Non-Standard-Datenbanken und Data Mining

Semistrukturierte Datenbanken und Volltextindizierung

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

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>
```

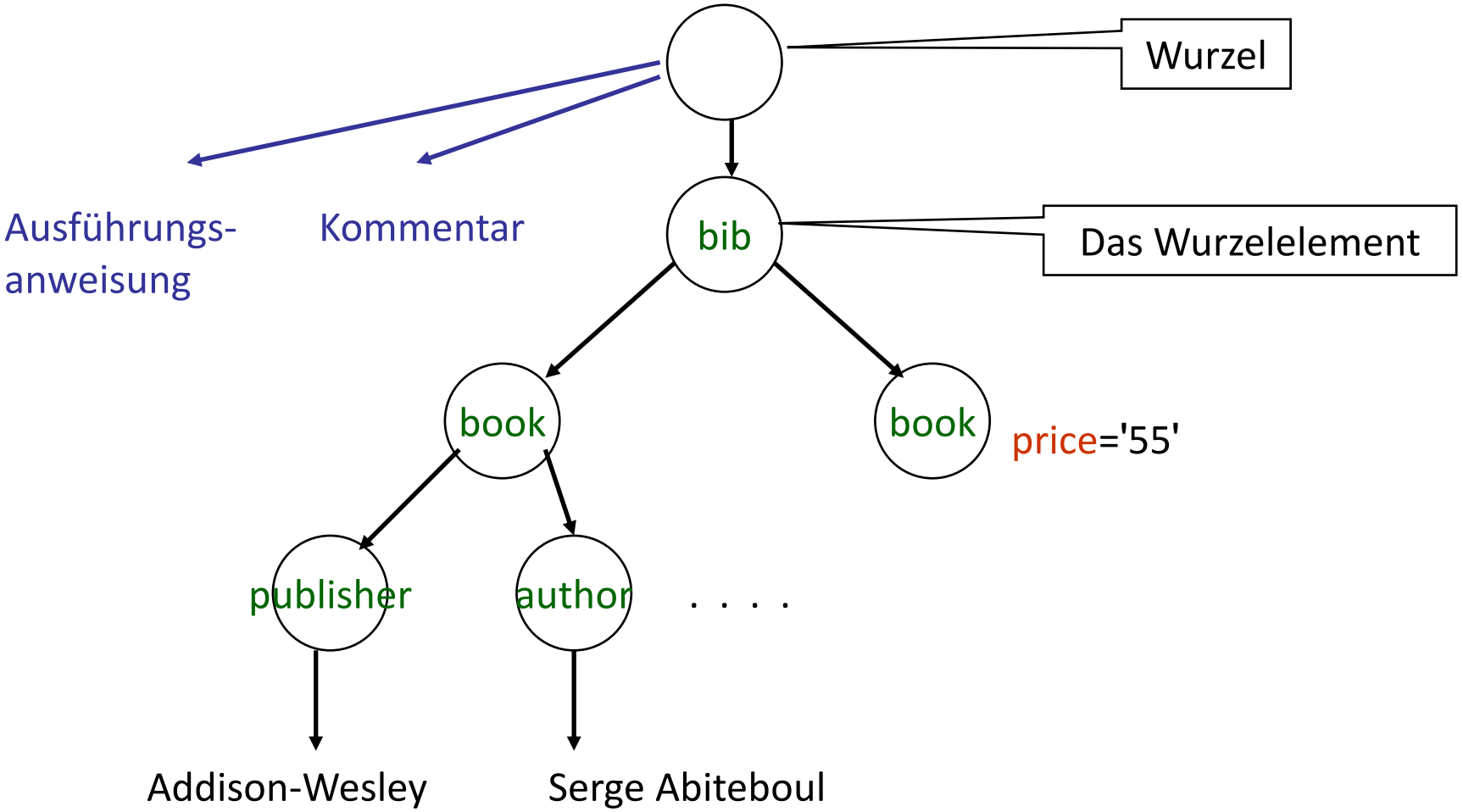
Semistrukturierte Datenbanken: Überblick

- **Datenspezifikationssprache XML**
  - Gedacht zur Datenkommunikation und –integration
  - Begriff des XML-“Dokuments“
  - Strukturprüfung möglich (DTD, XML-Schema)
  - Repräsentation von XML-Daten in SQL möglich?

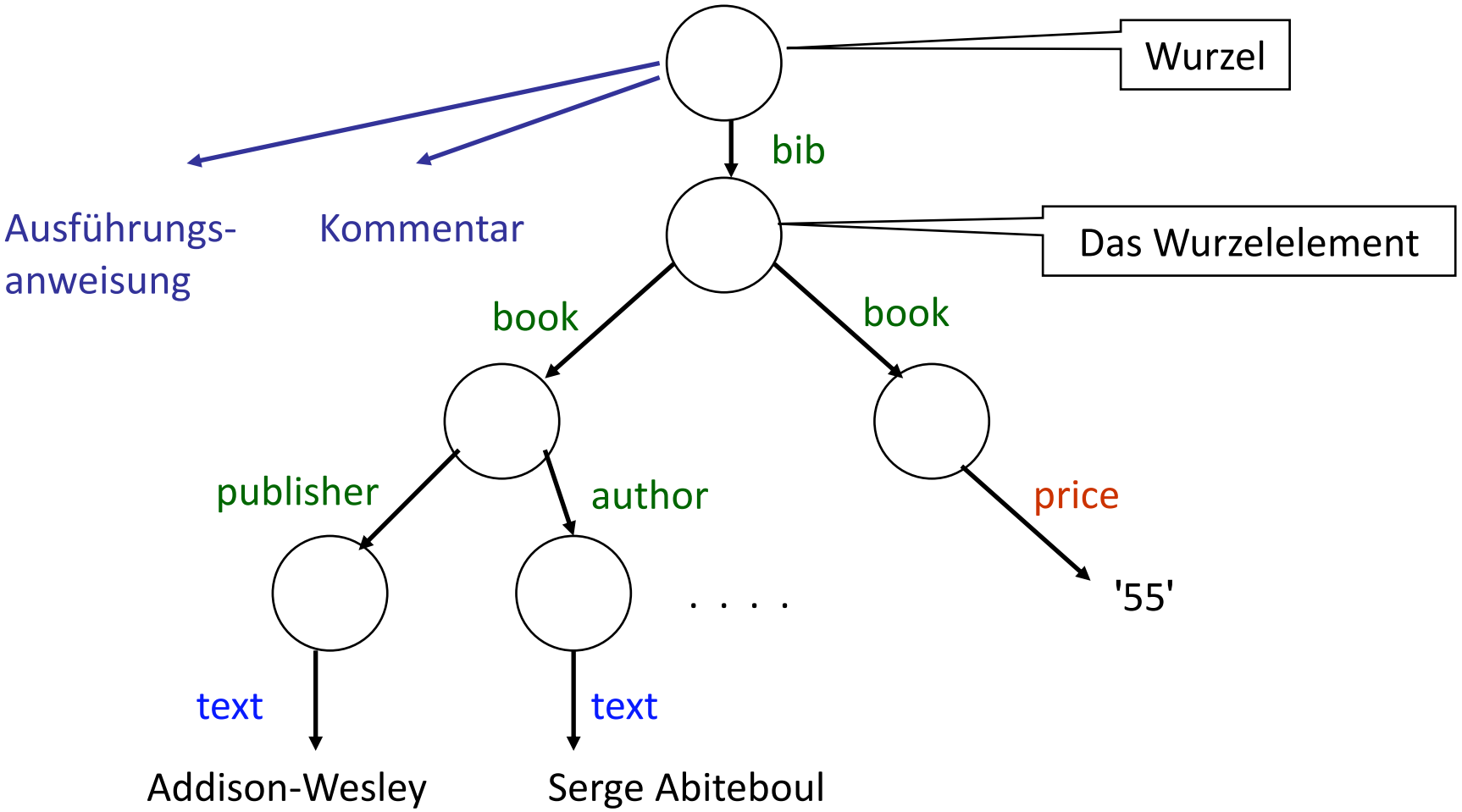
- **Anfragesprache XPath**
  - W3C-Standard: http://www.w3.org/TR/xpath (11/99)
  - Übersetzung nach SQL möglich?
  - Einige Aspekte der optimierten Anfragebeantwortung

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

Datenmodell für XPath



Datenmodell für XPath: Duale Sicht



Dokumenttypdefinitionen (DTDs)

- Beschreibung von akzeptierten Strukturen (optional)
- Kontextfreie Grammatik

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

- Siehe auch XML-Schema
  (ausdrucksmächtiger als DTDs,
  z.B. Minimal- und Maximalangaben von Nachfolgern)

XPath: Einfache Ausdrücke

/bib/book/year

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

/bib/paper/year

Ergebnis: empty (keine Papiere in der bib)

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']

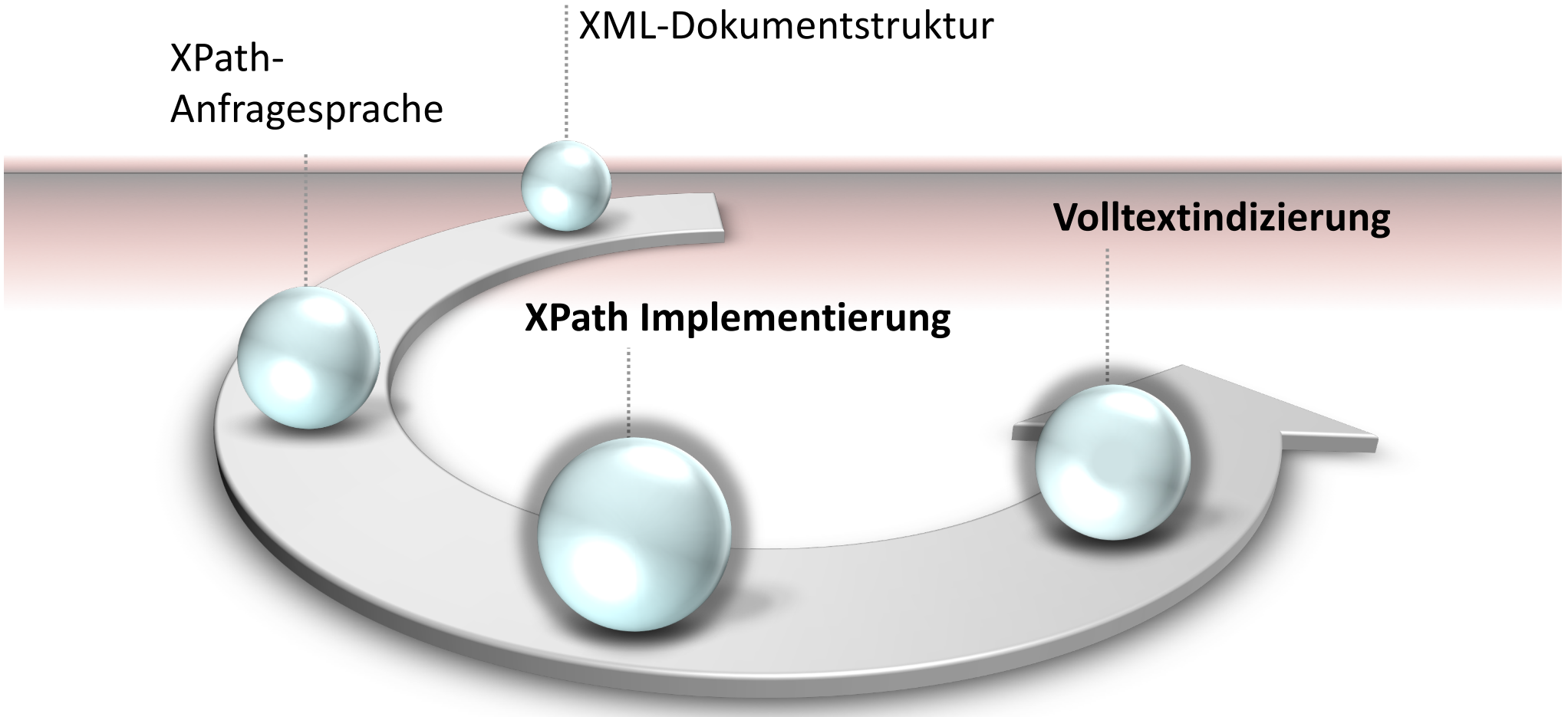
/bib/book[author/@age < '25']

/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 …

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

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**.

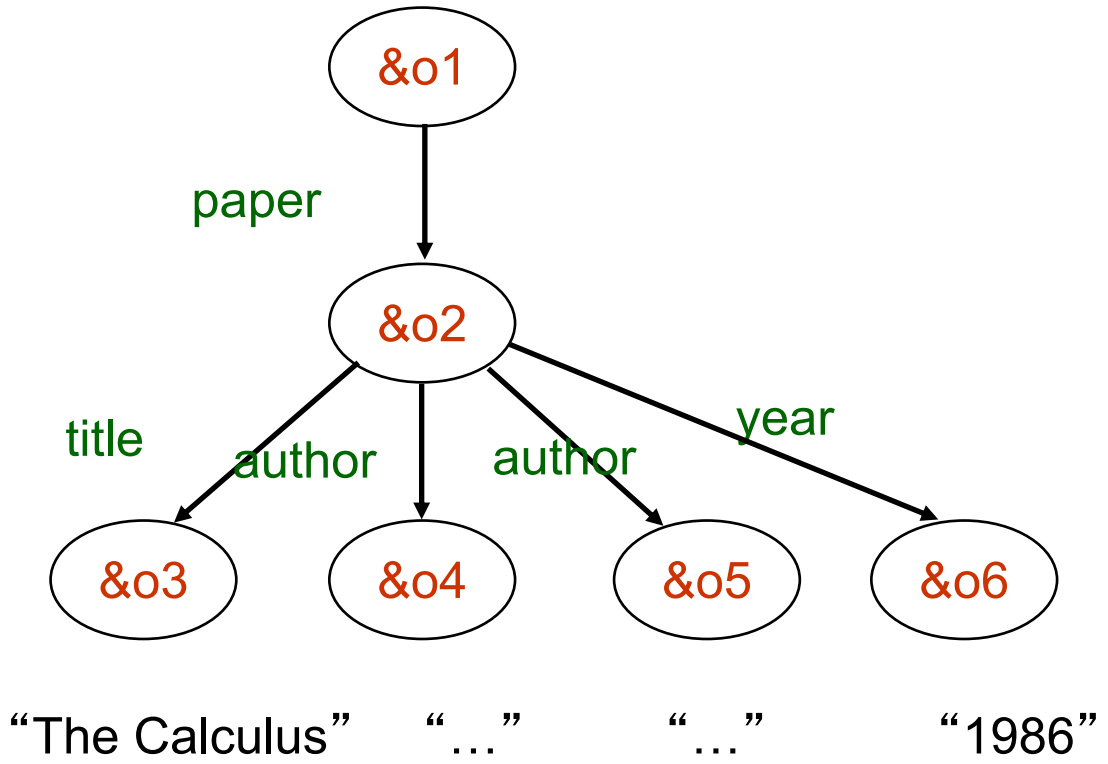
- Verwendung einer sog. Pfad-Relation

  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)

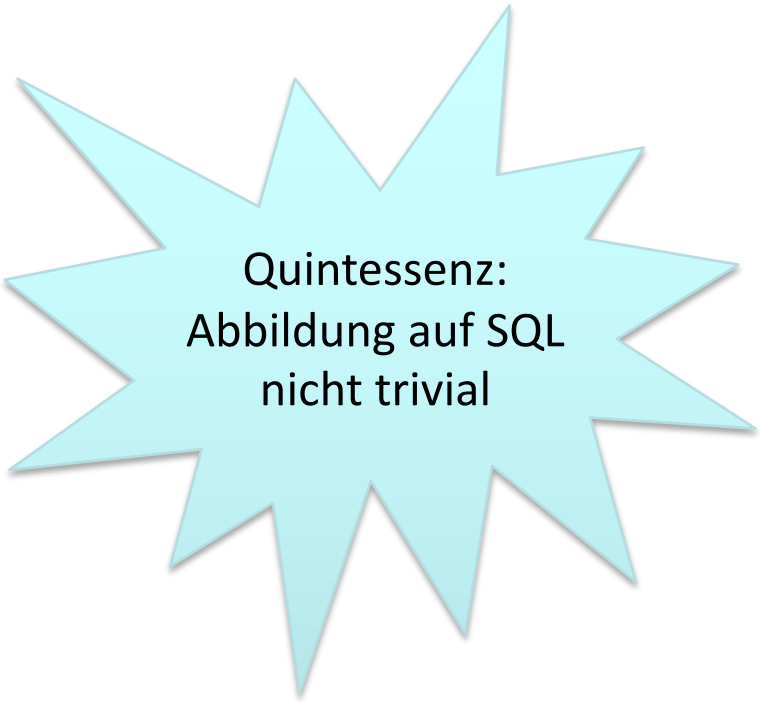
RefTag2(Source, Dest)

…

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

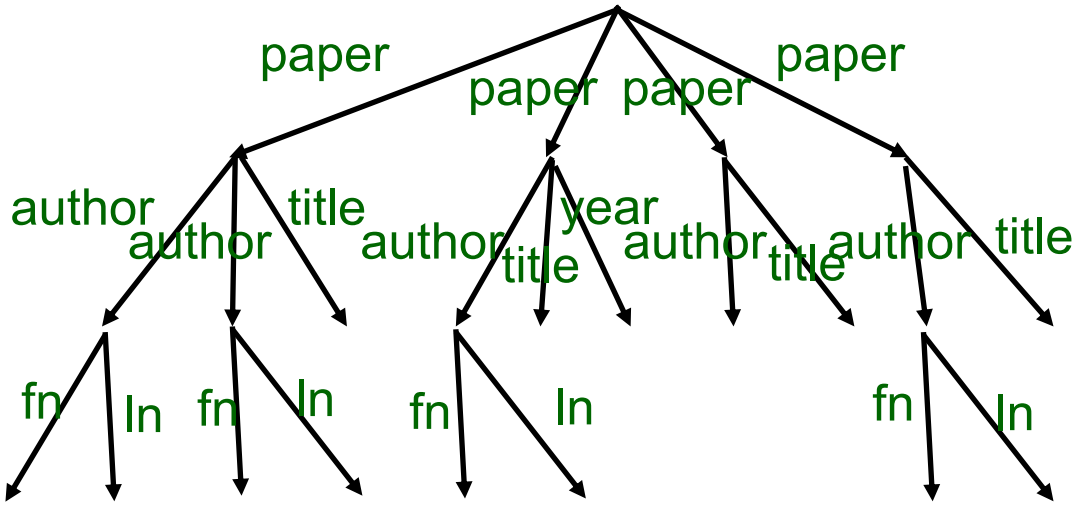
- (Große) XML Dokumente
- Kein Schema bzw. DTD

- 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(author, 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

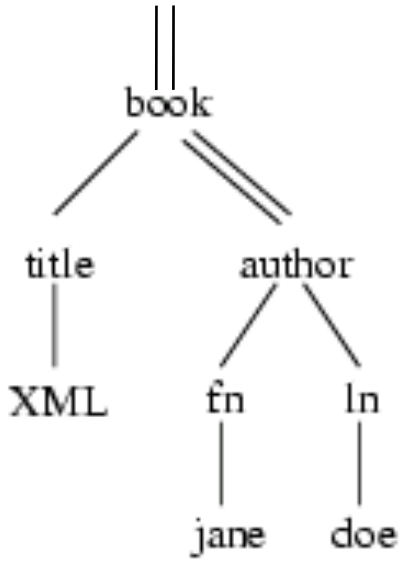
- Schema wie oben vereinbart
- XPath: /bib/paper[year='1986']//figure
- SQL:

XPath vs. XQuery

- 13 Navigationsmöglichkeiten in XPath:
  nicht nur Nachfolger
- XQuery: Nicht nur Pfad-,
  sondern Zweig-basierte Anfragen:

Finde das Publikationsjahr
aller Bücher

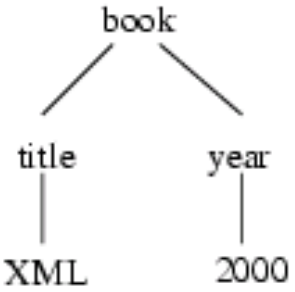
über “XML” geschrieben
von “Jane Doe”.



Frühere Ansätze

- Basierend auf binären Joins [Zhang 01, Al-Khalifa 02].
- Umfassende SQL-Optimierung nötig
  - Zwischenresultate können groß sein
  - SQL-Maschinen überfordert

Join mit Values-Tabelle



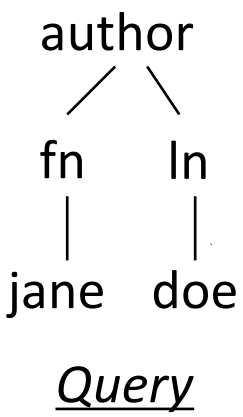
- ((book ⋈ title) ⋈ XML) ⋈ (year ⋈ 2000)
- (((book ⋈ year) ⋈ 2000) ⋈ title) ⋈ XML

Viele andere Möglichkeiten…

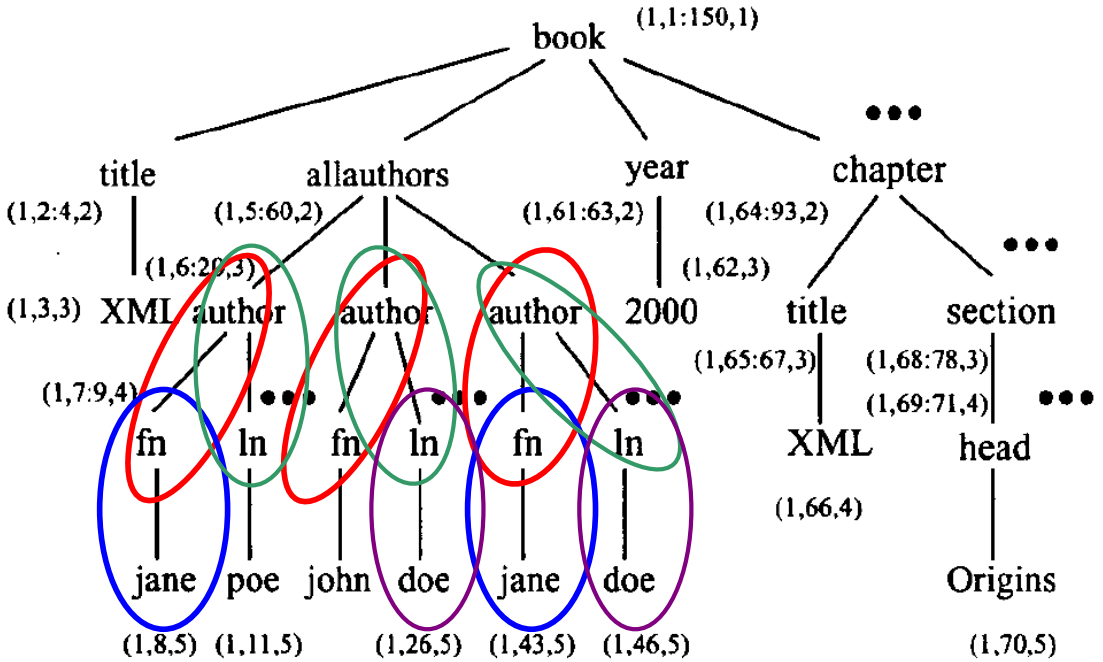
[Zhang 01] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, G. Lohman, On Supporting Containment Queries in Relational Database Management Systems, In SIGMOD 2001, pp. 425-436, **2001**

[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**

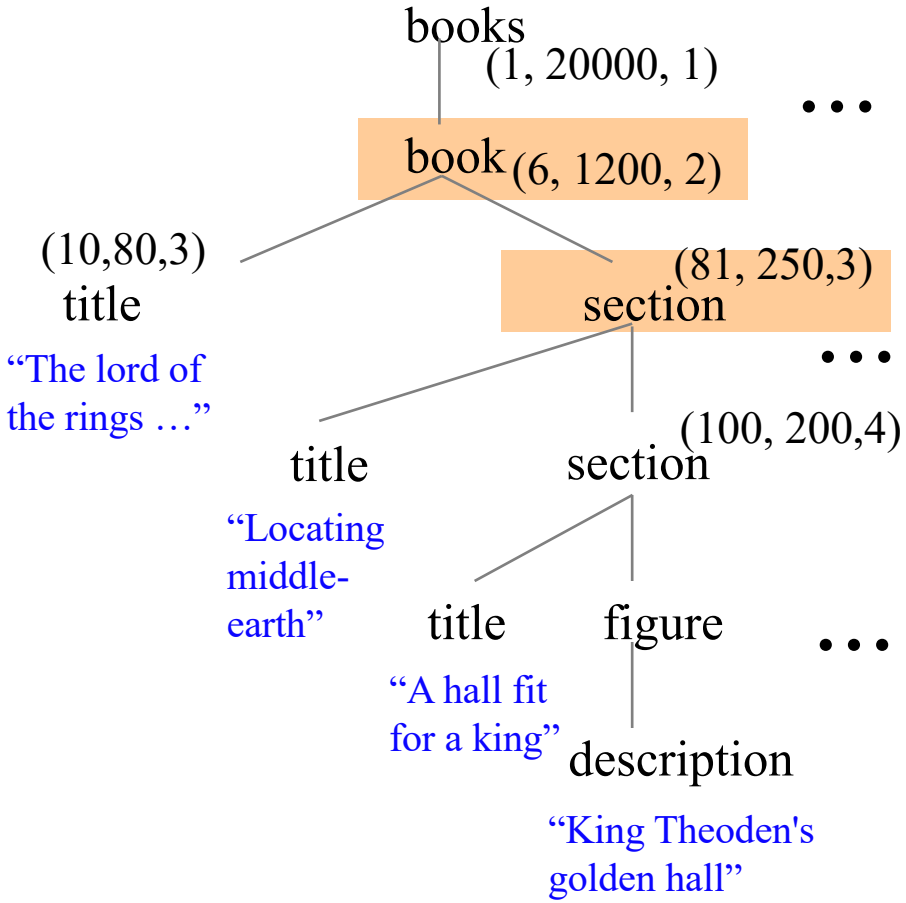
BLAS: An Efficient XPath Processing System

- **Bi**-**LA**beling based XPath processing **S**ystem
  - D-Beschriftung
    - <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**

D-Beschriftung: Dynamische Intervallkodierung



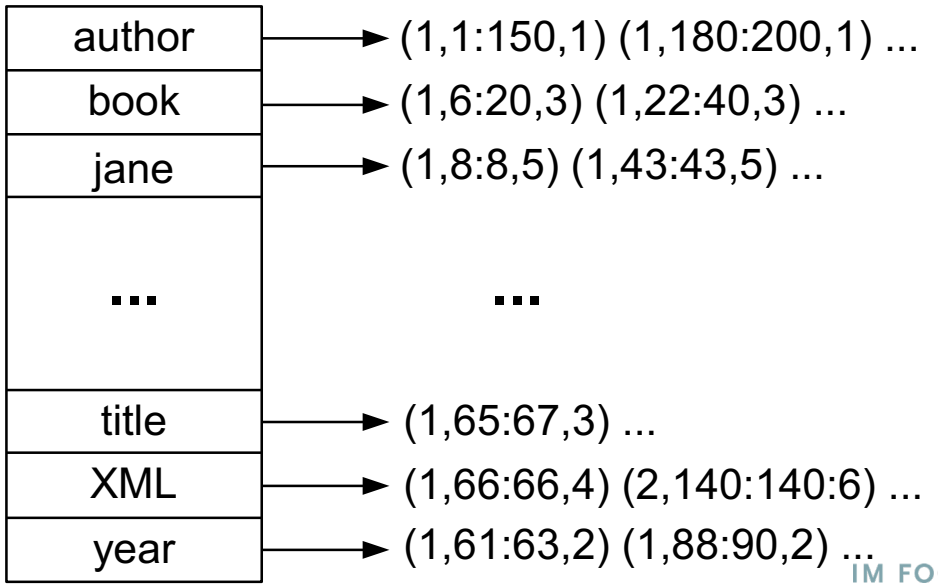
- Beschriftung <Start, Ende, Ebene> kann verwendet werden, um Vorgänger-Nachfolger-Beziehungen in einem Baum zu entdecken

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**

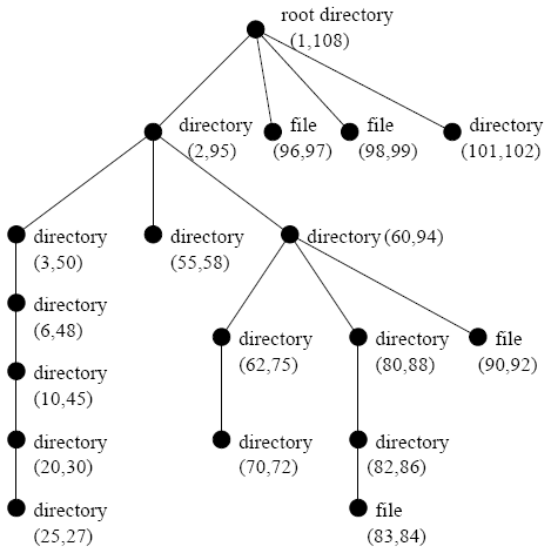
D-Beschriftungen

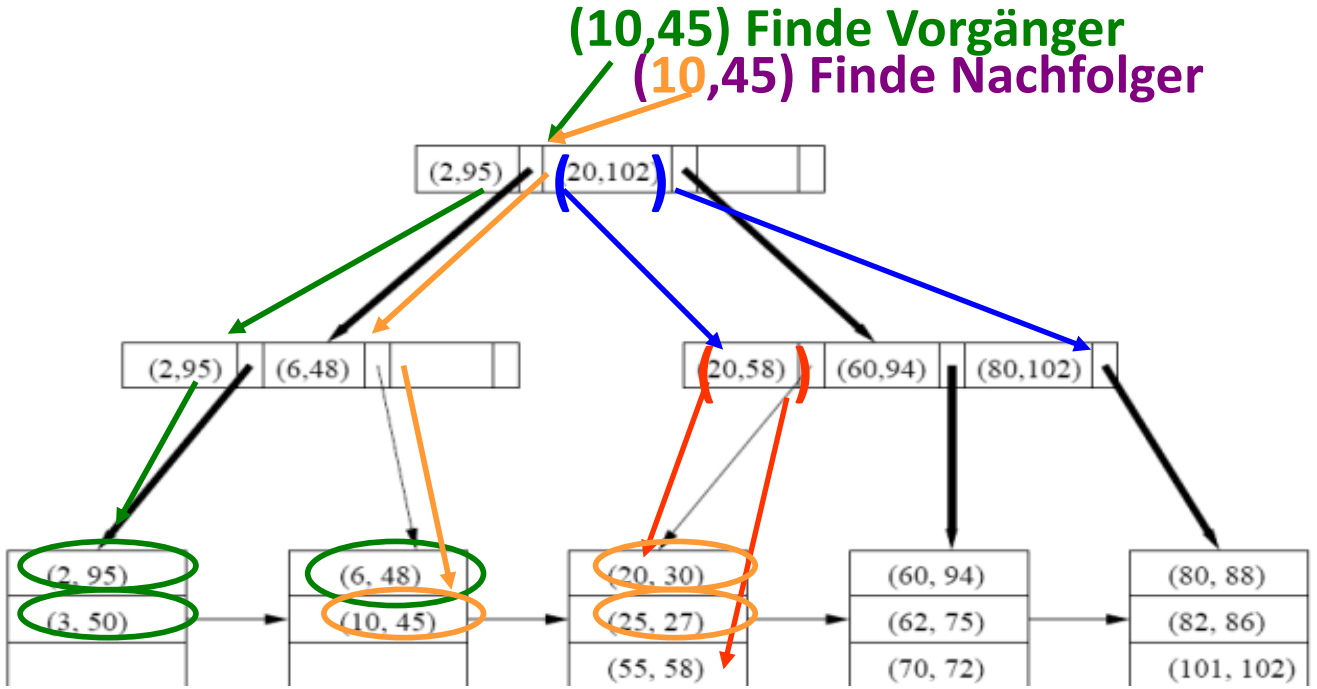
- Intervalle zur Nachfolgeranzeige: <Start, Ende, Ebene>
- Tripel <d_1,d_2,d_3> wird für jeden XML-Knoten n vergeben ($n.d_1 \leq n.d_2$)
  - m ist Nachfolger von n gdw. $n.d_1 < m.d_1$ und $m.d_2 < n.d_2$
  - m ist Kind von n gdw.
    m ist Nachfolger von n und $n.d_3+1 = m.d_3$
- Verwendung mit Tag-basierter SQL-Tabellen-repräsentation



XB-Tree: Nachfolger und Vorgängernavigation

- Verwendung von Indexen zur Effizienzsteigerung
- XB-Bäume sind wie 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**

Zugriff über B-Baum: Clustered Index

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

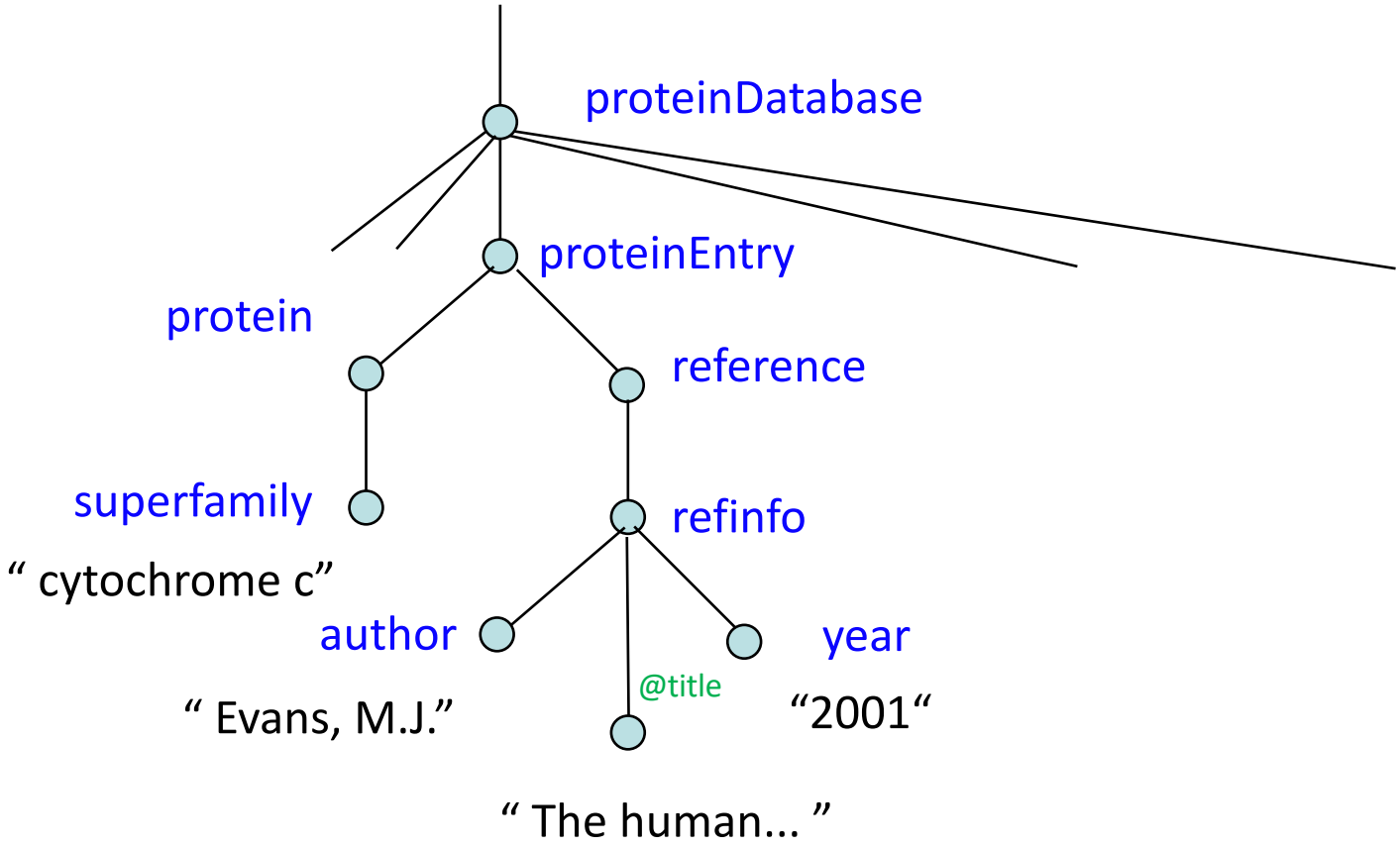
Ning Zhang, Varun Kacholia & M. Tamer Özsu. A Succinct Physical Storage Scheme for Efficient Evaluation of Path Queries in XML. In Proc. ICDE, pages 54–63, **2004**

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

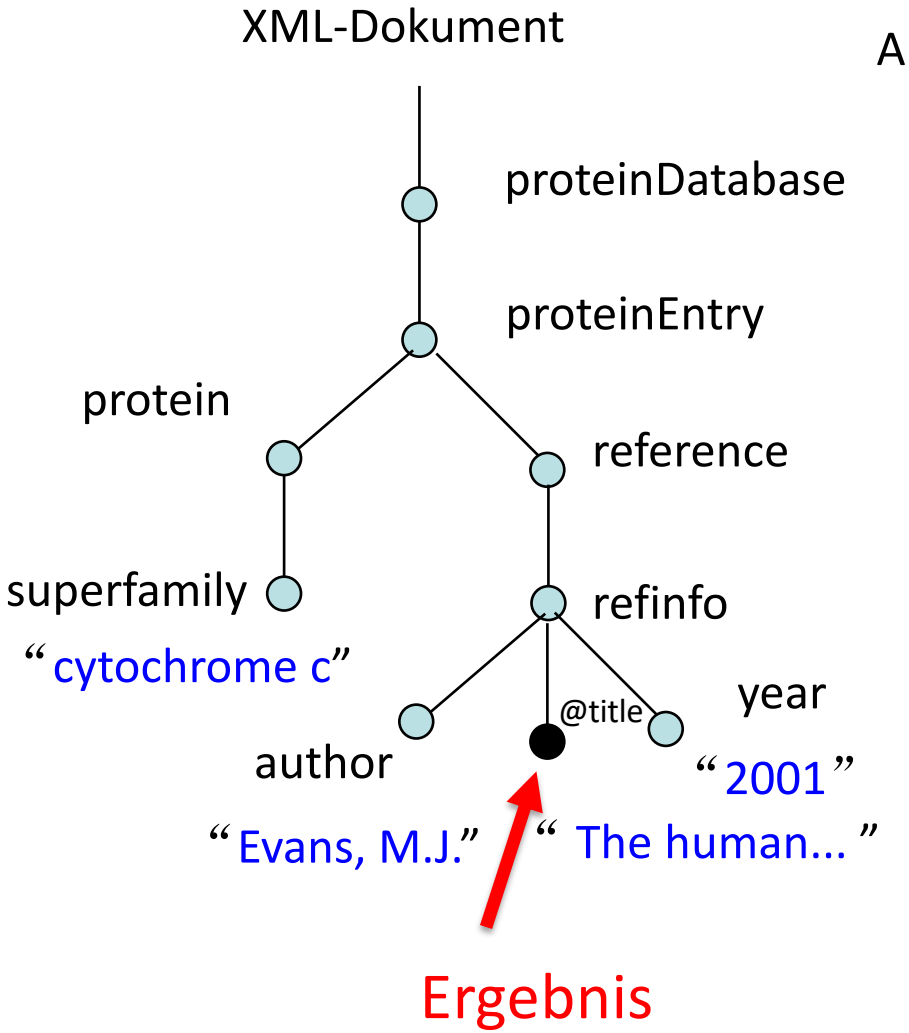
Ein Beispieldokument

```
<proteinDatabase>
   <proteinEntry>
      <protein>
         <Name> cytochrome c</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>
```

Graphische Darstellung eines kleinen Teils



Beispiel



Anfrage: //proteinDatabase//refinfo/@title

Suche alle Knoten proteinDatabase und refinfo

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

select refinfo.title

from proteinDatabase pDB, refinfo

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

P-Beschriftungen

- Kind-Navigation sollte ebenfalls effizient implementiert werden
  - Beispiel /proteinDatabase/proteinEntry/protein/name
- Bisher: Extrem viele Joins bei naiver SQL-Codierung
- Aufgabe: Verbesserung der "/"-Auswertung für Pfade
- Fokus auf Suffix-Pfadanfragen:
  - Beispiel: //protein/name

Definitionen / Notation

- Die Auswertung eines Pfadausdrucks P (geschrieben [P]) gibt eine Sequenz (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. $[P] \cap [Q] = \emptyset$ (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 im XML Baum(SP = source path)
- Die Auswertung eines Suffix-Pfadausdrucks Q ist die Bestimmung aller Knoten n, so dass SP(n) $\sqsubseteq$ Q

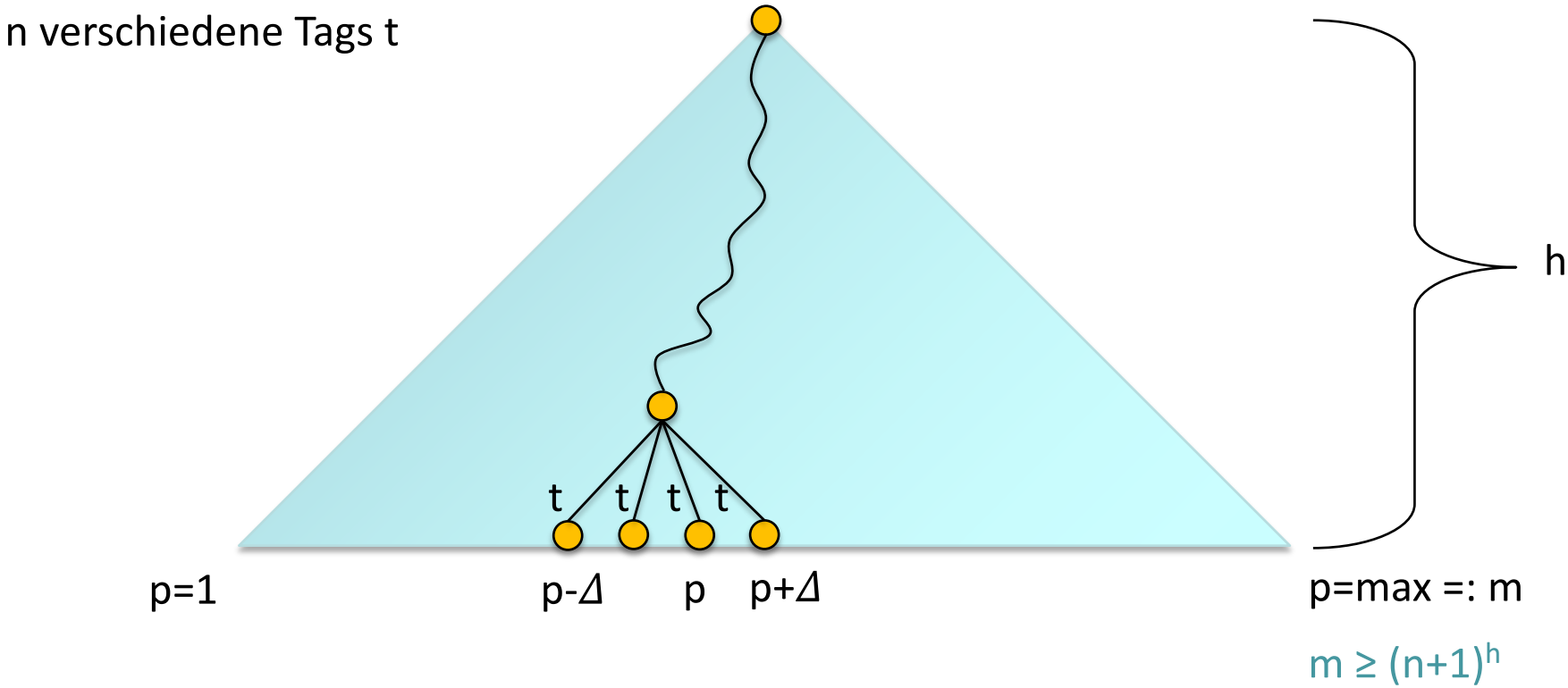
P-Beschriftungen

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

P-Beschriftungen



P-Beschriftung Beispiel

- Annahme: Längster Pfad: h=6
- Sei die Maxanzahl n der Auszeichner (Tags) auf 99 festgelegt
- Wähle maximalen p-Wert $m = 100^6 = 10^{12}$
  (wir wollen, dass $m \geq (n+1)^h$)
- Jedem Auszeichner (Tag) wird ein Bereich r zugewiesen:
  $r_i = 0.01 = 1/(99+1)$
- P-Beschriftung für jeden Suffix-Pfad bestimmen
- Beispiel P = /ProteinDatabase/ProteinEntry/protein/name

Ordnung der Tags für Partitionierung wie in P)

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	$< 4.030201 \times 10^{10}, 4.03020101 \times 10^{10} - 1 >$

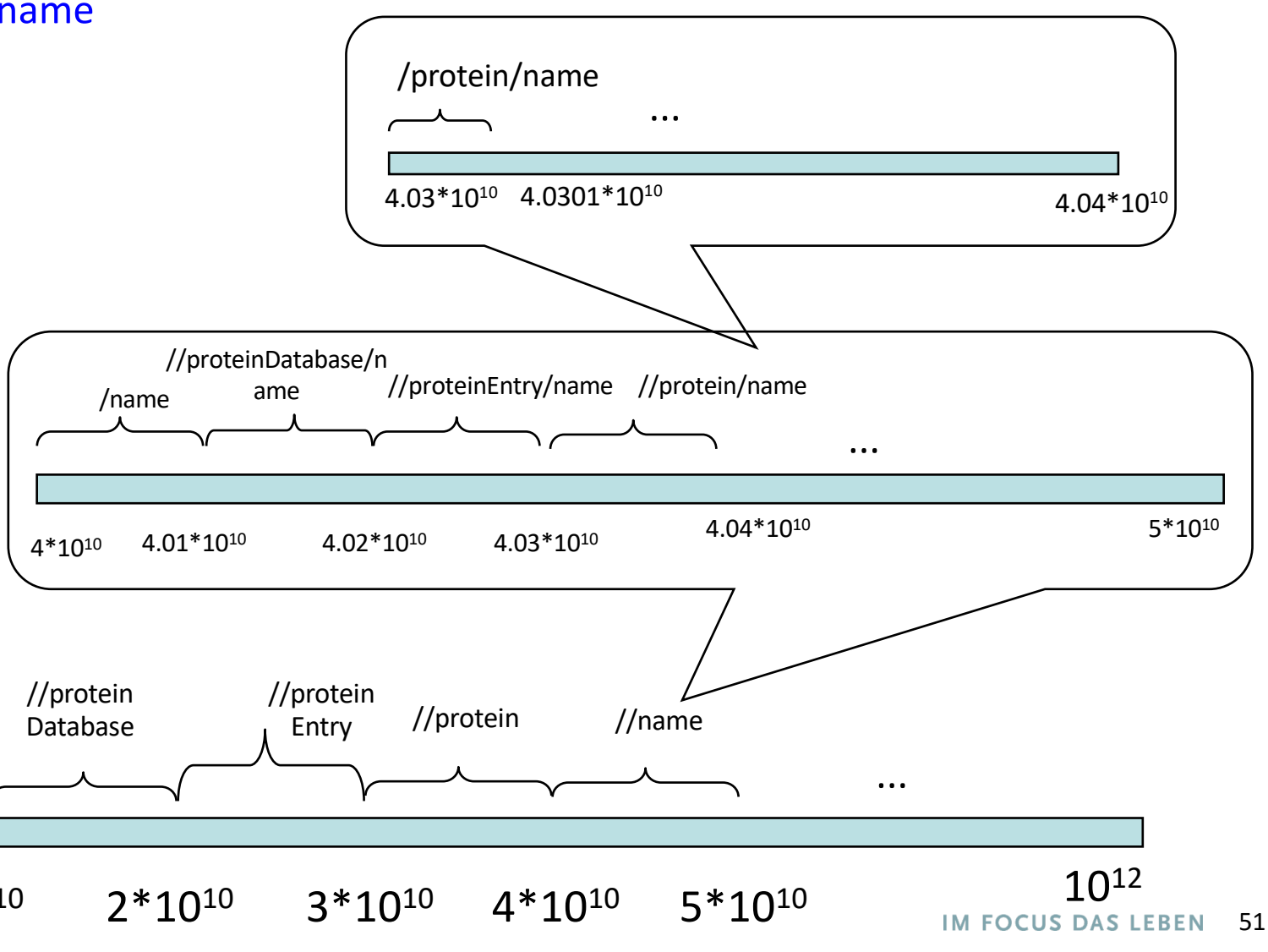
P-Beschriftungen (Beispiel)

Anfrage: //protein/name

$m=10^{12}$

99 Auszeichner

$r_i = 0.01$

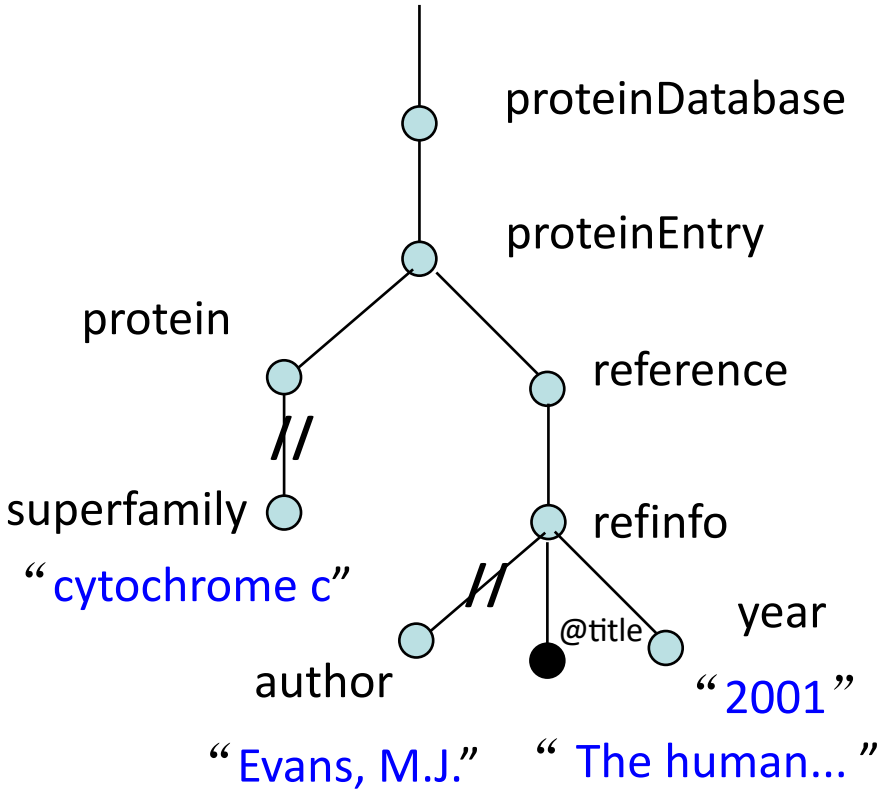


Zum Nachvollziehen zuhause:

- Seien n Auszeichner gegeben $(t_1,t_2,....,t_n)$.
- Weise "/" einen Wert r_0 und jedem Auszeichner t_i einen Wert r_i zu, so dass $r_0+r_1+r_2+.....+r_n = 1$.
- 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 "/" und $<p_i, p_{i+1}>$ für jedes t_i, so dass $(p_{i+1} - p_i)/m=r_i$ und $p_1/m = r_0$

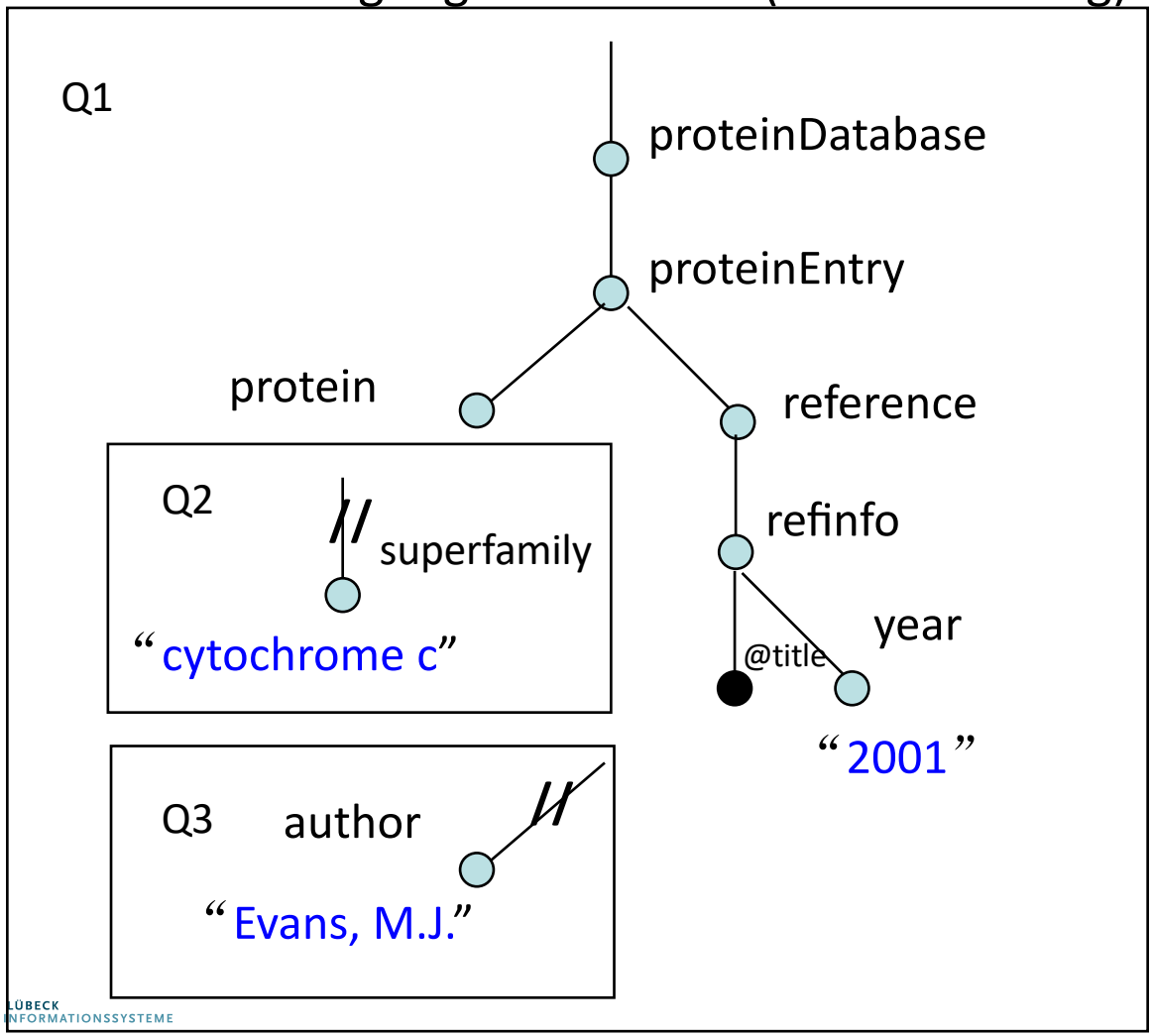
Zweiganfragen (TWIG-Patterns)

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



Ein Beispiel

- Aufspaltung (D-Eliminierung):
  - Rekursiver Selbstaufruf; Unterprozedur Verzweigungselimination (B-Eliminierung)



p//q → p and //q

Tiefensuche

Spalte p//q in p and //q

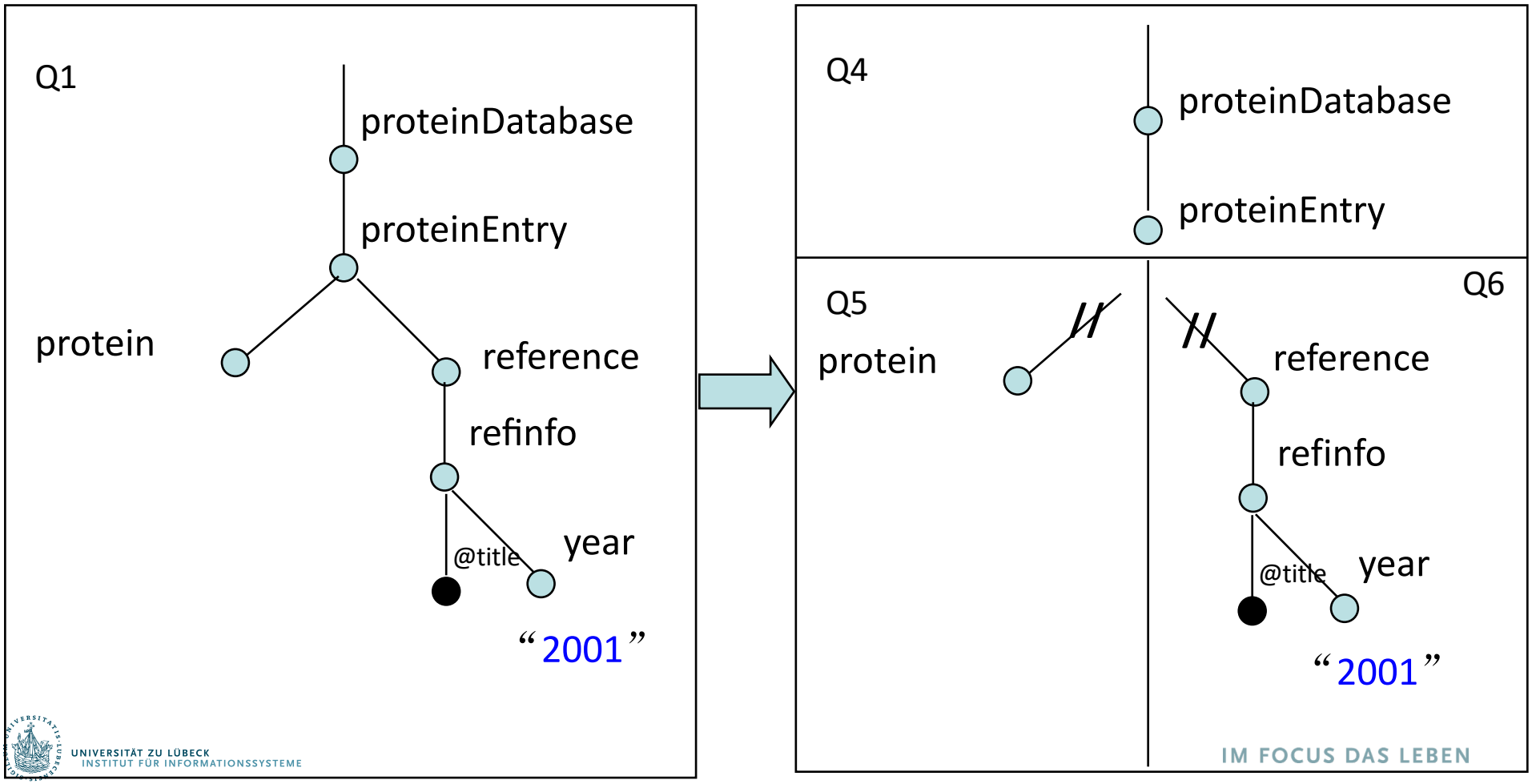
Eliminiere Verzweigung falls vorhanden, sonst evaluiere Q mit P-Beschriftungen

Join für Zwischenresultate unter Verwendung der D-Beschriftungen

Ein Beispiel

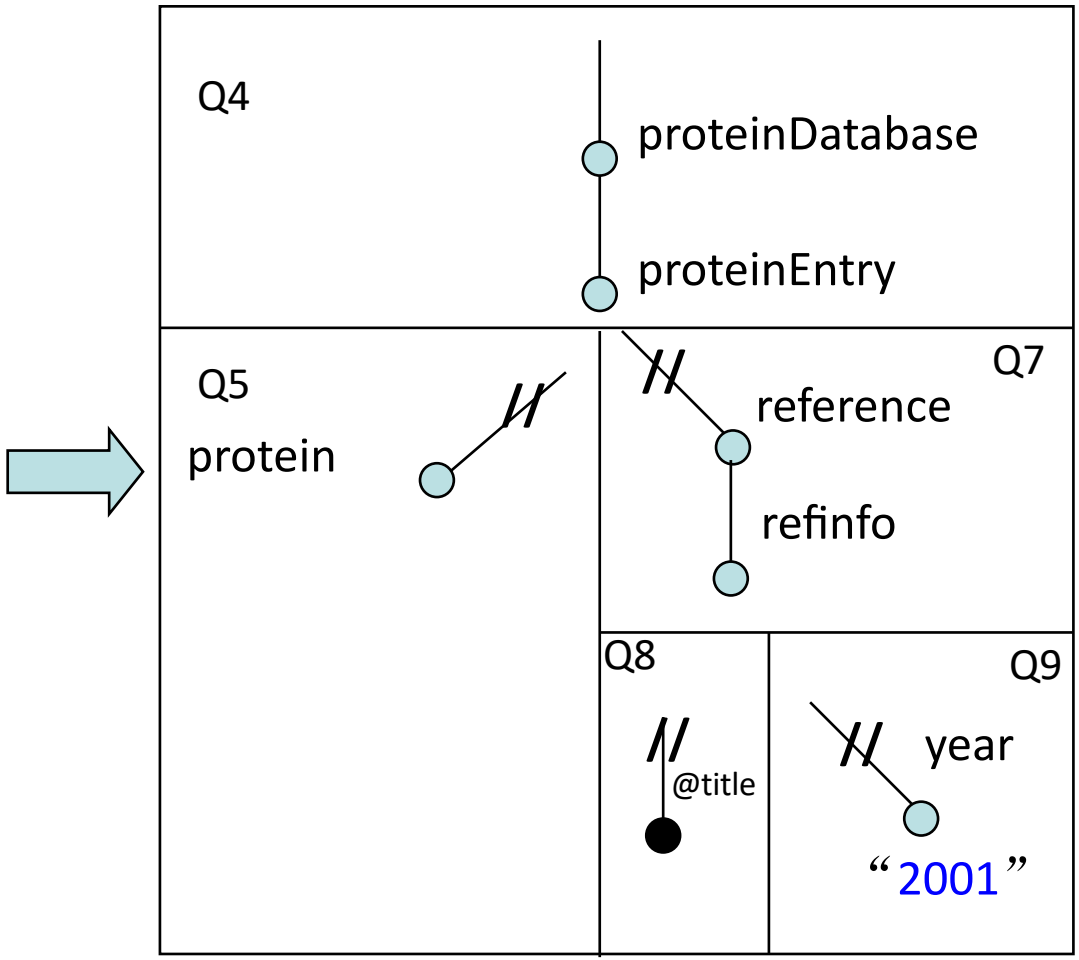
- Verzweigungselimination (B-Eliminierung)

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

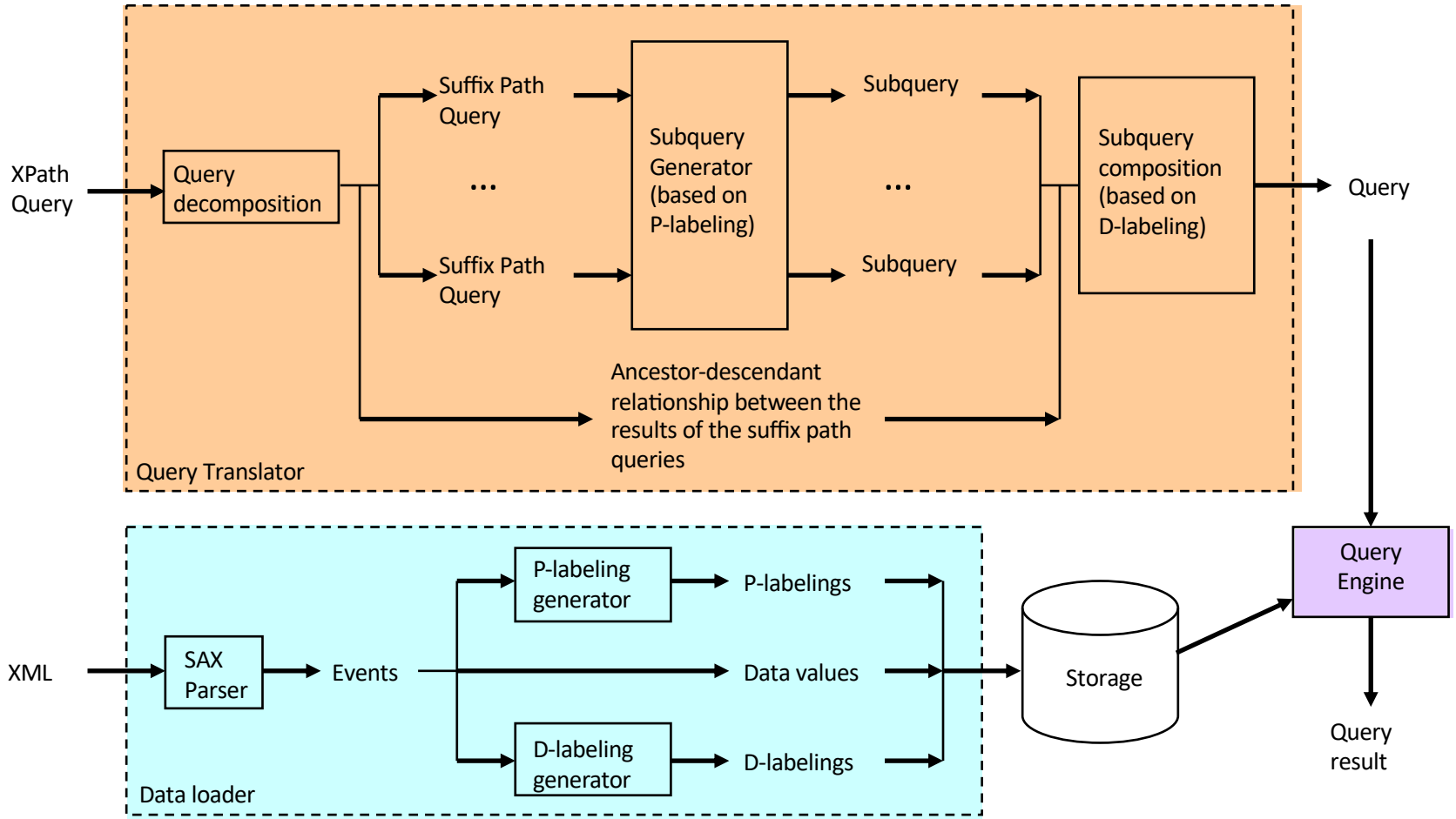


Ein Beispiel

Verzweigungselimination



BLAS Architektur



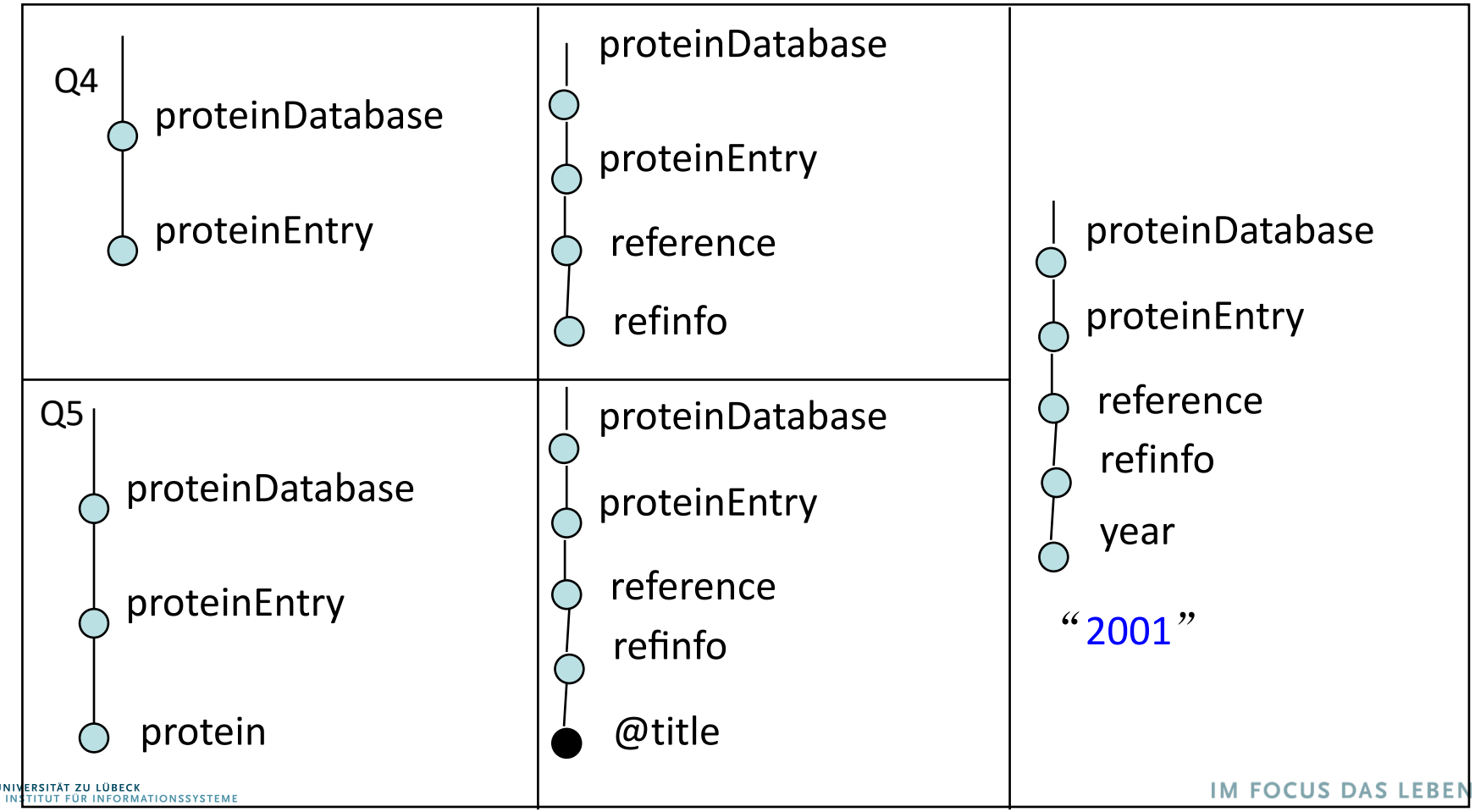
Aufspaltung von Anfragen

- Übersetzung einer 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

BLAS: Verfeinerung 1: Vermeidung von Joins

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



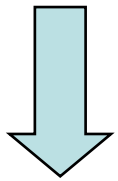
BLAS: Verfeinerung 2: Vermeidung von Joins

Entfaltungsalgorithmus als Nachfolger-Eliminierung (Enfaltungs-D-Eliminierung)

$p//q \rightarrow p/r_1/q,\ p/r_2/q, \ldots\ldots, p/r_i/q$

Ein Beispiel:

***Q*2=/ProteinDatabase/ProteinEntry/protein//superfamily="cytochrome c"**



***Q*21 = /ProteinDatabase/ProteinEntry/protein/classification/superfamily="cytochrome c"**

,

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

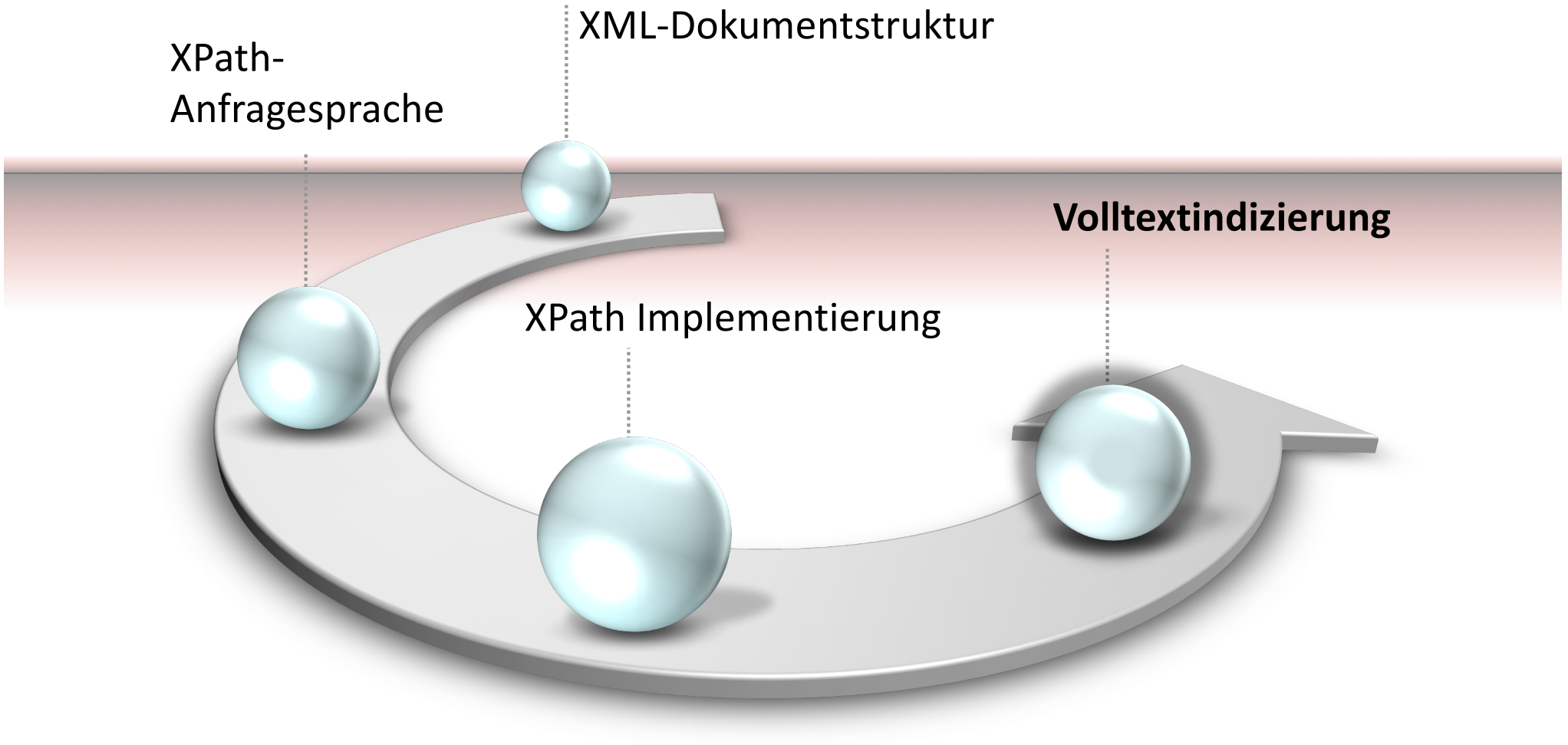
  Jinghua Groppe, Speeding up XML Quering: Satisfiability Test & Containment Test of XPath Queries in the Presence of XML Schema Definitions, Dissertation, **IFIS Universität Lübeck**, dissertation.de: Verlag im Internet GmbH, **2008**

- 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)

Non-Standard-Datenbanken

Semistrukturierte Datenbanken



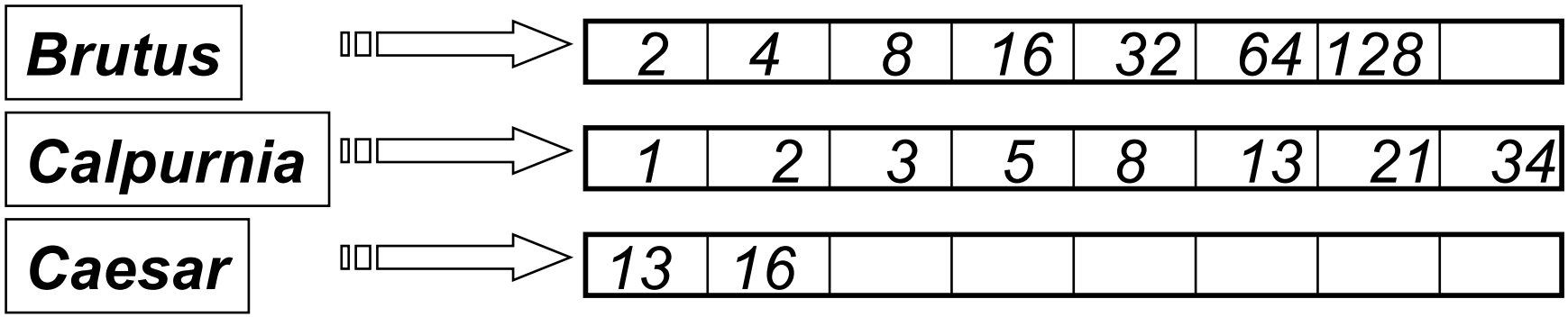
Text als unstrukturierte Daten

Welche Stücke //drama[author='shakespeare'] enthalten die Worte **Brutus** UND **Caesar** aber NICHT **Calpurnia**?

- //drama[author='shakespeare' and matches(content,'Brutus') & matches(content 'Caesar') and not(matches(content, 'Calpurnia'))]
- Verwendung z.B. von **Knuth-Morris-Pratt-Algorithmus,** um alle Stücke von Shakespeare mit **Brutus** und **Caesar** zu finden, um dann solche mit **Calpurnia** zu entfernen?
  - Langsam (für große Korpora)
  - NICHT **Calpurnia** ist keinesfalls einfacher zu behandeln
  - Andere Operationen (z.B. Finde das Wort **romans** in der Nähe von **countrymen**) sind nicht einfach umsetzbar
  - Ggf. Bewertung von Ergebnissen gewünscht

Invertierter Index

- Für jeden Term T, müssen wir eine Liste aller Dokumente, die T enthalten, speichern
- Sollen wir ein Feld oder eine Liste verwenden?

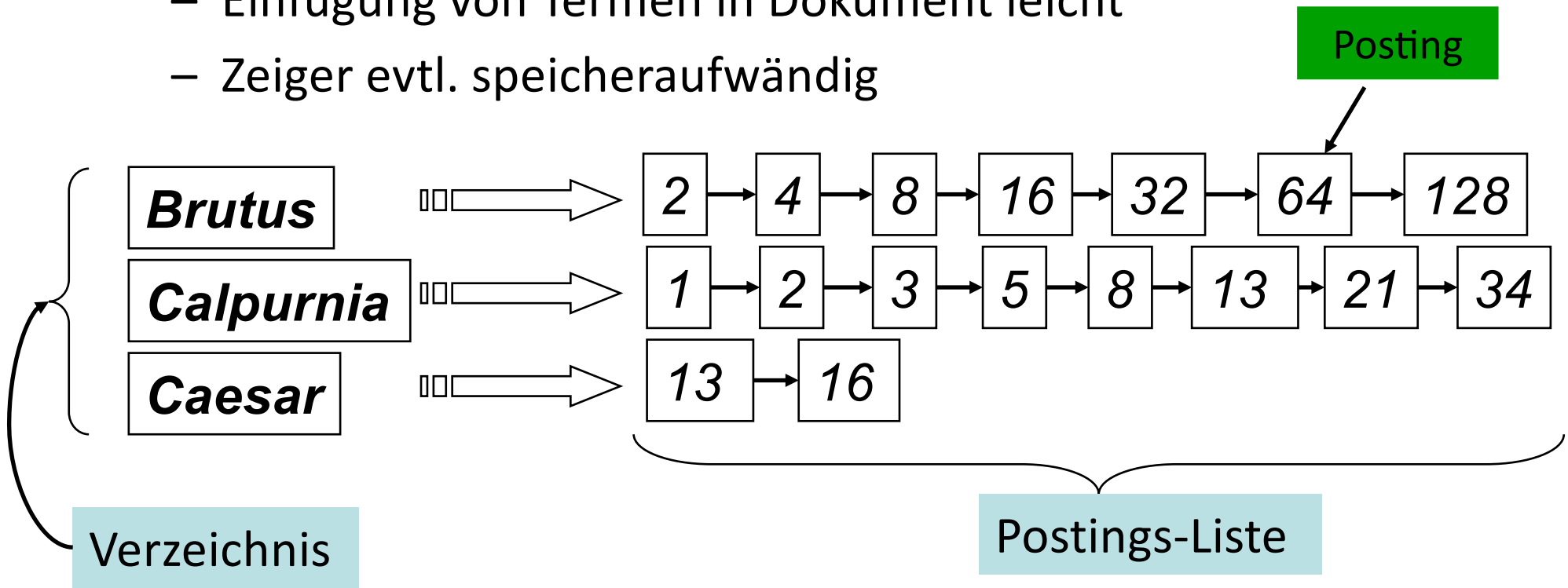


*Was machen wir, wenn das Wort **Caesar** zu Dokument 14 hinzugefügt wird?*

Luhn, H. P., Auto-encoding of documents for information retrieval systems. In M. Boaz, *Modern Trends in Documentation* (pp. 45-58). London: Pergamon Press, **1959**

Invertierter Index

- Verkettete Listen i.a. bevorzugt
  - Dynamische Speicherallokation
  - Einfügung von Termen in Dokument leicht
  - Zeiger evtl. speicheraufwändig



Sortiert nach docID (später mehr dazu)

Antworten im Kontext

Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,
When Antony found Julius **Caesar** dead,
He cried almost to roaring; and he wept
When at Philippi he found **Brutus** slain.

Lineare Suche des Teilausschnitts könnte schon zu langsam sein
→ Positionen auch abspeichern

Hamlet, Act III, Scene ii

Lord Polonius: I did enact Julius **Caesar** I was killed i' the
Capitol; **Brutus** killed me.

Non-Standard-Datenbanken und Data Mining

Semistrukturierte Datenbanken und Volltextindizierung

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Indexierungsschritte

- Geg.: Sequenz von Paaren (Token, DocID).

Doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Indizierungsschritte

Sortierung nach Term

Hoher Aufwand

Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Indizierungsschritte

- Mehrfacheinträge für Terme aus einem Dokument werden verschmolzen
- Anzahlinformation wird hinzugefügt

Warum Anzahl?
Bewertung von Antworten.

Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Term	Doc #	Term freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

Umsetzung in SQL?

- SQL:

TermDoc

Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Term	Doc #	Term freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

Das Ergebnis wird in eine Verzeichnis- und eine Postings-Tabelle unterteilt

Term	Doc #	Freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

Term	N docs	Coll freq	Zeiger
ambitious	1	1	
be	1	1	
brutus	2	2	
capitol	1	1	
caesar	2	3	
did	1	1	
enact	1	1	
hath	1	1	
I	1	2	
i'	1	1	
it	1	1	
julius	1	1	
killed	1	2	
let	1	1	
me	1	1	
noble	1	1	
so	1	1	
the	2	2	
told	1	1	
you	1	1	
was	2	2	
with	1	1	

Doc #	Freq
2	1
2	1
1	1
2	1
1	1
1	1
2	2
1	1
1	1
2	1
1	2
1	1
2	1
1	1
1	2
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1

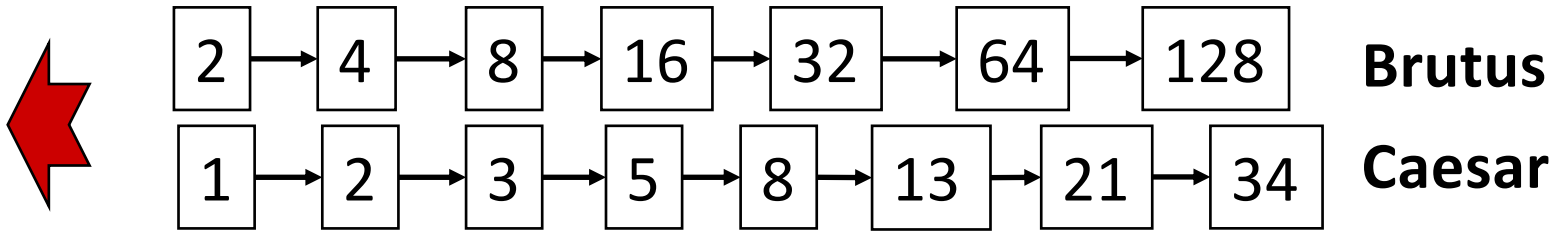
Warum N docs und Coll freq?
Bewertung von Antworten.

Anfrageverarbeitung: UND

- Betrachten wir folgende Anfrage:

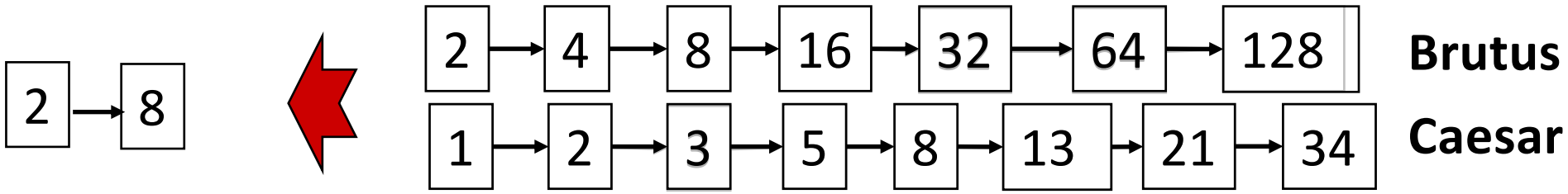
  Brutus UND **Caesar**

  – Suche **Brutus** im Verzeichnis
    - Hole die zugeordnete Postings-Liste
  – Suche **Caesar** im Verzeichnis
    - Hole die Postings-Liste
  – “Verschmelze” die Listen



Die Verschmelzung

- Gehe mit zwei Zeigern durch die Postings-Listen



Falls die Listen die Länge x und y haben, dauert die Verschmelzung O(x+y) Schritte.

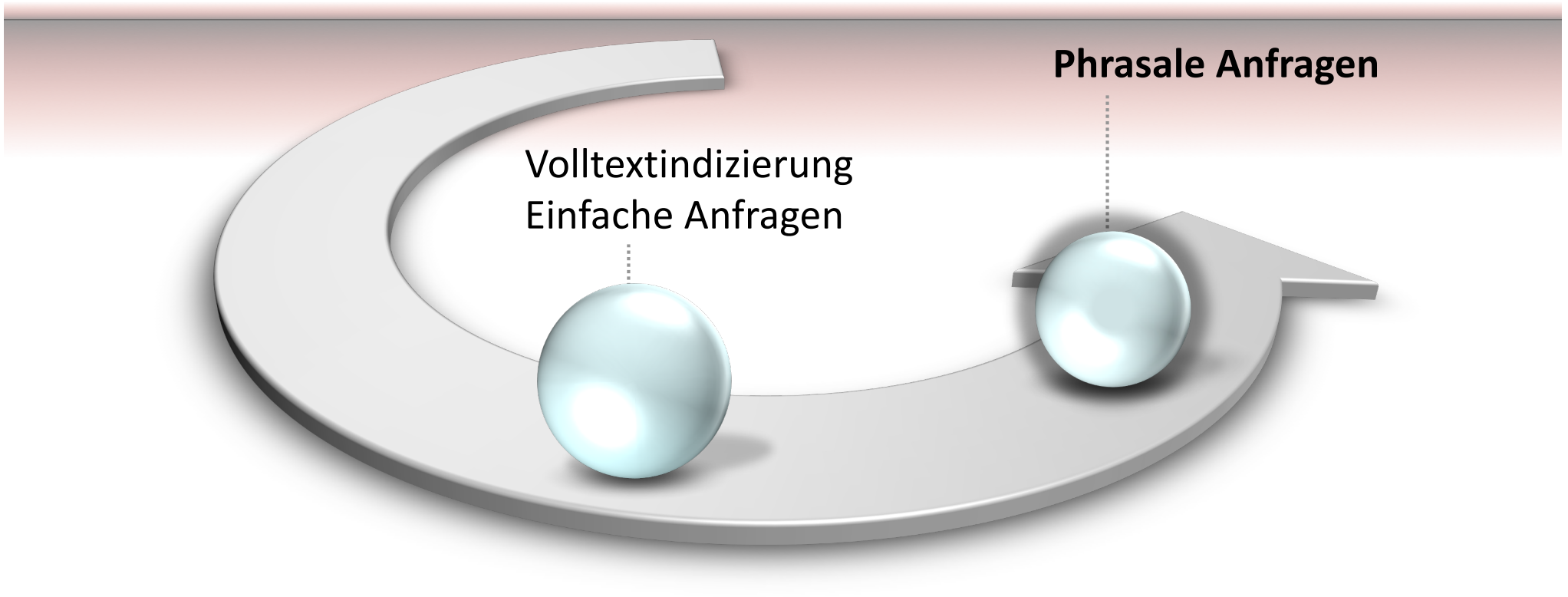
Notwendig: Postings nach docID sortiert

Knuth, D. E. Kap. 6.5. “Retrieval on Secondary Keys". The Art of Computer Programming, Reading, Massachusetts: Addison-Wesley, **1973**

Andere Methode: Signature Files z.B. mit Bloom Filter (später vertieft)
Bloom, Burton H., Space/Time Trade-offs in Hash Coding with Allowable Errors, Communications of the ACM 13 (7), 422–426, **1970**

Non-Standard-Datenbanken

Von semistrukturierten Datenbanken zur Volltextsuche



Phrasale Anfragen

- Gesucht sind Antworten auf
  "**stanford university"** – als eine Phrase
- Der Satz "I went to university at Stanford" stellt keinen Treffer dar
  - Das Konzept der Phrasenanfrage wird von Nutzern gut verstanden und einigermaßen häufig verwendet
- Es reicht nicht, nur <Term : docID>-Einträge zu speichern

Erster Versuch: Zweiwort-Indexe

- Indexiere jedes aufeinanderfolgende Paar von Termen im Text als Phrase
- Beispieltext: “Friends, Romans, Countrymen”
- Zweiworte:
  - **friends romans**
  - **romans countrymen**
- Jedes dieser Zweiworte wird nun ein Eintrag im Verzeichnis
- Zweiwort Phrasenanfragen können nun einfach behandelt werden

Längere Phrasenanfragen

- Beispiel:

  stanford university palo alto

- Behandlung als boolesche Kombination von Zweiwortanfragen:

  stanford university UND **university palo**
  UND **palo alto**

Falsch-positive Lösungen möglich!

- Nachbetrachtung der gefundenen Dokumente notwendig

Denkaufgabe:

Können wir den nachträglichen Dokumentenabgleich zur Vermeidung von falsch-positiven Ergebnissen vermeiden?

Positionsbezogene Indexe

- Speichere für jeden Term folgende Einträge: <Anzahl der Dokumente, die Term enthalten;

  Doc_1: $Position_1$, $Position_2$ … ;

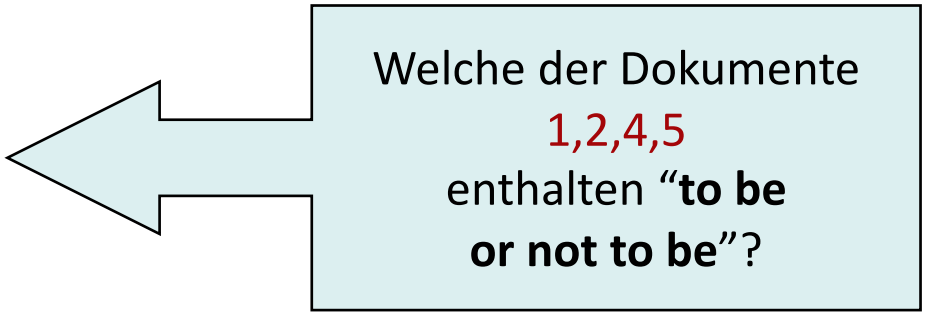
  Doc_2: $Position_1$, $Position_2$ … ;

  …>

- $Position_i$ kann Offset sein oder Wortnummer

Beispiel: Positionsbezogener Index

<**be**: 993427;
1: 7, 18, 33, 72, 86, 231;
2: 3, 149;
4: 17, 191, 291, 430, 434;
5: 363, 367, …>



- Datenkomprimierung möglich
- Allerdings erhöht sich der Speicherverbrauch substantiell

Verarbeitung einer Phrasenanfrage

- Extrahiere die Einträge im invertierten Index für **to, be, or, not.**
- Verschmelze die Dok-Positions-Listen um alle Positionen zu finden mit
  "**to be or not to be**".
  - **to**:
    - 2:1,17,74,222,551; 4:8,16,190,429,433; 7:13,23,191; ...
  - **be**:
    - 1:17,19; 4:17,191,291,430,434; 5:14,19,101; ...
- Auch anwendbar für Suchen mit "In der Nähe von"-Operatoren

Denkaufgabe:

Finde alle Dokumente, die ein Wort enthalten, das mit "mon" beginnt.

Wie können wir das realisieren?

Platzhalter-Anfragen: *

- **mon*:** Finde alle Dokumente, die ein Wort enthalten, das mit “mon” beginnt
- Verwendung eines B-Baums mit Eintragungen in lexikographischer Ordnung
- Finde alle Worte **w**, so dass **mon ≤ w < moo**

Denkaufgabe:

Finde alle Dokumente, die ein Wort enthalten, das mit "mon" endet.

Wie können wir das realisieren?

Platzhalter-Anfragen

- ***mon:** Finde Worte, die mit mon enden
- Mögliche Lösung:
  - Erstelle B-Baum für Terme rückwärts geschrieben
  - Realisierung der Anfrage als Bereichsanfrage: **nom ≤ w < non**.

Denkaufgabe:

Wie können wir alle Terme (und damit Dokumente) finden, die auf pro*cent passen?

Was machen wir bei se*ate AND fil*er?

*’ne ans Ende rotieren

- Was machen wir bei multiplen Platzhaltern?
- Neuer Ansatz:

  Transformiere Anfrage, so dass Platzhalter am Ende der Anfrage auftreten

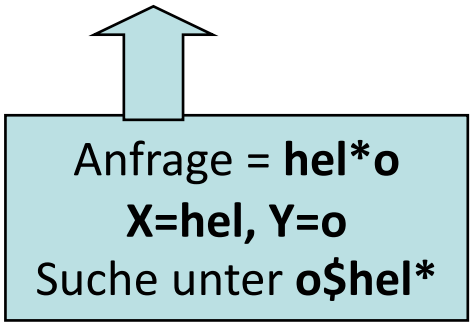
  → “Permuterm”-Index

Permuterm-Index

- Für Term **hello** erstelle Indexeinträge:
  - **hello$, ello$h, llohe, lohel, o$hell, $hello**

  Wobei $ ein spezielles Symbol ist

- Behandlung von Anfragen:
  - **X** suche unter **X$**
  - ***X** suche unter **X$***
  - **X*Y** suche unter **Y$X***
  - **X*** suche unter **$X***
  - ***X*** suche unter **X***



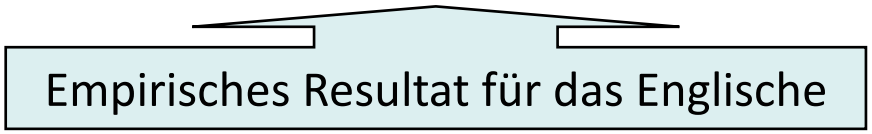
Herbert Marvin Ohlman **1957**
Hans Peter Luhn **1960**

Denkaufgabe:

Wie behandeln wir X*Y*Z?

Permuterm-Anfrageverarbeitung

- Rotiere Anfrageplatzhalter nach rechts
- Verwende B-Baum-Zugriff wie bekannt
- Permuterm-Problem: ≈ Vervierfachung der zu speichernden Daten im “Lexikon”



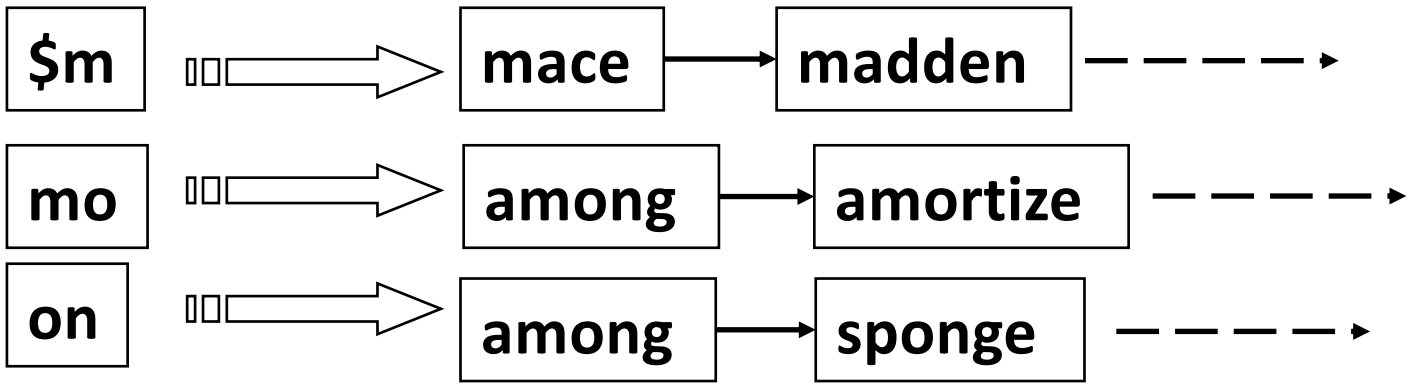
Bigramm-Indexe

- Bestimme alle n-Gramme (Sequenz von n Zeichen), die in den Termen vorkommen
- Beispieltext "**April is the cruelest month**"
  Wir erhalten folgende 2-Gramme (Bigramme)

  a,ap,pr,ri,il,l,i,is,s,t,th,he,e,$c,cr,ru,
  ue,el,le,es,st,t$, m,mo,on,nt,h

  – $ ist ein besonderes Abgrenzungssymbol
- Aufbau eines invertierten Index für Bigramme auf Verzeichnisterme, in denen das Bigramm vorkommt

Beispiel: Bigramm-Index



Verarbeitung von n-Gramm-Platzhaltern

- Anfrage **mon*** kann als
  - **$m** AND **mo** AND **on** formuliert werden
- Schnell und speichereffizient
- Hole Terme und gleiche die UND-Version der Platzhalteranfrage ab
- Leider kriegen wir z.B. auch **moon** zu fassen
- Nachverarbeitung notwendig
- Überlebende Terme führen dann über den Term-Dokument-Index zum Dokument (und ggf. zur Position darin zur Hervorhebung)

- Suche nach ungefähr passenden Texten? Plagiate?

BLAST: Basic Local Alignment Search Tool

- Annahme: Falls zwei Sequenzen eine gute lokale Übereinstimmung haben, dann ist mit großer Wahrscheinlichkeit eine kleine Teilzeichenkette sogar identisch (Seed)

```
TTGACTCGATTATAGTCGCGGATATACTATCG
CCTATCACAAGAATATAGTCCCTGATCCAGC

   TTGACTC GATTATAGTCGCGGAT ATACTATCG
CCTATCACAA GAATATAGTCCCTGAT CCAGC

   TTGACTC GATTATAGTCGCGGAT ATACTATCG
CCTATCACAA GAATATAGTCCCTGAT CCAGC
```

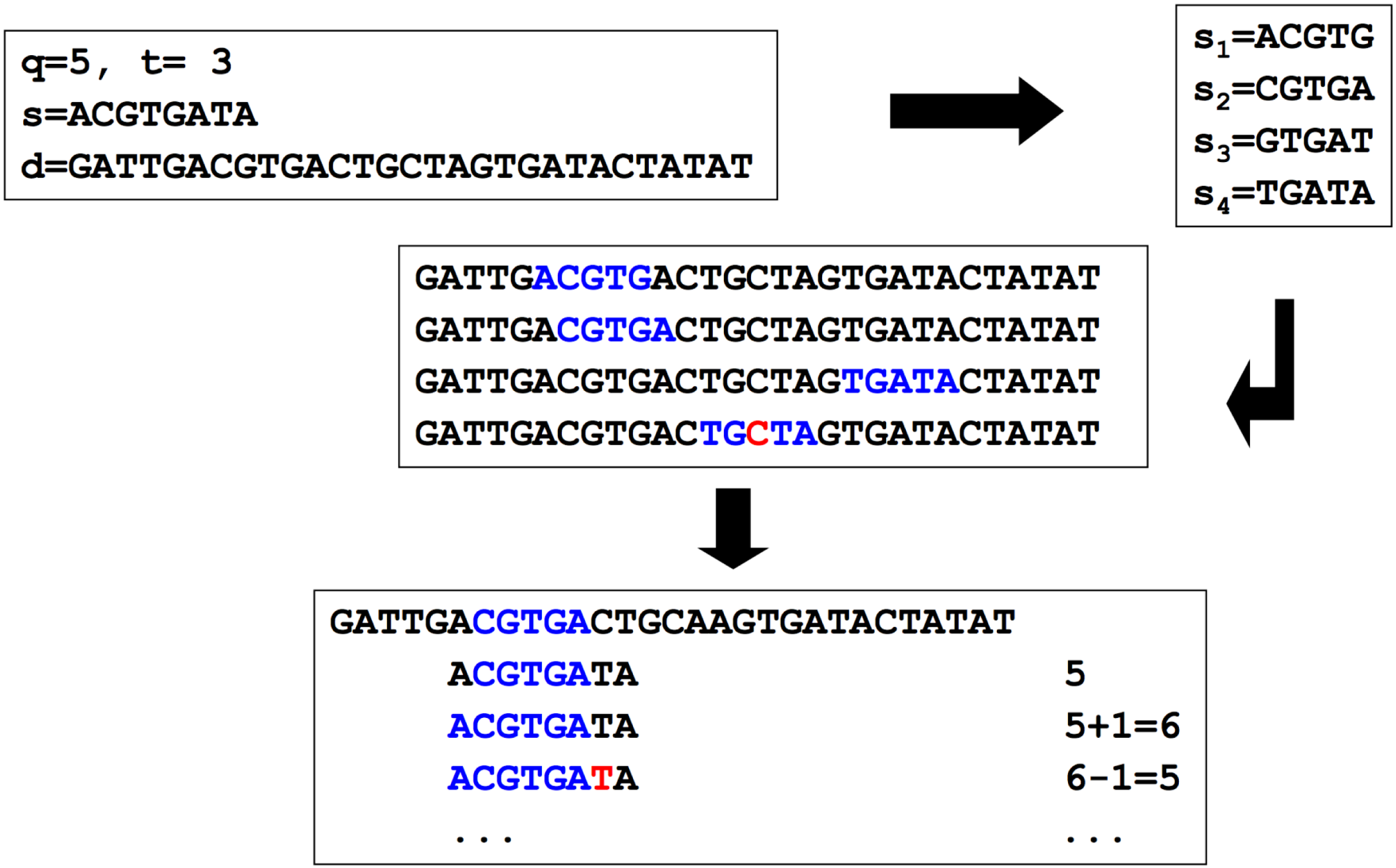

Altschul, Gish, Miller, Myers, Lipman: „Basic Local Alignment Search Tool“, J Mol Bio, **1990**

BLAST-Algorithmus

Gegeben sei eine Anfragesequenz s und eine Sequenzdatenbank $D=\{d_i\}$

1. Berechne Teilsequenzen s_i von s der Länge q
   - Auch Q-Gramme genannt (wieviele?)
   - Füge evtl. Q-Gramme mit Änderungen hinzu, um die Anzahl der Passungen zu erhöhen (mit geringerer Bewertung)
2. Finde alle approximativen Vorkommen aller s_i in allen d_j
   - Lückenfreie Ausrichtung (initiale Treffer) mit Bewertung
3. Erweitere Seeds nach links und rechts in si und dj
   - Dynamische Programmierung
   - Ergänzung des Bewertungsmaßes der initialen Treffer
   - Schwellwert t

Beispiel


q=5, t= 3
s=ACGTGATA
d=GATTGACGTGACTGCTAGTGATACTATAT
s1=ACGTG
s2=CGTGA
s3=GTGAT
s4=TGATA
GATTGACGTGACTGCTAGTGATACTATAT
GATTGACGTGACTGCTAGTGATACTATAT
GATTGACGTGACTGCTAGTGATACTATAT
GATTGACGTGACTGCTAGTGATACTATAT
GATTGACGTGACTGCAAGTGATACTATAT
ACGTGATA 5
ACGTGATA 5+1=6
ACGTGATA 6-1=5
... ...

Eigenschaften

- Finden von initialen Treffern aufwendig
  - Vorberechnung der Q-Gramme aller d_i
  - Aufbau des invertierten Index
  - Gegeben s, berechne alle Q-Gramme
    (evtl. mit Editieroperationen, falls initiale Treffermenge zu klein)

- Geeignet zur Tippfehlerkorrektur
  für Zugriff auf Volltextindex

Zusammenfassung