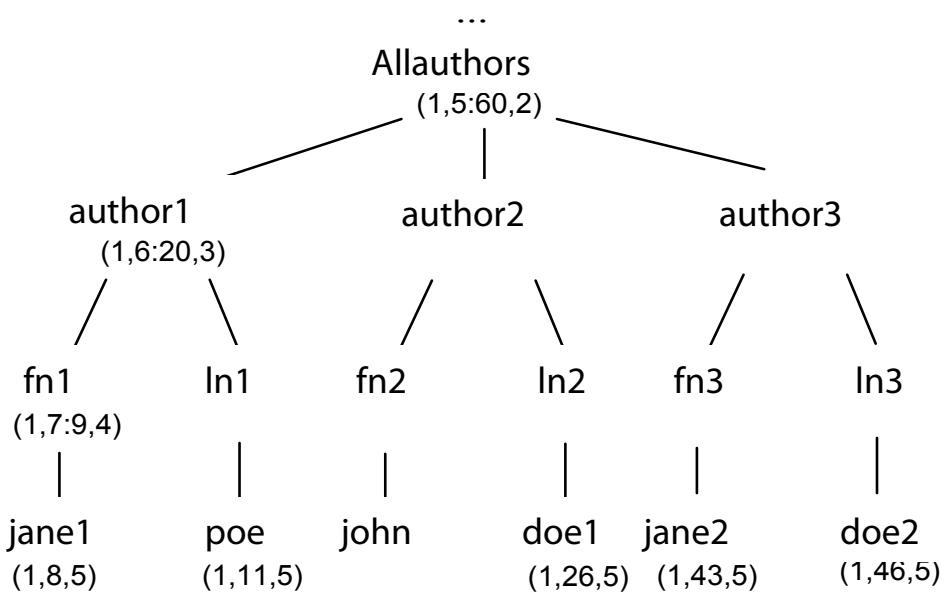
Dekomposition

author-fn-jane → **2**

author-ln-doe → **2**

TwigStack





Document

Streams

T_a: a1, a2, a3

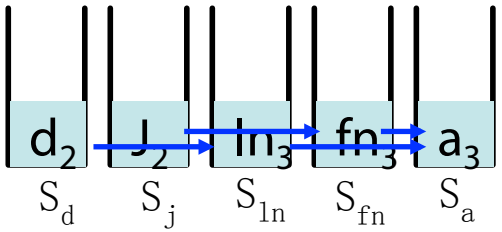
T_{fn}: fn1, fn2, fn3

T_{ln}: ln1, ln2, ln3

T_j: j1, j2

T_d: d1, d2

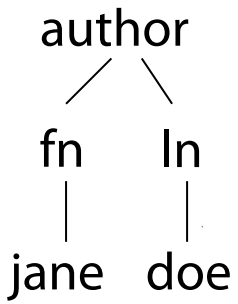
Stacks



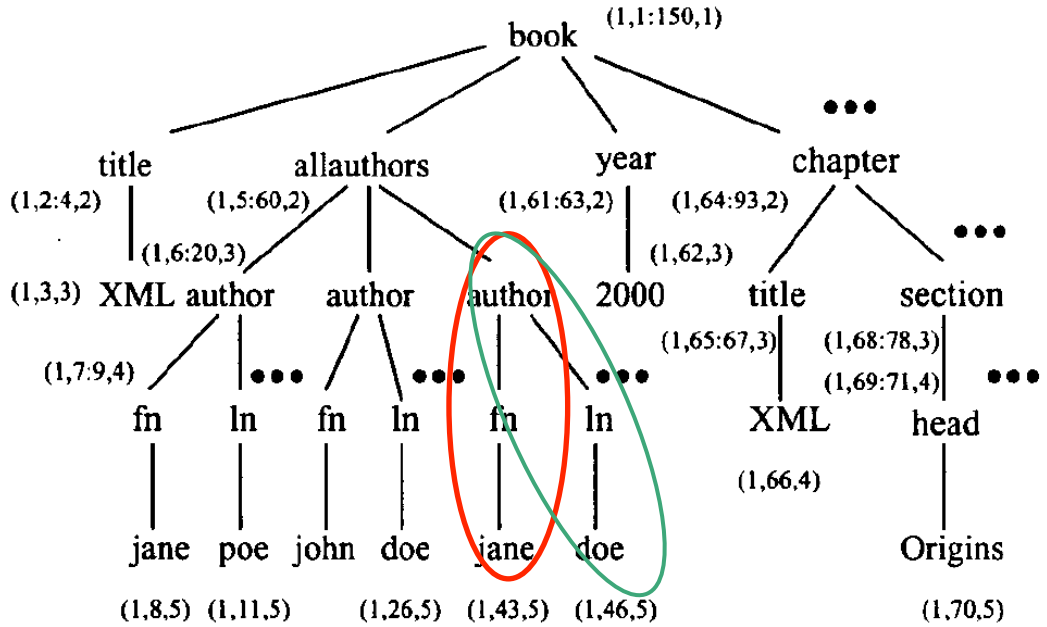
1 -> (j2, fn3, a3)

2 ->(d2, ln3, a3)

TwigStack



Query



Document

Decomposition

author-fn-jane → **1**

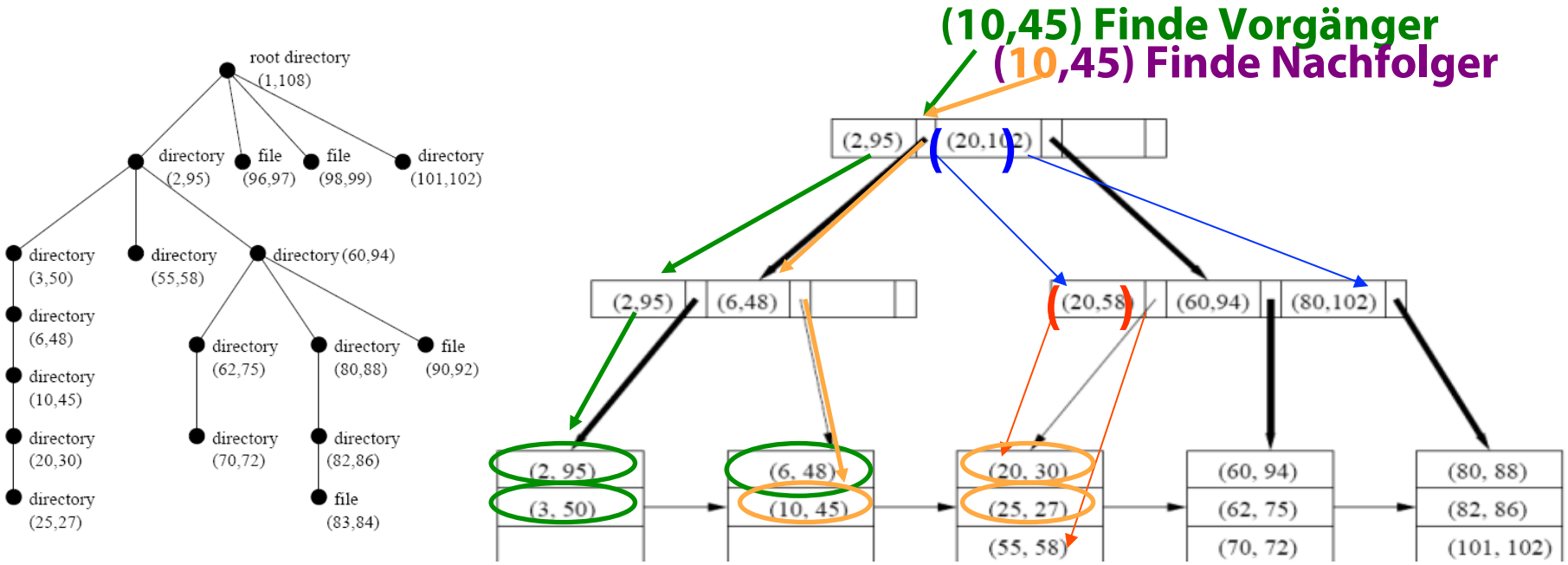
author-ln-doe → **1**

Analyse von TwigStack

- **Theorem**: TwigStack bestimmt Anfrageergebnisse korrekt in O(|input|+|output|) I/O- und CPU-Schritten für Vorgänger/Nachfolger-Beziehungen

XB-Tree

- Verwendung von Indexen zur Effizienzsteigerung
- XB-Bäume sind wie R-Bäume und B-Bäume
  - Interne Knoten haben die Form [L:R], sortiert nach L
  - Eltern-Intervall enthält Kinder-Intervalle



Vergleiche hierzu: An Evaluation of XML Indexes for Structural Join, H. Li. M.L. Lee. W. Hsu. Ch. Chen, SIGMODRecord, 33(3), Sept. **2004**.

Zusammenfassung

- Holistische Pfad-Join-Algorithmen sind unabhängig von der Größe der Zwischenergebnisse der strukturellen binären Joins
- TwigStack generalisiert PathStack für Zweig-Anfragen
- XB-Bäume integriert in TwigStack

Weitere Arbeiten: Anfrageoptimierung

- Automatische Selektion von Indizierungstechniken

  Beda Christoph Hammerschmidt, KeyX: Selective Key-Oriented Indexing in Native XML-Databases, Dissertation, **IFIS Universität Lübeck**, Akademische Verlagsgesellschaft Aka GmbH, DISDBIS 93, **2005**

  Christian Mathis, Storing, Indexing, and Querying XML Documents in Native XML Database Management Systems, Dissertation, Universität Kaiserslautern, **2009**

- Erfüllbarkeits- und Enthaltenseins-Tests unter Berücksichtigung von XML-Schema-Spezifikationen

  Jinghua Groppe, Speeding up XML Quering: Satisfiability Test & Containment Test of XPath Queries in the Presence of XML Schema Definitions, Dissertation, **IFIS Universität Lübeck**, dissertation.de: Verlag im Internet GmbH, **2008**

XML in verteilten Systemen (IFIS / ITM)

- Neue Optimierungstechniken für XML-Anfragen bei Datenverteilung in großen Sensornetzwerken (Kommunikation energieaufwändig)
- Caches in der Nähe von Datenquellen zur Vermeidung von Kommunikation und Verifikation der Cache-Kohärenz bei nicht-zuverlässigen Kommunikationskanälen

Nils Höller, Efficient XML Data Management and Query Evaluation in Wireless Sensor Networks, Dissertation **IFIS Universität Lübeck**, **2010** (vgl. auch das IFIS/ITM-Projekt AESOP's TALE)

Hauptansätze zur XQuery-Implementierung

- Abbildung auf SQL
  - Beispiel: BLAS
- Spezielle Datenstrukturen und Algorithmen
  - Beispiel: Holistic Twig Joins

Nicht nur TWIG-Ausdrücke…

```
for $b in //book
where some $p in $b//para satisfies
  contains($p, "sailing")
  and contains($p, "windsurfing")
return $b/title
```

Non-Standard-Datenbanken

Zusammenfassung und Ausblick