XPath-Aware Chunking of XML-Documents

Wolfgang Lehner

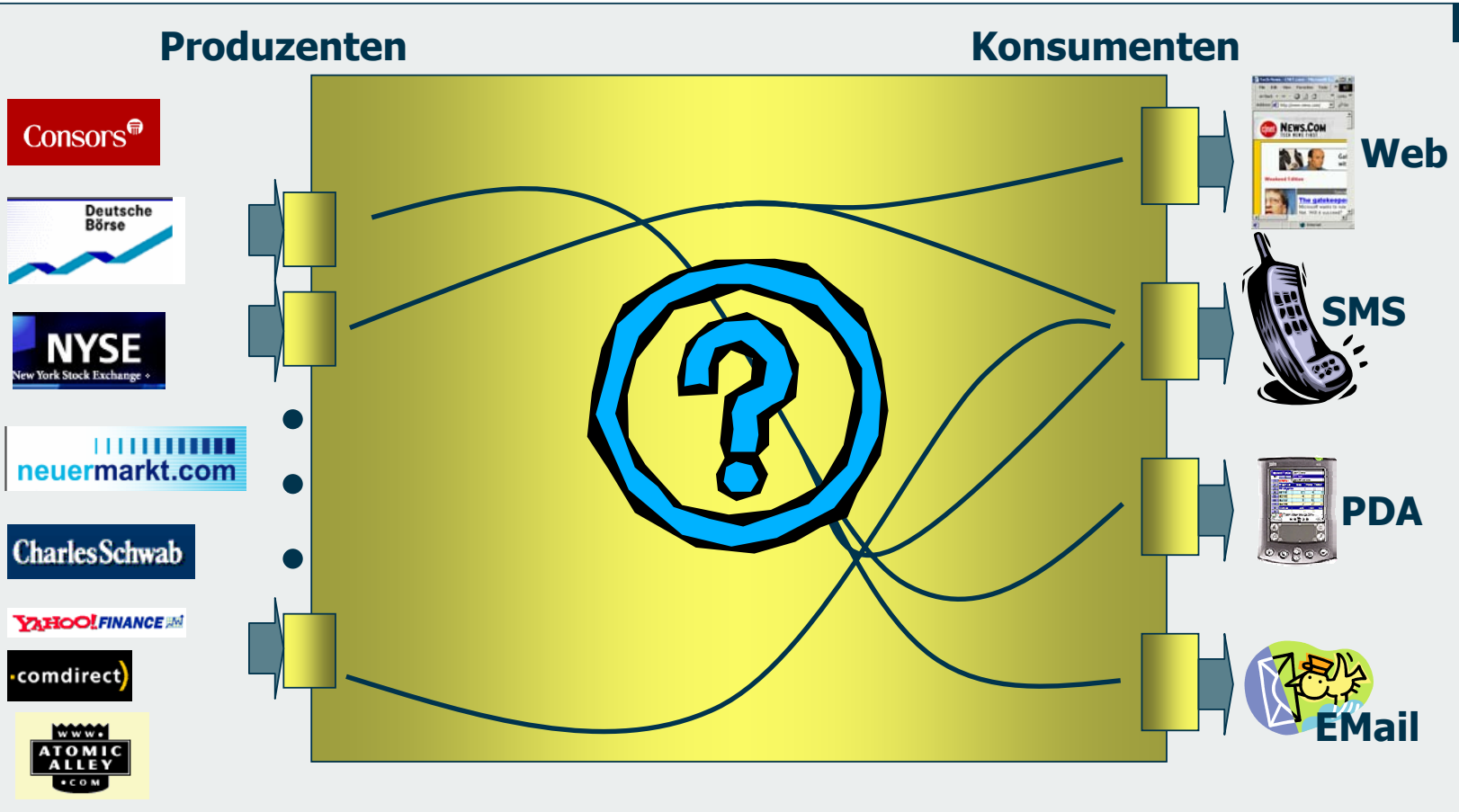
Technische Universität Dresden
Fakultät Informatik
Institut für Systemarchitektur (Datenbanken)

Florian Irmert

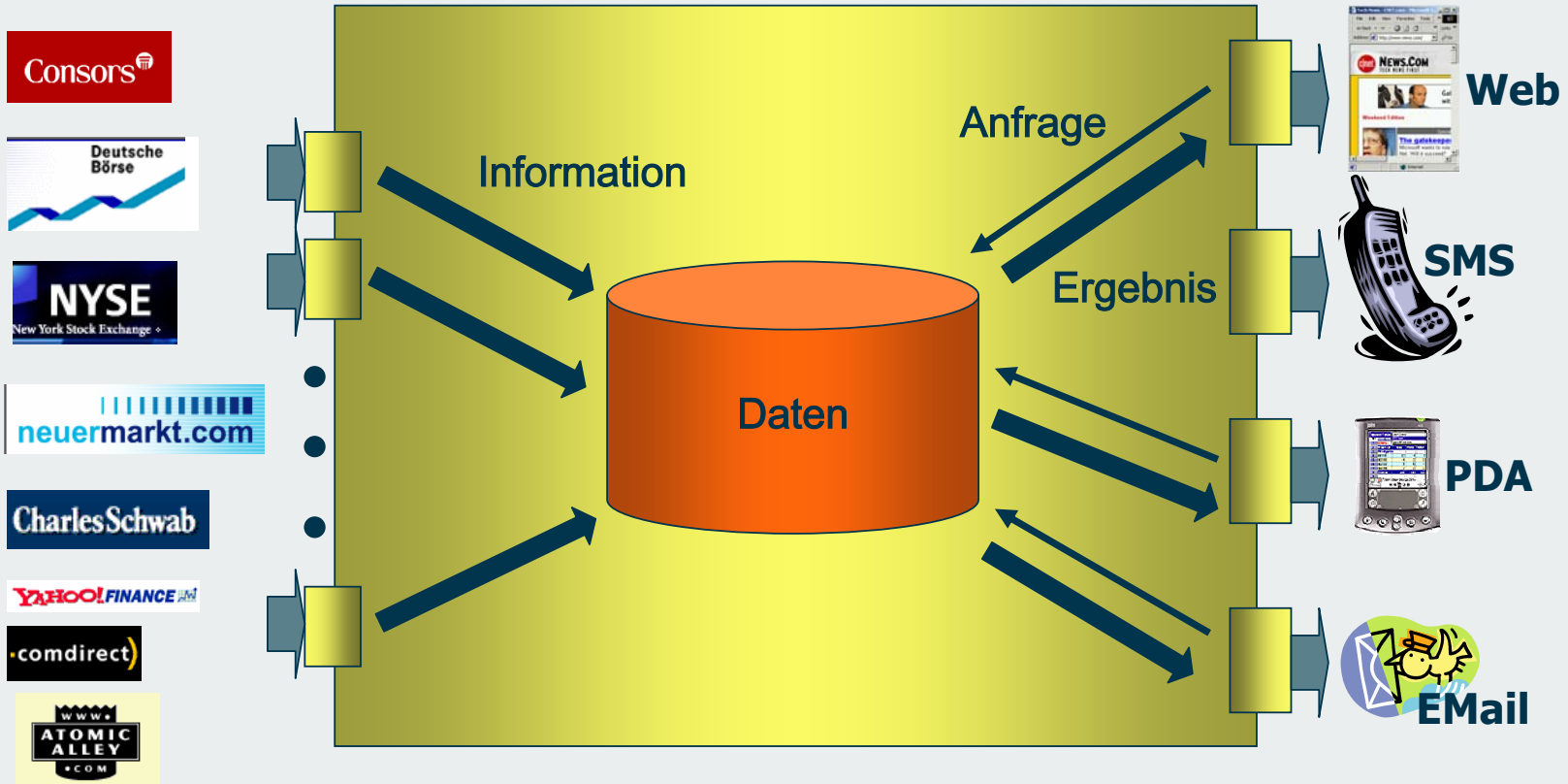
Universität Erlangen-Nürnberg
Institut für Informatik
Lehrstuhl für Datenbanksysteme

BTW 2003 - Leipzig

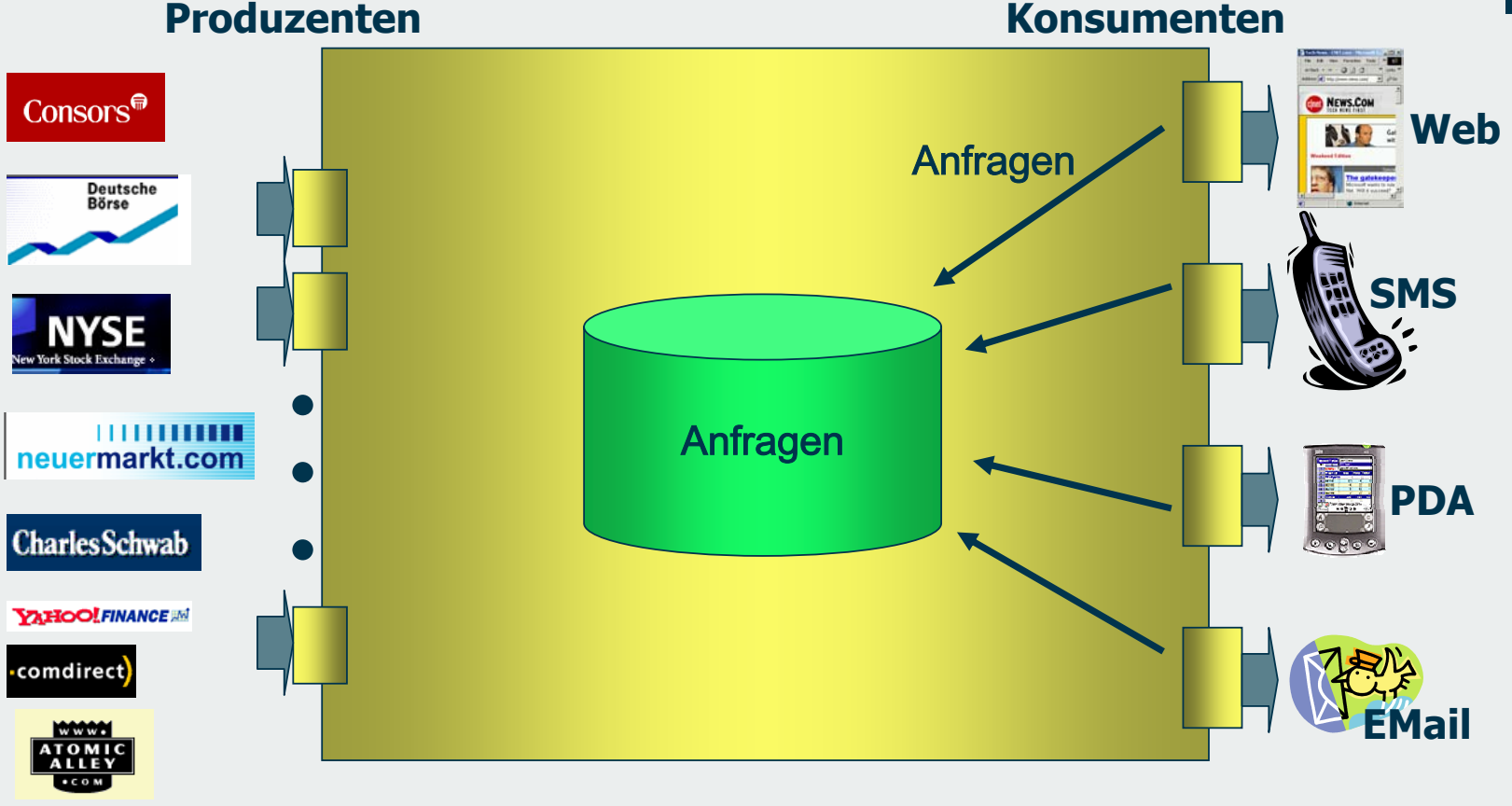
Beispielszenario



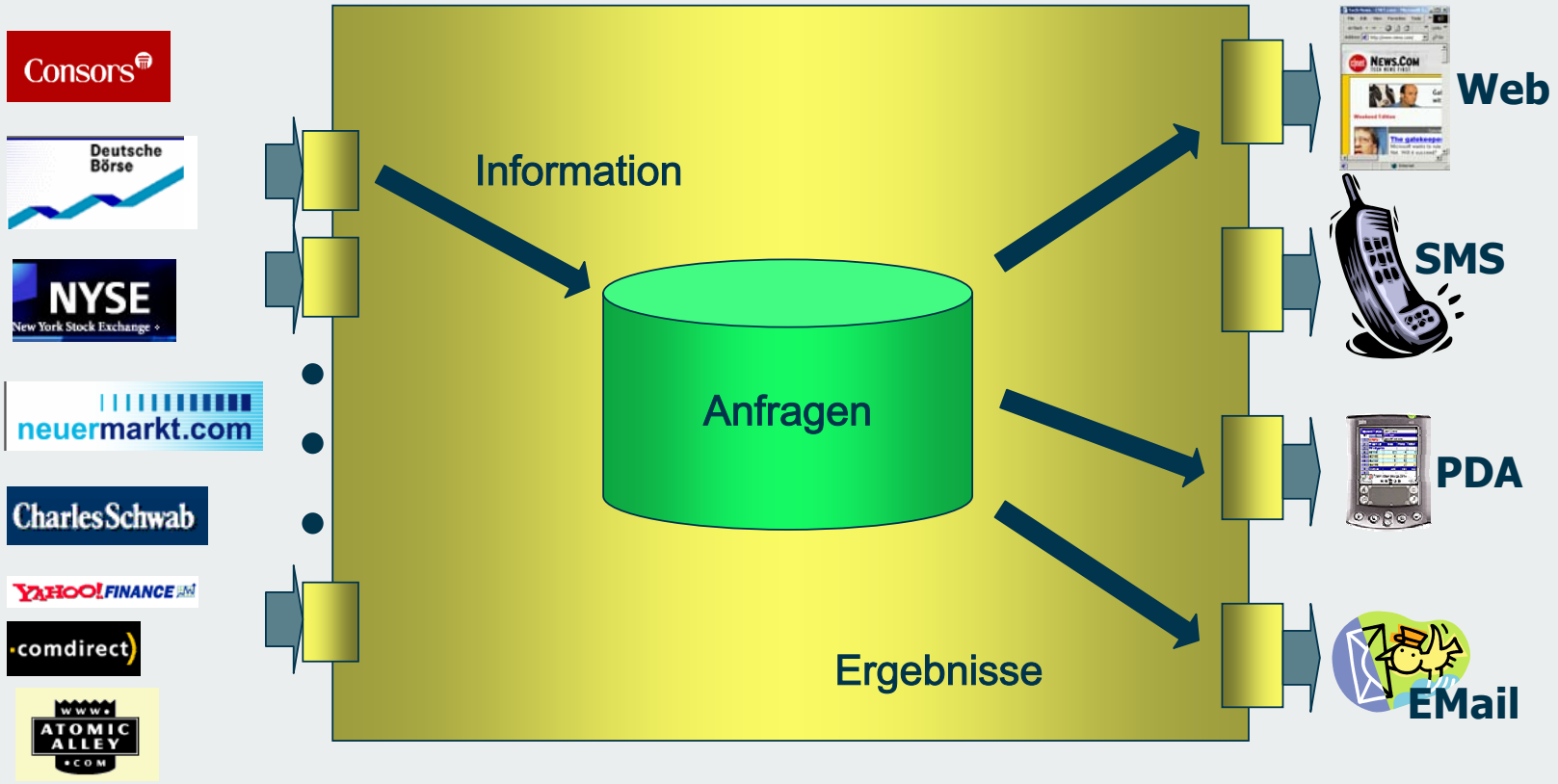
Nachfragegetriebene Informationsversorgung



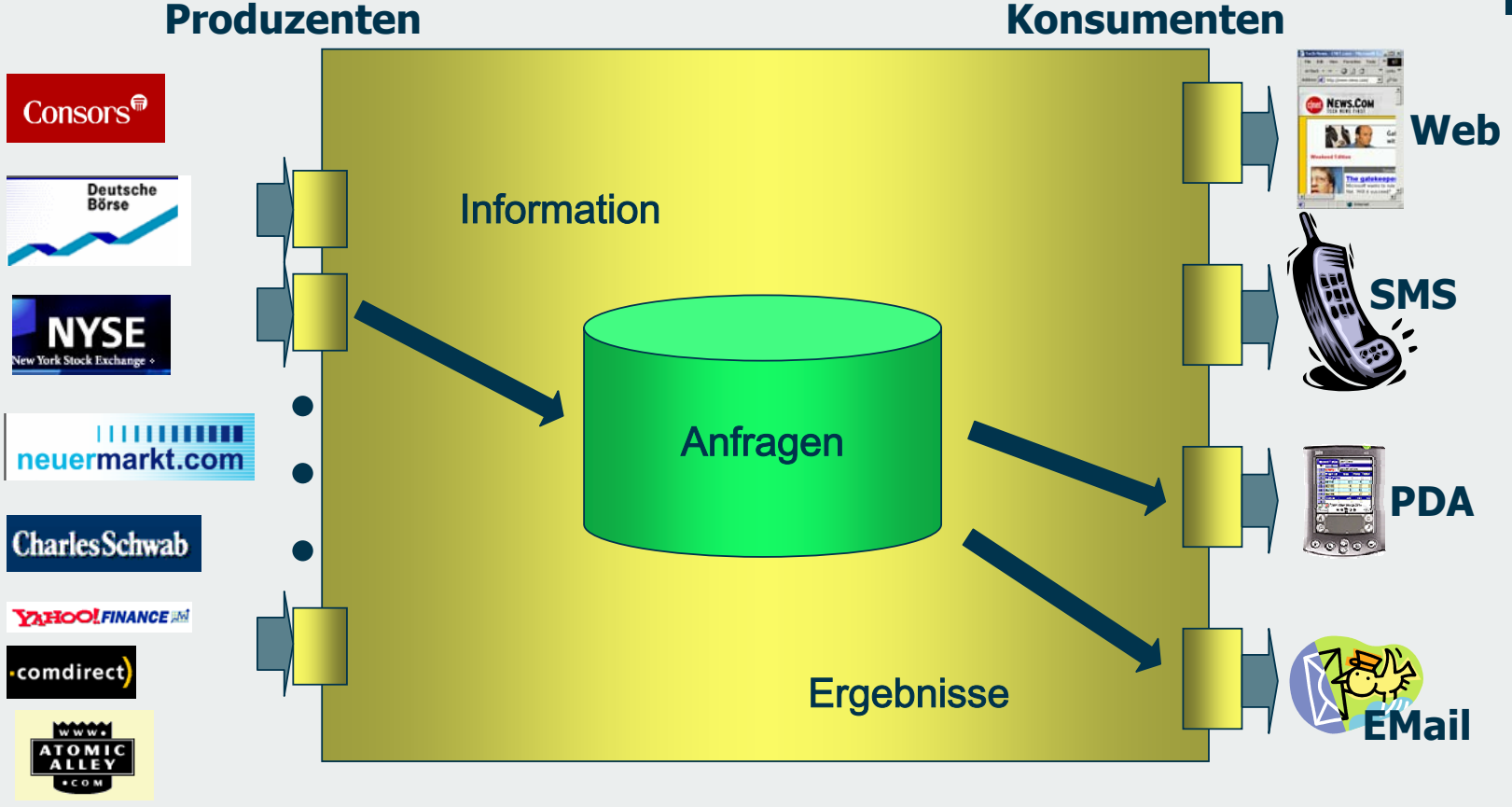
Angebotsgetriebene Informationsversorgung



Angebotsgetriebene Informationsversorgung

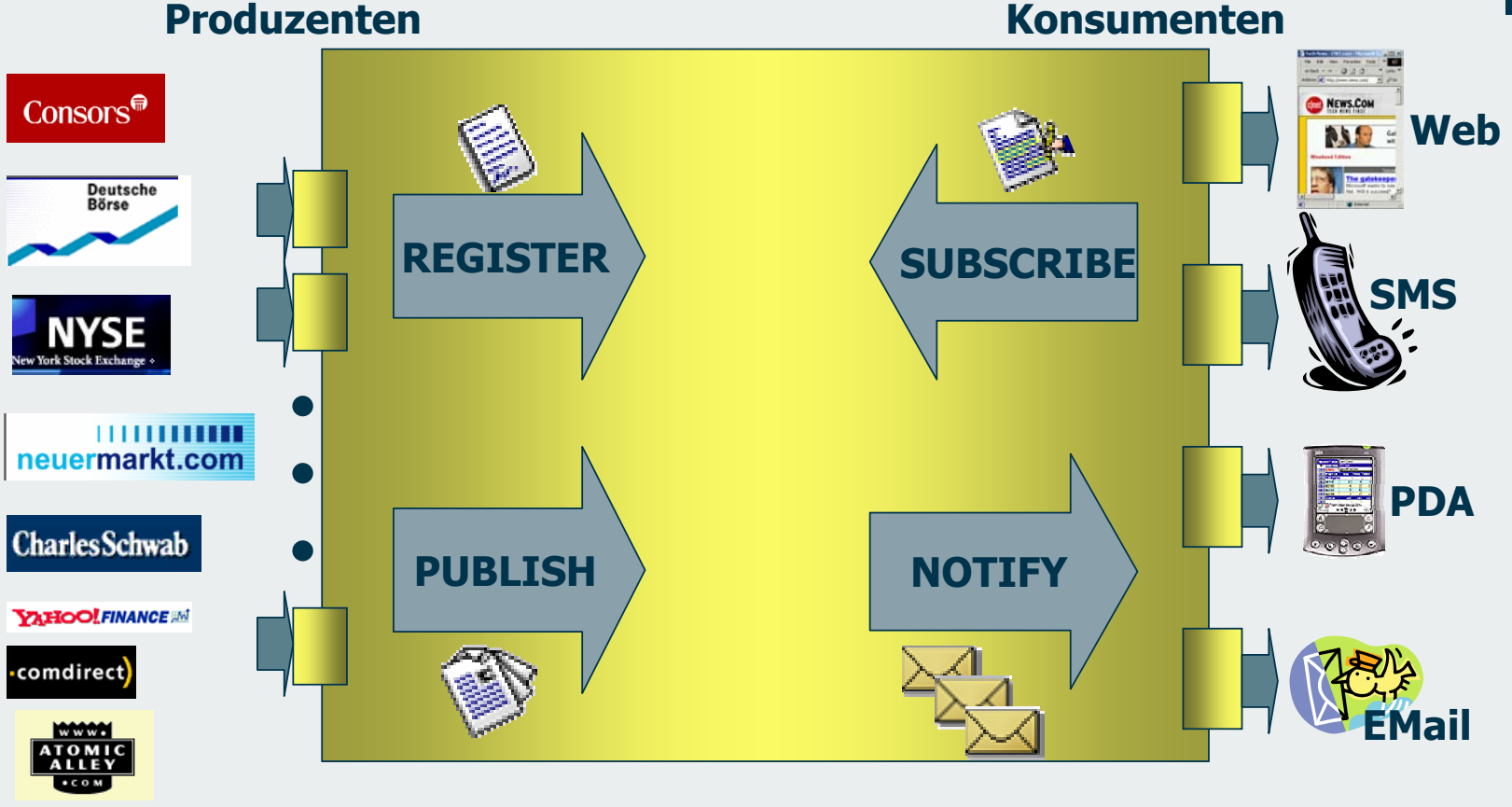


Angebotsgetriebene Informationsversorgung

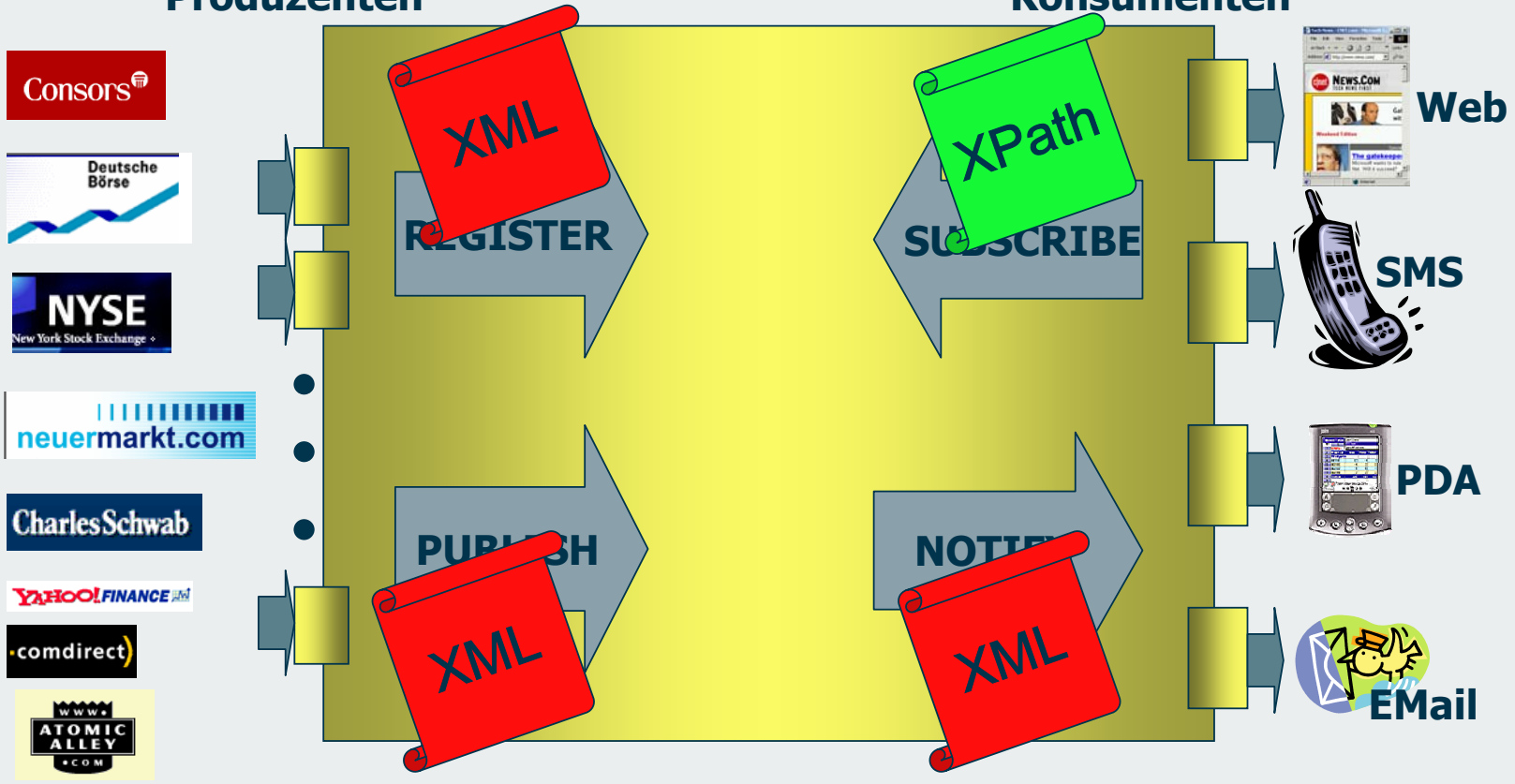


Angebotsgetriebene Informationsversorgung

Publish/Subscribe: Dienstprimitive

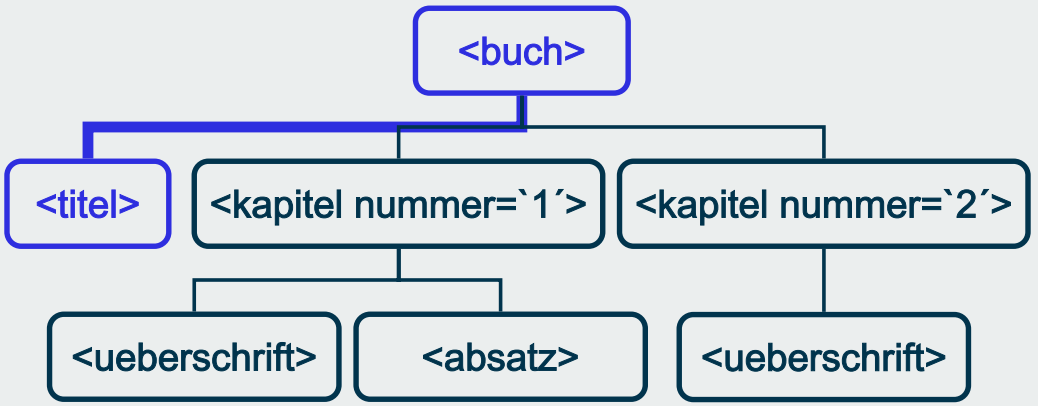


PubScribe: Dienstprimitive



XPath-Ausdrücke auf XML-Dokumenten

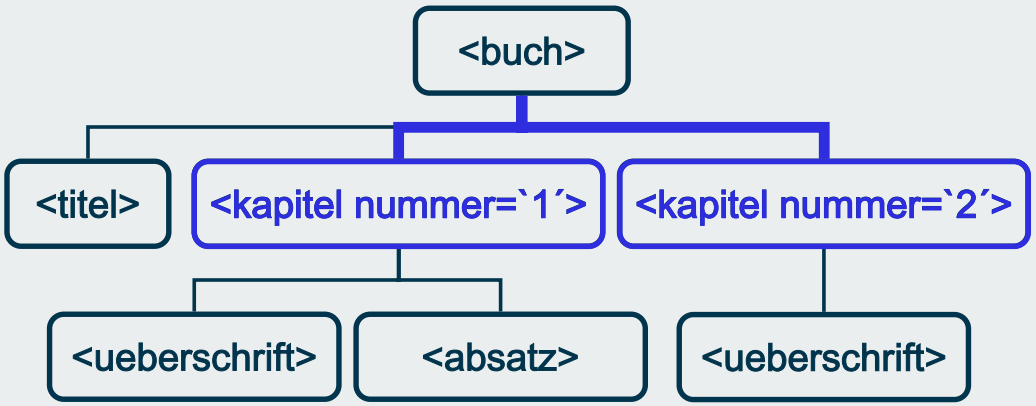
/buch/titel




```
<buch>
    <titel>
        Harry Potter
    </titel>
    <kapitel nummer='1'>
        <ueberschrift>
            Vier Freunde
        </ueberschrift>
        <absatz/>
    </kapitel>
    <kapitel nummer='2'>
        <ueberschrift>
            Eulenpost
        </ueberschrift>
    </kapitel>
</buch>
```

XPath-Ausdrücke auf XML-Dokumenten

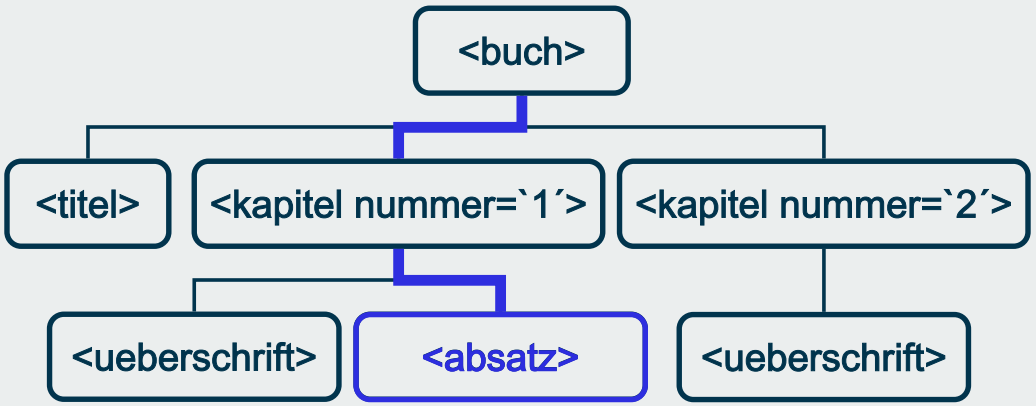
//kapitel




```
<buch>
    <titel>
        Harry Potter
    </titel>
    <kapitel nummer='1'>
        <ueberschrift>
            Vier Freunde
        </ueberschrift>
        <absatz/>
    </kapitel>
    <kapitel nummer='2'>
        <ueberschrift>
            Eulenpost
        </ueberschrift>
    </kapitel>
</buch>
```

XPath-Ausdrücke auf XML-Dokumenten

/buch/*/absatz

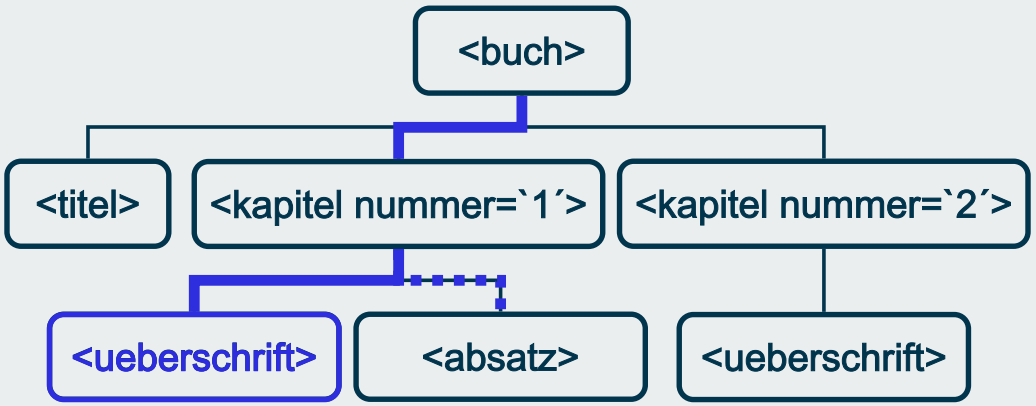



```
<buch>
    <titel>
        Harry Potter
    </titel>
    <kapitel nummer='1'>
        <ueberschrift>
            Vier Freunde
        </ueberschrift>
        <absatz/>
    </kapitel>
    <kapitel nummer='2'>
        <ueberschrift>
            Eulenpost
        </ueberschrift>
    </kapitel>
</buch>
```

XPath-Ausdrücke auf XML-Dokumenten

/buch/kapitel/absatz/../ueberschrift

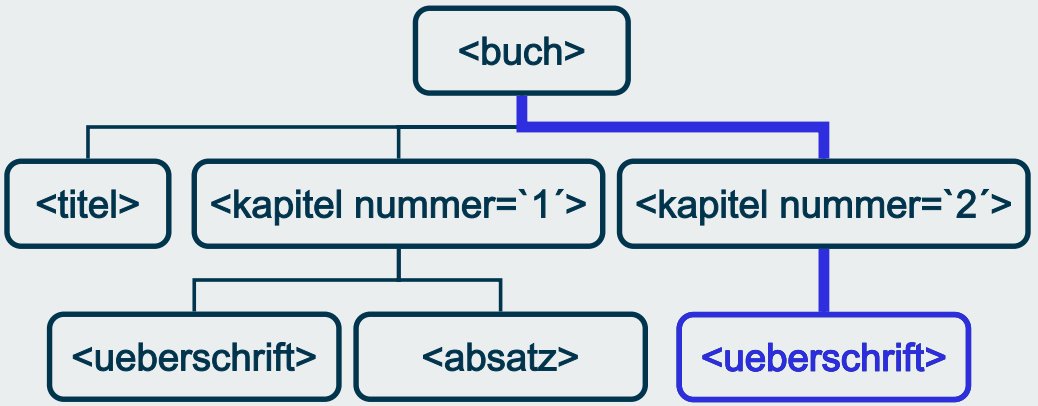
= /buch/kapitel[absatz]/ueberschrift




```
<buch>
    <titel>
        Harry Potter
    </titel>
    <kapitel nummer='1'>
        <ueberschrift>
            Vier Freunde
        </ueberschrift>
        <absatz/>
    </kapitel>
    <kapitel nummer='2'>
        <ueberschrift>
            Eulenpost
        </ueberschrift>
    </kapitel>
</buch>
```

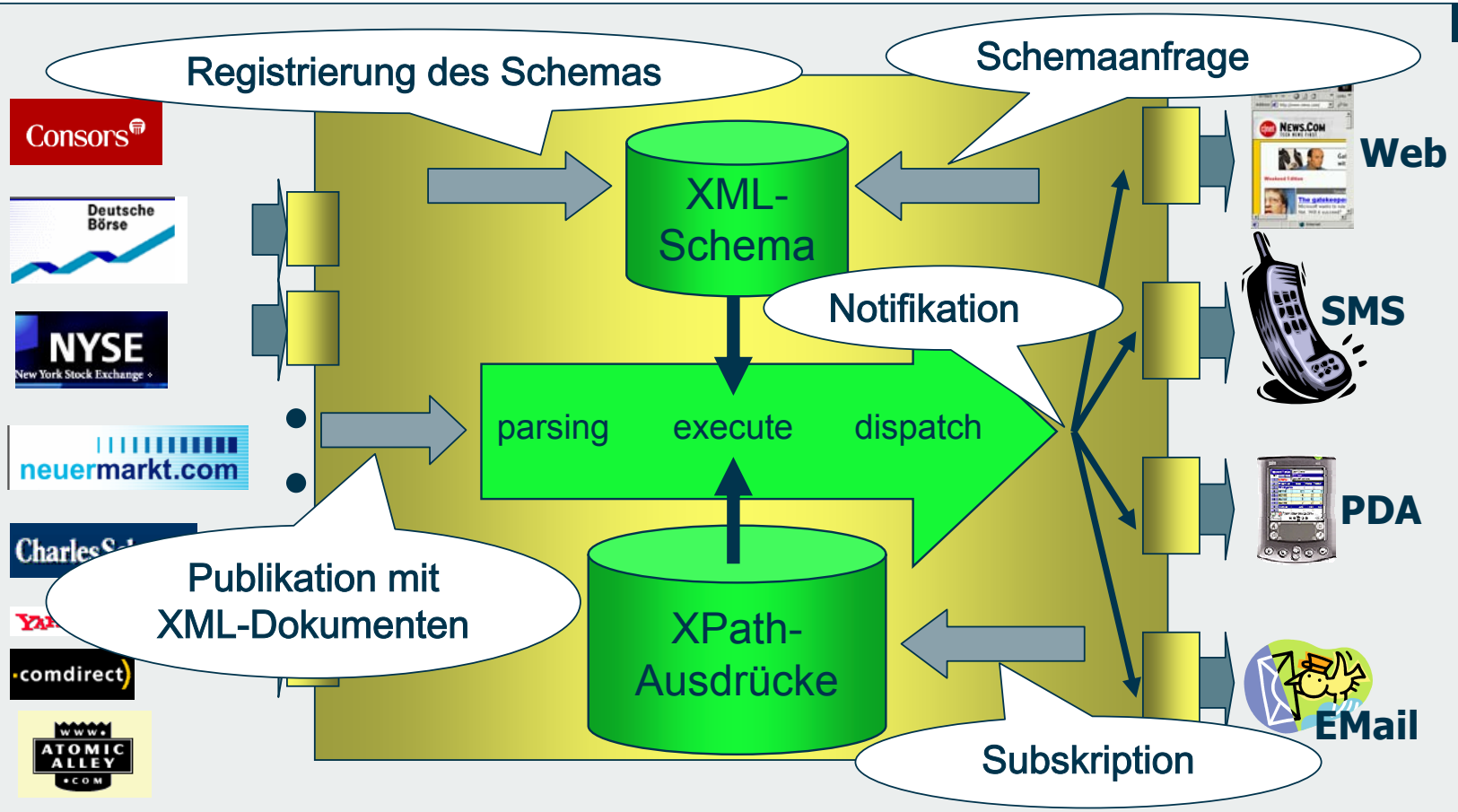
XPath-Ausdrücke auf XML-Dokumenten

/buch/kapitel[@nummer="2"]/ueberschrift




```
<buch>
    <titel>
        Harry Potter
    </titel>
    <kapitel nummer='1'>
        <ueberschrift>
            Vier Freunde
        </ueberschrift>
        <absatz/>
    </kapitel>
    <kapitel nummer='2'>
        <ueberschrift>
            Eulenpost
        </ueberschrift>
    </kapitel>
</buch>
```

XPath-Verarbeitung im PubScribe-System



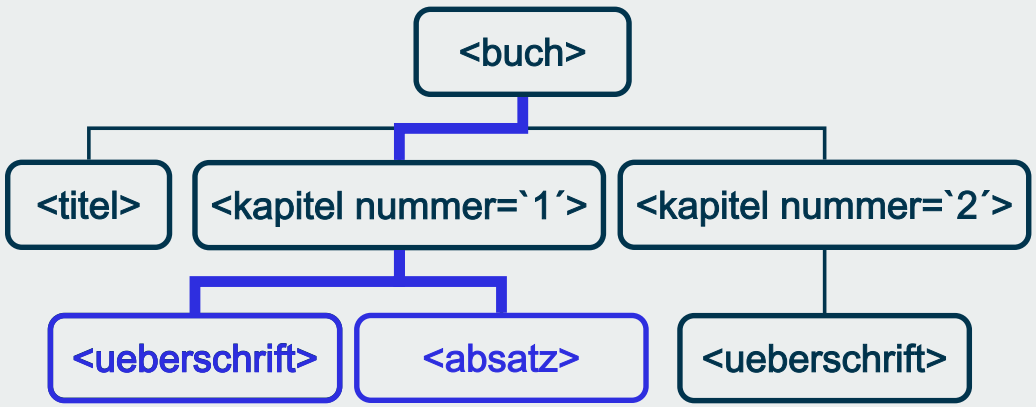
Beschleunigung der XPath-Auswertung

- Vorhandene Ansätze
  - XFilter, XTrie, ...
  - Gültigkeitsprüfung von XPath-Ausdrücken auf gesamtem XML-Dokument
- Ziel: Schnelle Auswertung der XPath-Ausdrücke
- Eigener Ansatz
  - Problem: Auswertung erfordert XML-Dokument im Hauptspeicher
  - Idee: Effiziente Reduktion des XML-Dokuments

Beispiel zur Reduktion

/buch/*/absatz

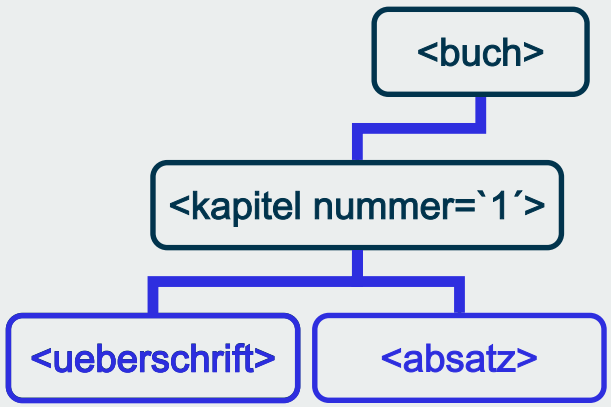
/buch/kapitel/absatz/../ueberschrift = /buch/kapitel[absatz]/ueberschrift



Beispiel zur Reduktion

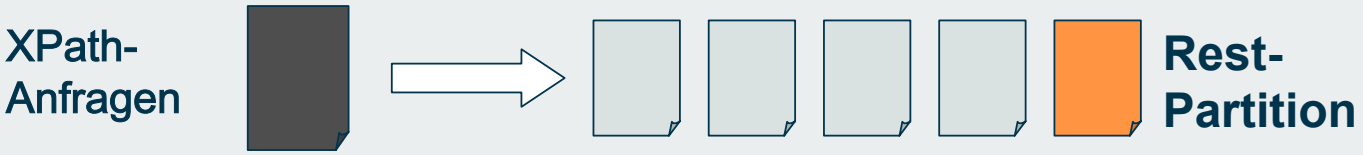
/buch/*/absatz

/buch/kapitel/absatz/../ueberschrift = /buch/kapitel[absatz]/ueberschrift

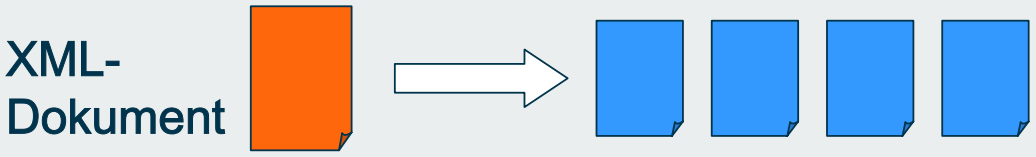


Beschleunigung der XPath-Auswertung durch Chunking

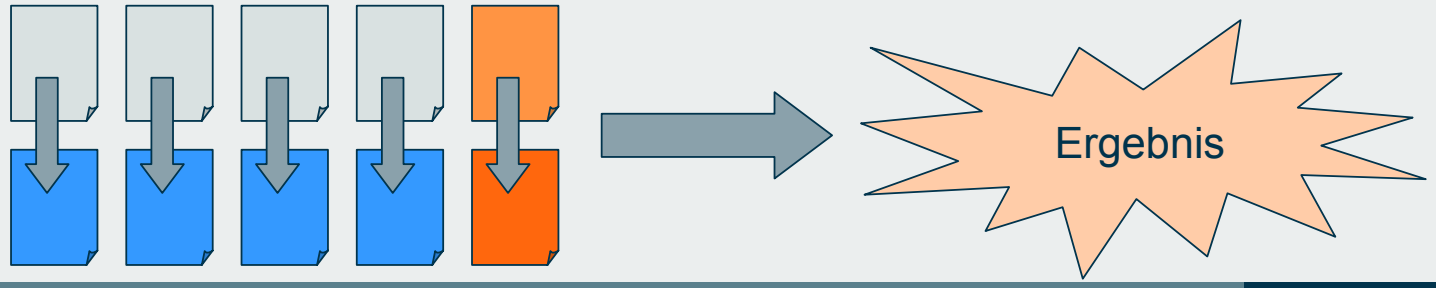
- Aufteilen der XPath-Anfragen (Präfix-Partitionen)



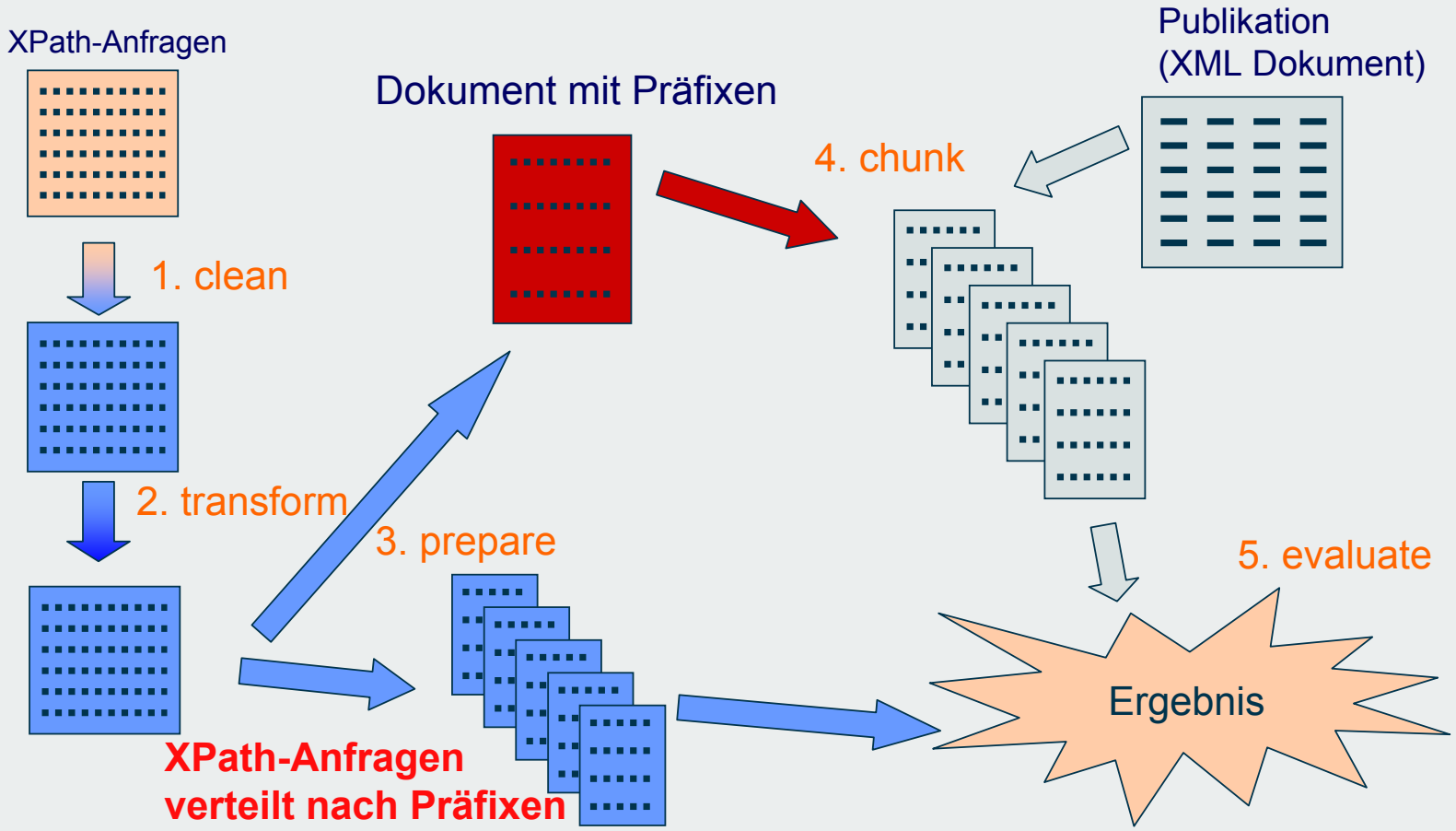
- Erzeugen von XML-Chunks



- Auswerten der XPath-Partitionen auf den Chunks



Prototyp “eXtract” im PubScribe-System



Betrachtung der einzelnen Schritte

- Clean-Schritt (Beseitigung der „parent“ - Ausdrücke)

  Beispiel a/b/../c → a[b]/c

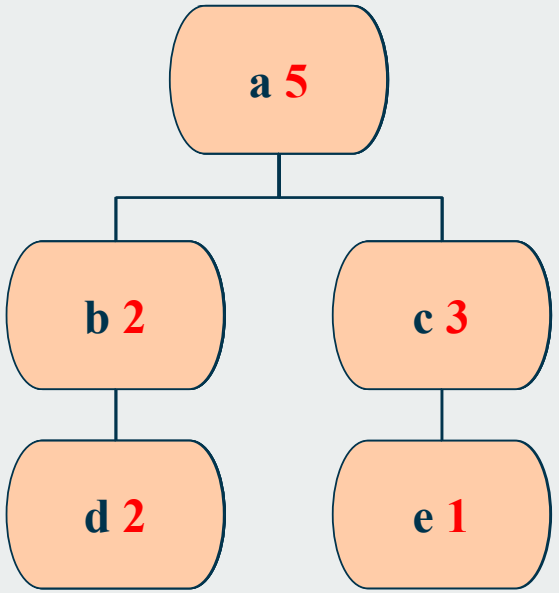
- Transform-Schritt (Auflösen von Wildcards (*) und //-Ausdrücken)

  Beispiel //c → /a/b//c und /a/d/e//c

  /a/*/b → /a/c/b, /a/e/b und /a/f/b

- Prepare-Schritt
  - Einfügen der XPath-Anfragen in Präfixbaum
  - Keine Berücksichtigung der Prädikate
  - Beispiel…

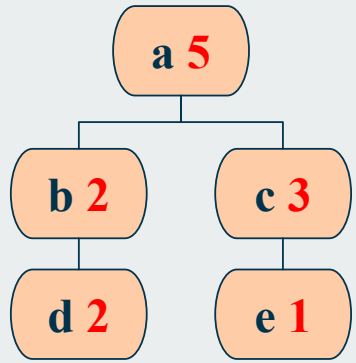
Beispiel: Einfügen in den Präfixbaum



/a/b/d
/a/c
/a[c]/b/d
/a/c/e
/a/c

Phase der Präfixfindung

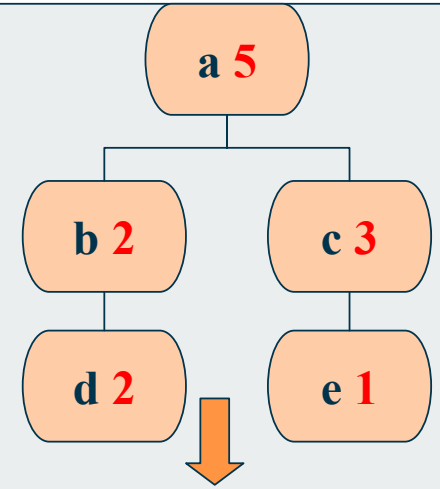
- Naive Präfixfindung
  - Sortierung der Knoten nach ihren Gewichten
  - Wahl der TOP(n)-Knoten als Chunk-Kriterium
  - Beispiel: /a und /a/c



- Alternative Präfixfindung

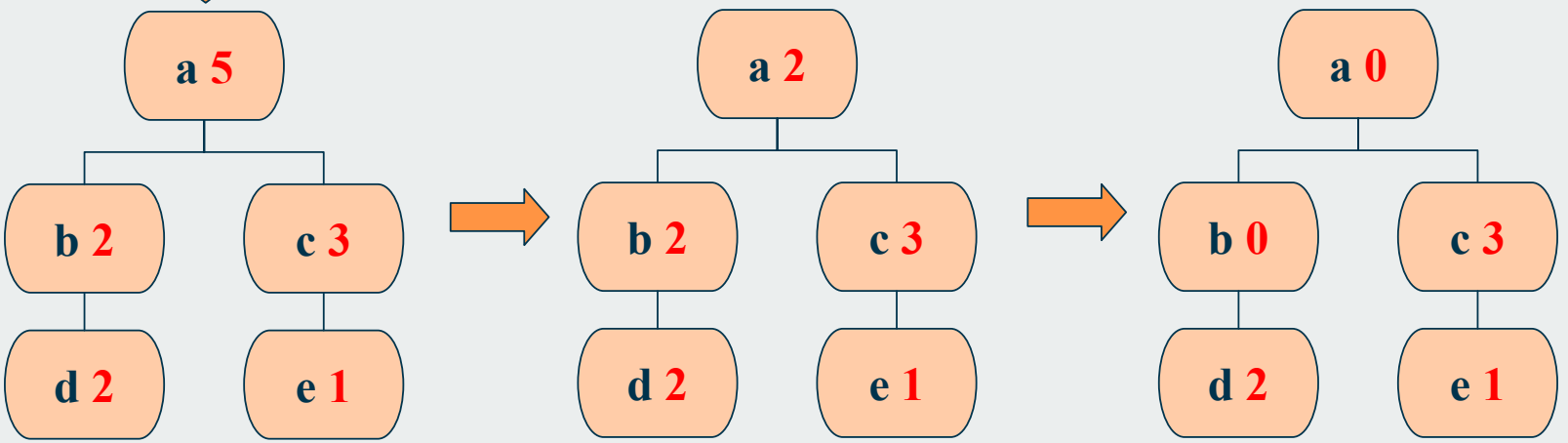
1. Ermittle Knoten mit größtem Gewicht = Kandidat
   falls mehrere Knoten gleich schwer sind → wähle Knoten mit größter Tiefe
2. Subtraktion des Gewichts von allen Knoten auf dem Weg zur Wurzel
3. Entfernung bereits selektierter – jetzt leichterer - Kandidaten
4. Wiederholung, bis „Anzahl Kandidaten = Anzahl Chunks“

Phase der Präfixfindung am Beispiel



Gesucht: 2 Präfixe (2 Chunks)

Präfix: <a> <c> <d>



Zuordnung der XPath-Anfragen

Präfixe: /a/c /a/b/d

Anfragen:

/a/b/d

/a/c

/a[c]/b/d

/a/c/e

/a/c

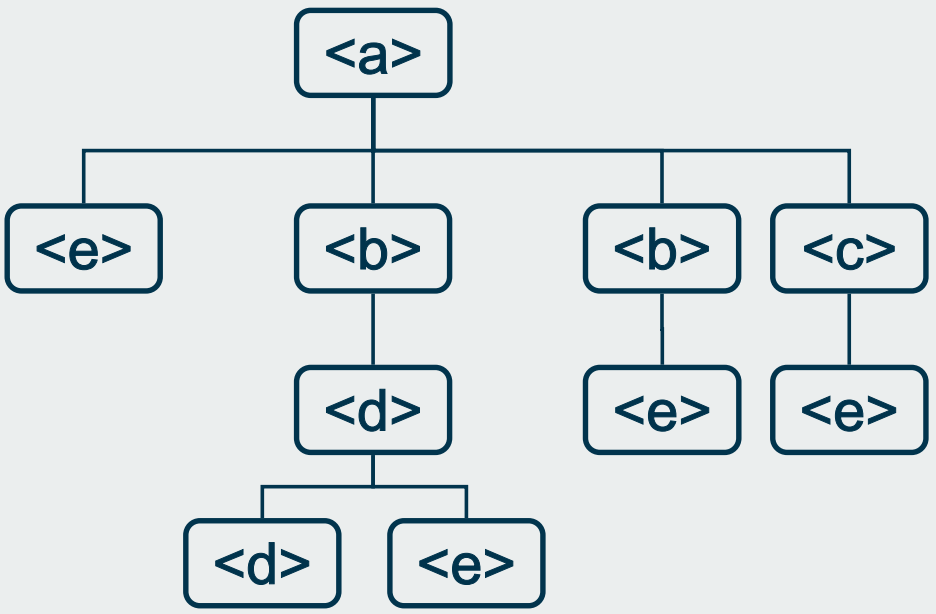
Zuordnung der XPath-Anfragen

Präfixe:	/a/c	/a/b/d
Anfragen:	/a/c	/a/b/d
	/a/c/e	/a[c]/b/d
	/a/c	

Chunk-Bildung

Präfix: /a/b/d

Anfragen: /a/b/d, /a[c]/b/d

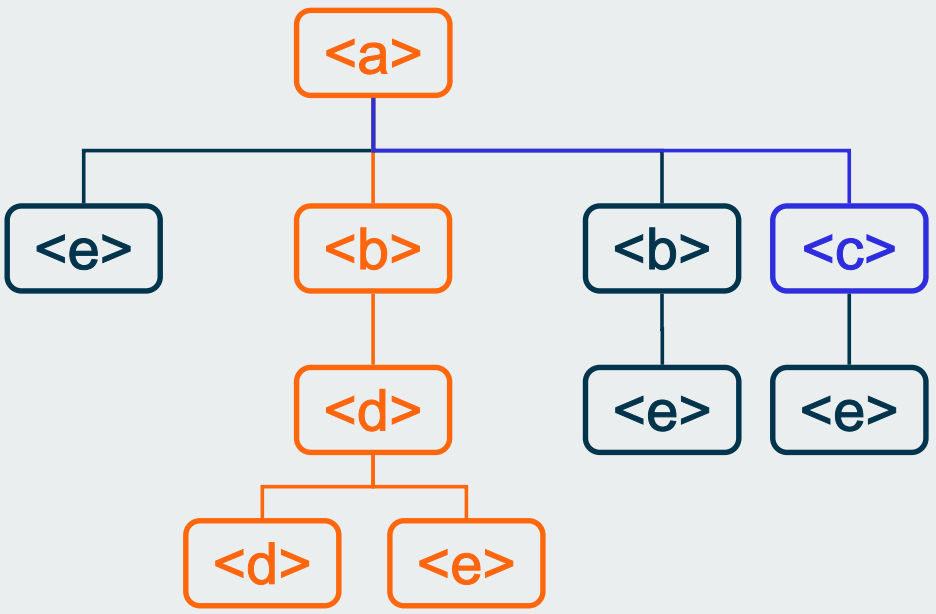



```
<a>
    <e/>
    <b>
        <d>
            <d/>
            <e/>
        </d>
    </b>
    <b>
        <e/>
    </b>
    <c>
        <e/>
    </c>
</a>
```

Chunk-Bildung

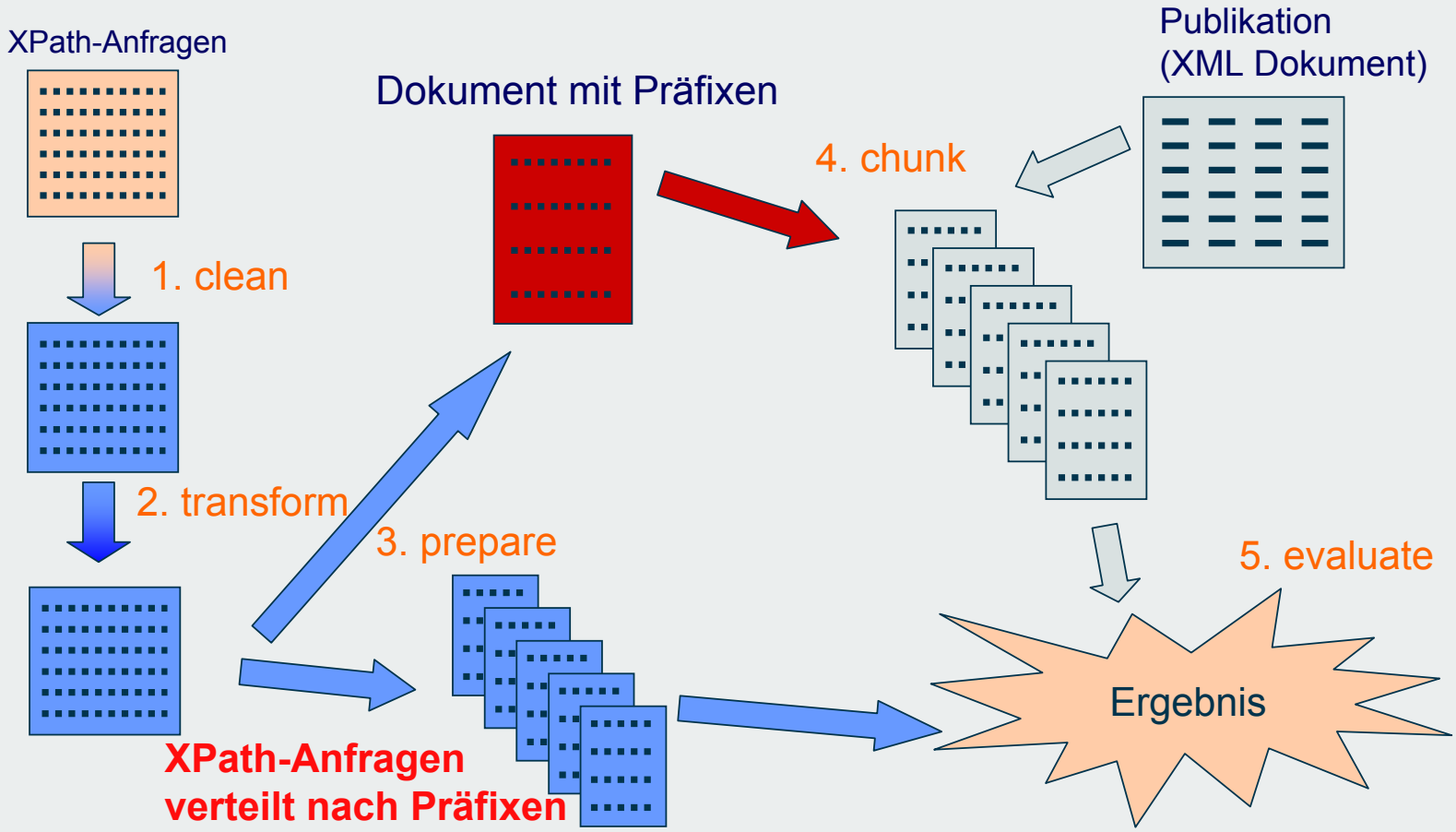
Präfix: /a/b/d

Anfragen: /a/b/d, /a[c]/b/d

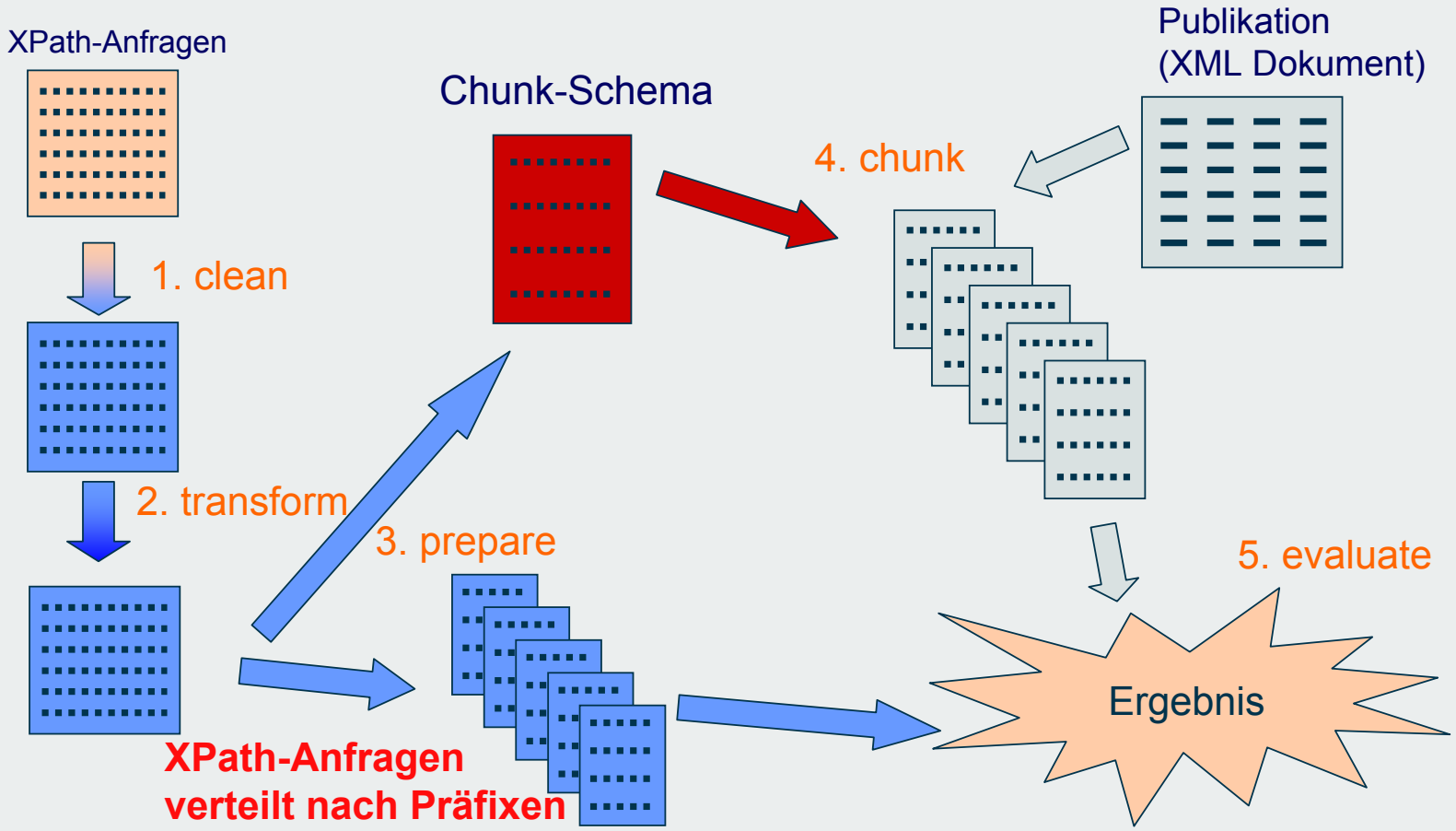



```
<a>
    <e/>
    <b>
        <d>
            <d/>
            <e/>
        </d>
    </b>
    <b>
        <e/>
    </b>
    <c>
        <e/>
    </c>
</a>
```

Prototyp “eXtract” im PubScribe-System



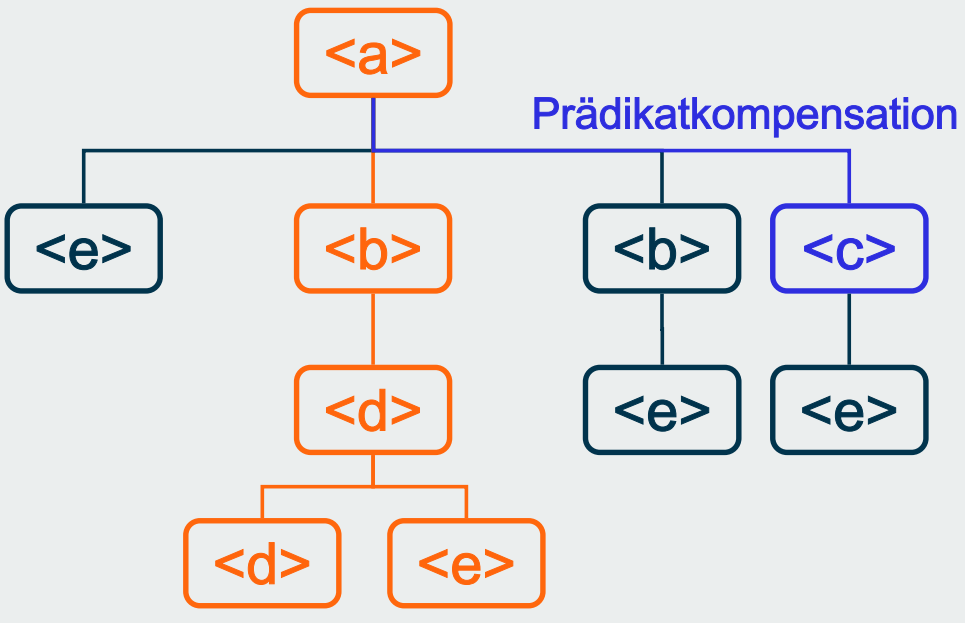
Prototyp “eXtract” im PubScribe-System

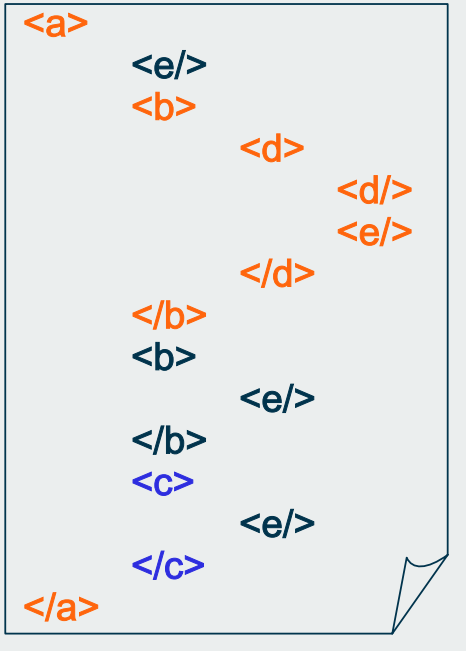


Chunk-Bildung

Präfix: /a/b/d

Anfragen: /a/b/d, /a[c]/b/d



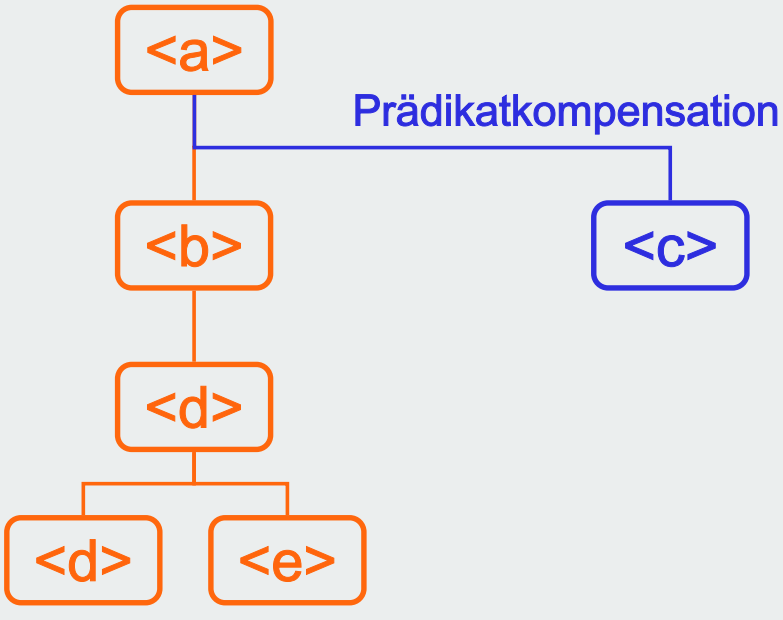


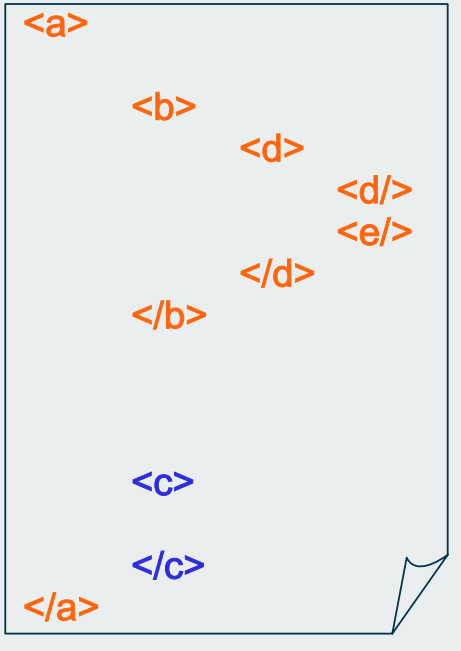
→ Präfix /a/b/d + Prädikatkompensation /a/c

Chunk-Bildung

Präfix: /a/b/d

Anfragen: /a/b/d, /a[c]/b/d

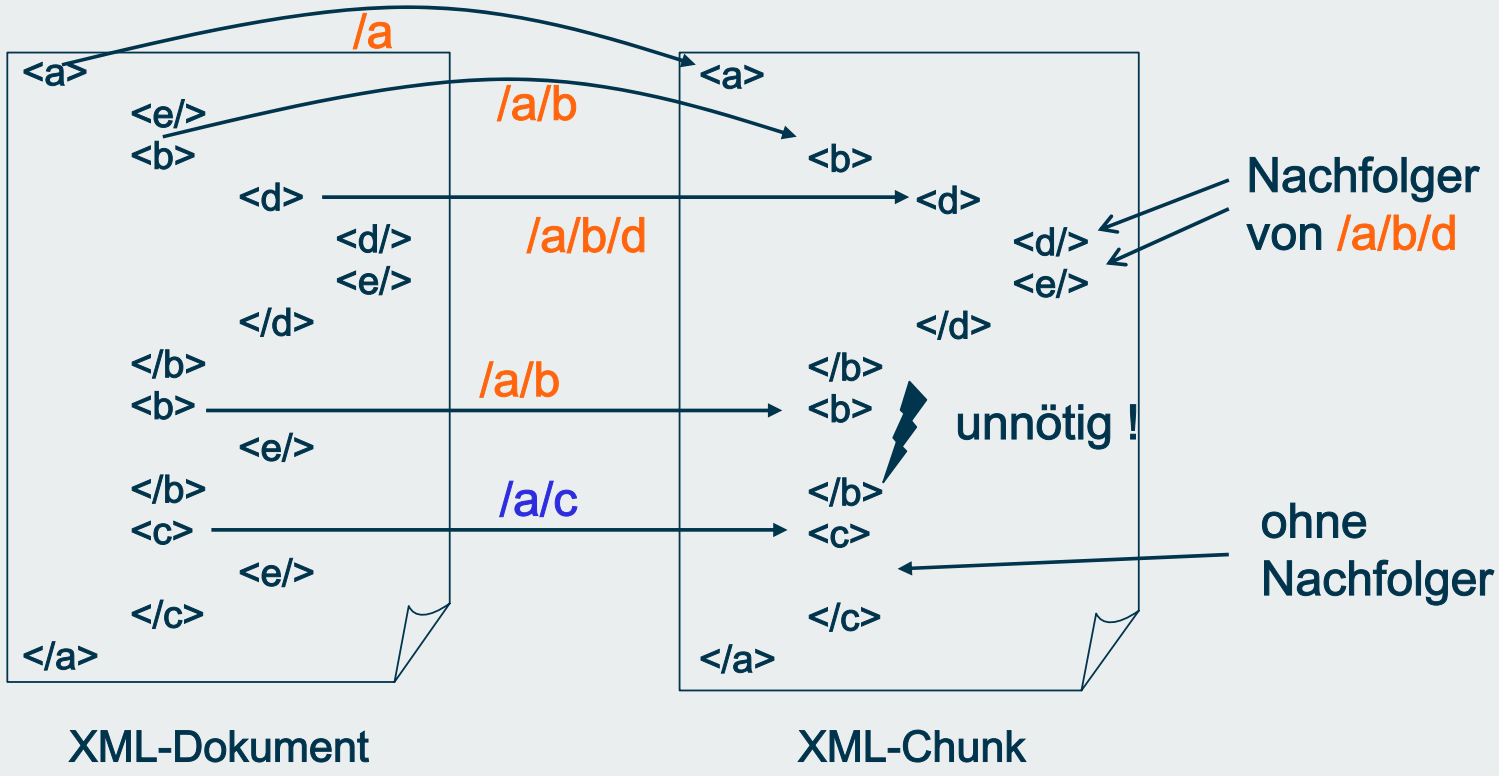




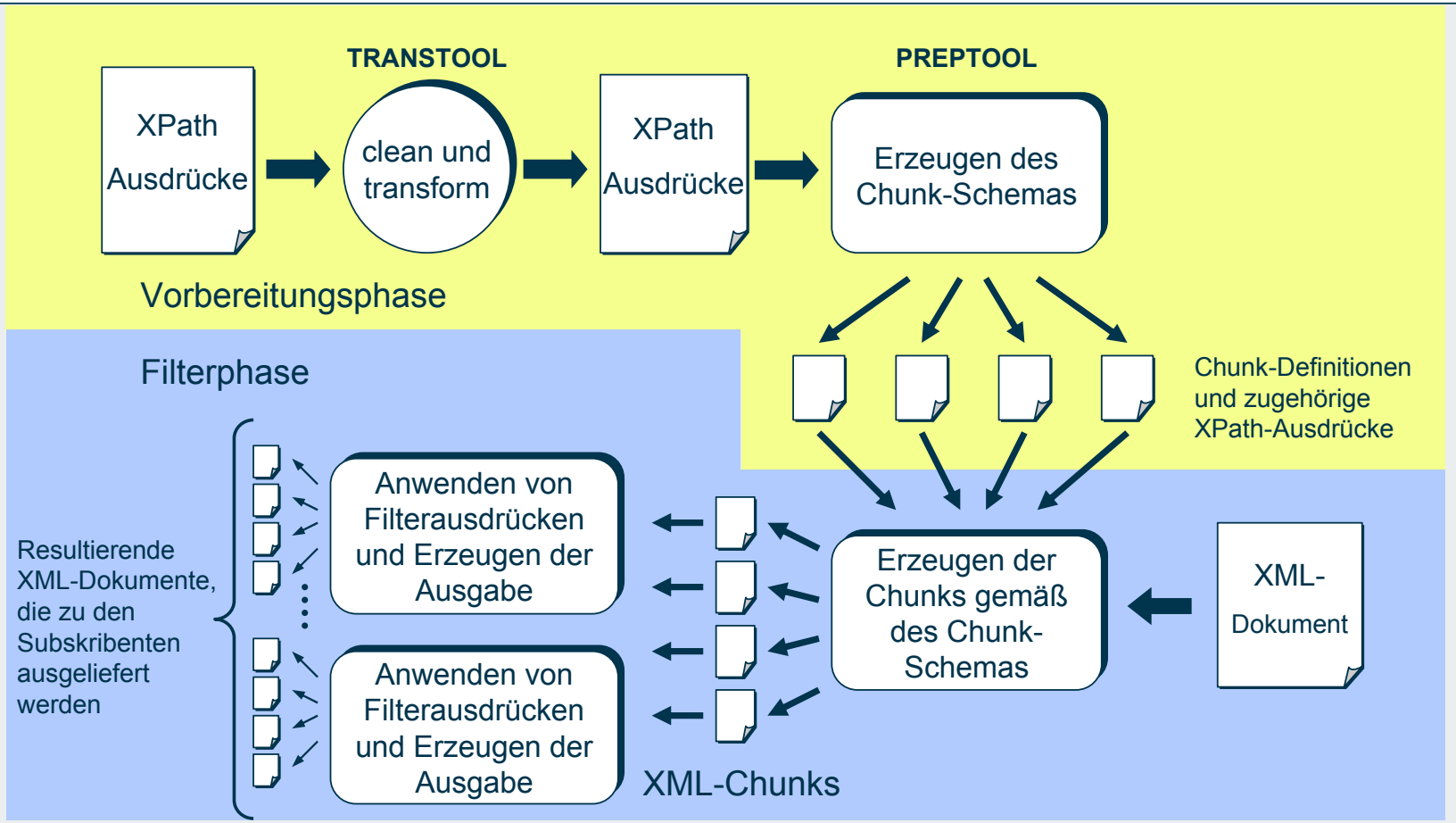
→ Präfix /a/b/d + Prädikatkompensation /a/c

Chunk-Bildung

Präfix /a/b/d, Prädikatkompensation /a/c



Realisierung der eXtract-Chunking-Strategie



Performanzmessungen

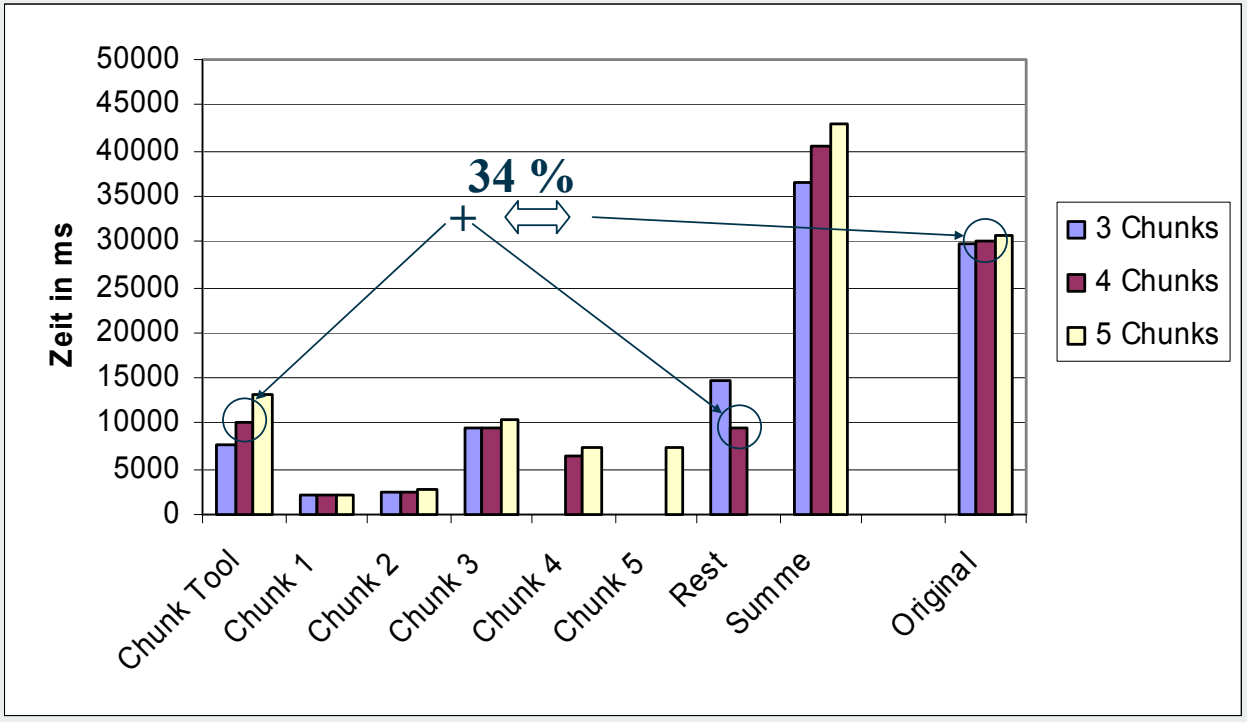
Struktur des Beispieldokuments:

```
<!ELEMENT db (student+,professor+,assistant+,secretary+,lecturer+)  >

<!ELEMENT student (name,email?,url?,link?)>
<!ELEMENT professor (name,email?,url?,link?)>
<!ELEMENT assistant (name,email?,url?,link?)>
<!ELEMENT secretary (name,email?,url?,link?)>
<!ELEMENT lecturer (name,email?,url?,link?)>

<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT link (#PCDATA)>
```

Kardinalität der Chunks:

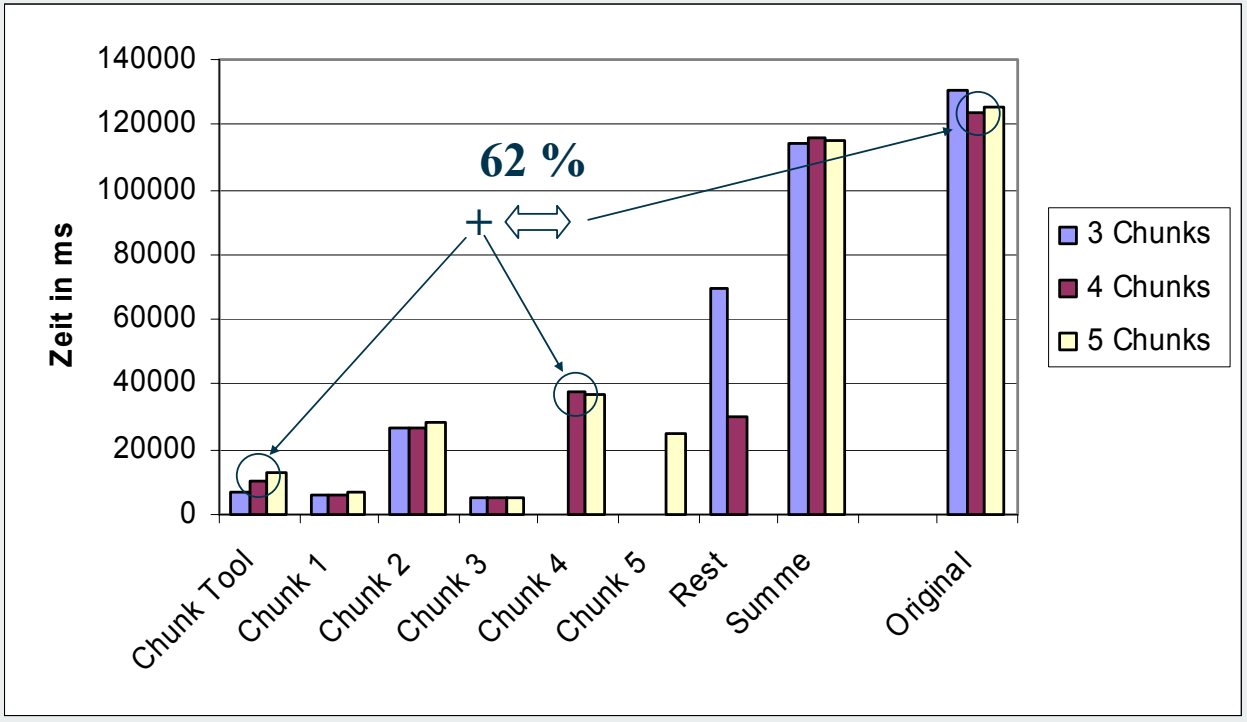
Unterbaum	Größe
student	1500KB
professor	1110KB
assistant	206KB
secretary	1200KB
lecturer	170KB

Performanzmessungen



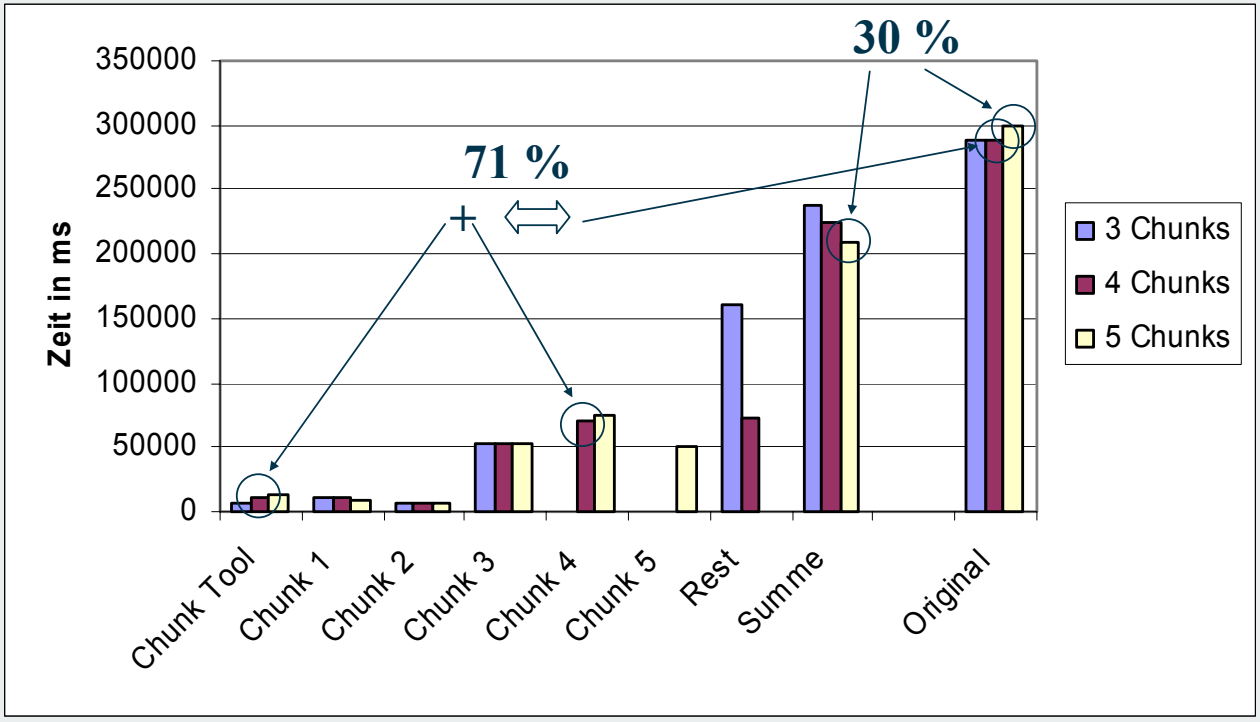
100 Anfragen

Performanzmessungen



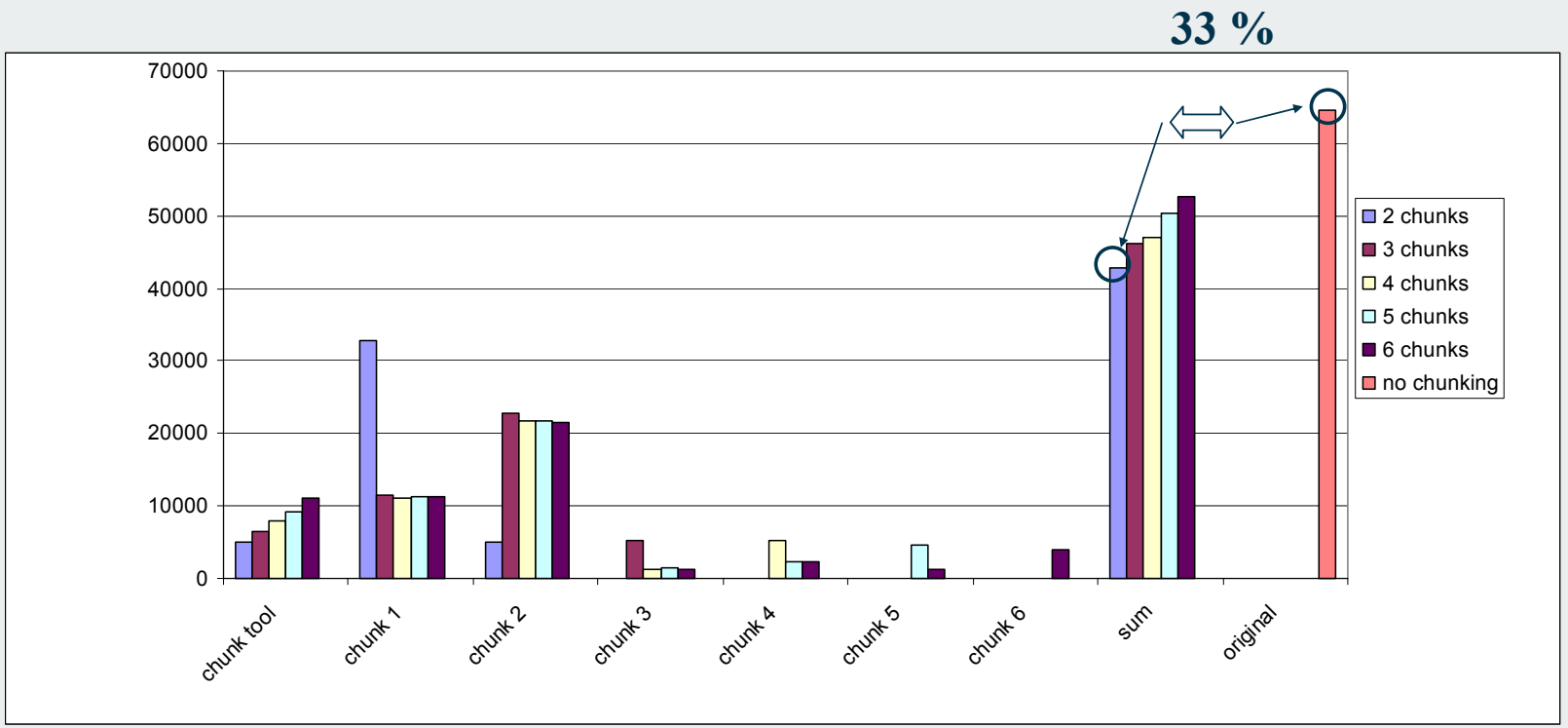
500 Anfragen

Performanzmessungen



2000 Anfragen

Weiteres Szenario:
EJB Deployment Deskriptor



100 Anfragen

Zusammenfassung

- Ausgangssituation: PubScribe-System
  - Publikation von XML-Dokumenten
  - Subskription mittels XPath-Ausdrücken
- Ziel: Schnelle Auswertung vieler XPath-Anfragen
- Lösung: Chunk-Bildung
  - Gruppierung der XPath-Ausdrücke nach Präfixen (= Partitionen)
  - Auswertung der Partitionen auf den Chunks
- Performanzgewinn
  - Sequenziell: >30%
  - Parallel: >50 %

Danke für Ihre Aufmerksamkeit

Weitere Informationen unter http://www.irmert.de/extract

Fragen ?