Speicherung von XML in (objekt-)relationalen Datenbanken

Burkhard Schäfer

Übersicht

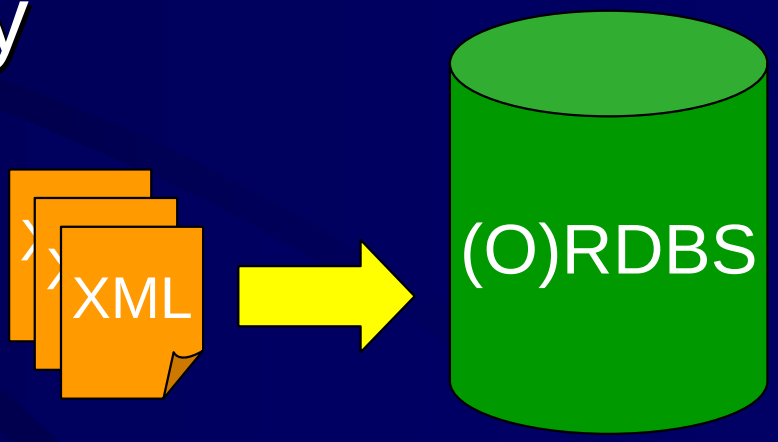
- Motivation
- Anforderungen
- Ansätze
  - modellorientiert
  - strukturorientiert
- Zusammenfassung

Motivation

Warum XML in Datenbanken speichern?

Einsatz bewährter Datenbanktechnologie:

- Transaktionsmanagement
- Logging und Recovery
- Synchronisation
- Integritätsregeln



Anforderungen (1)

- Definition einer Abbildung von XML-Strukturen in ein Datenbankschema
- Transformation von Dokumenten
- Transformation von Anfragen

Anforderungen (2)

Definition eines Abbildungsschemas

- Rückgewinnung des Originaldokuments möglich
- Unterstützung von Anfragen
- Erhaltung der Reihenfolge
- Formale Beschreibung der Abbildung

- Generisch oder Schema-spezifisch

Anforderungen (3)

Transformation von Dokumenten

- Zerlegung in Fragmente (“shredding”)
- Ablage der Fragmente in
  - mehreren Tupeln (RDBS)
  - geschachteltem Tupel (ORDBS)

Anforderungen (4)

Transformation von Anfragen

- Formulierung in XML-typischen Sprachen (XQuery, XPath, XML-QL)
- Umsetzung in SQL
- Transformation der Ergebnismenge nach XML

Naiver Ansatz

- Speichern des XML-Dokuments in einem BLOB-Feld

```
Documents (docID, docContent)
```

Naiver Ansatz

- Speichern des XML-Dokuments in einem BLOB-Feld

Documents (docID, docContent)

docID	docContent
4711	<buch> <titel>Professional XML</titel> ... </buch>

Naiver Ansatz

- Speichern des XML-Dokuments in einem BLOB-Feld

  Documents (docID, docContent)

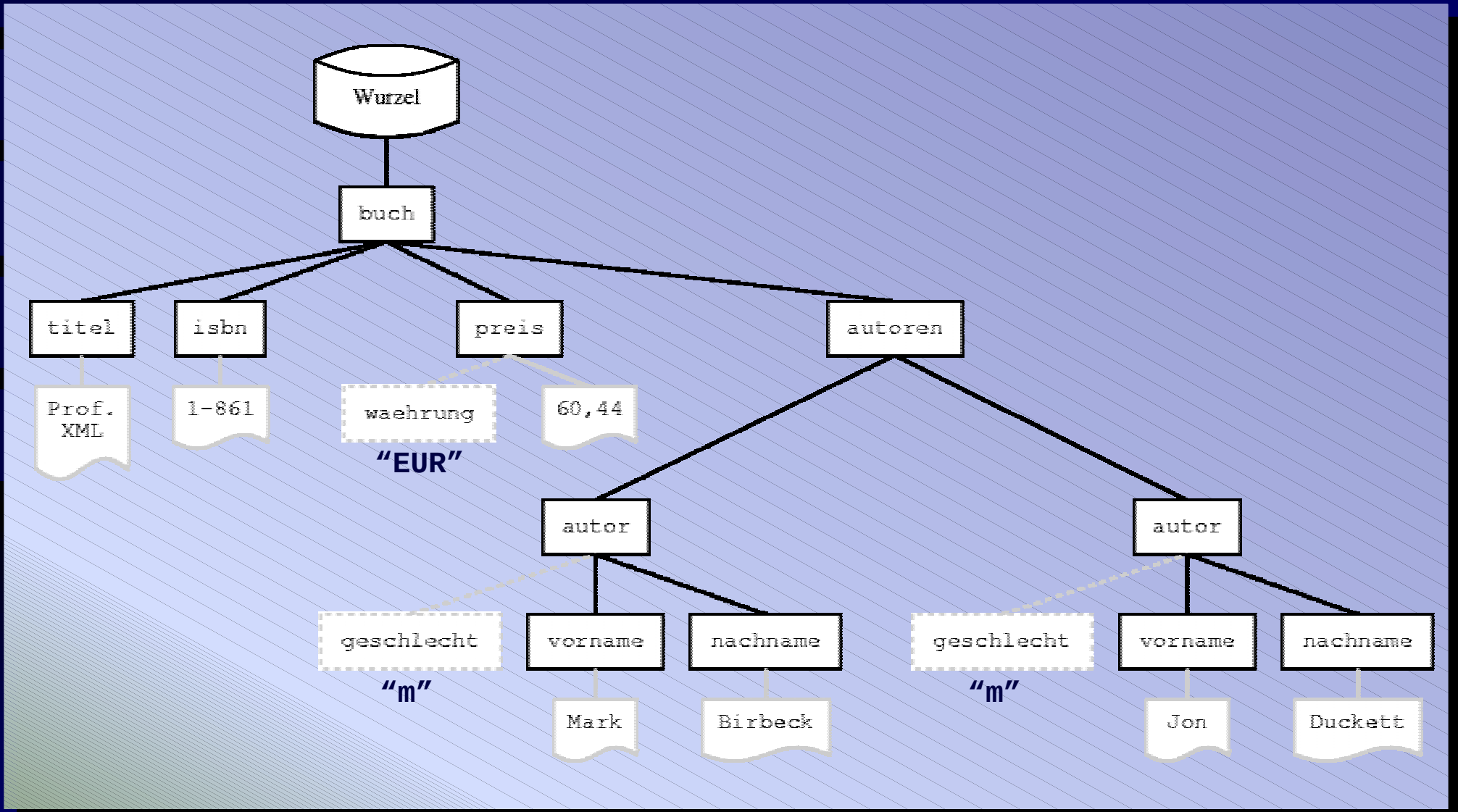
docID	docContent
4711	<buch> <titel>Professional XML</titel> ... </buch>

- niedriger Verwaltungsaufwand
- schwierige Anfrageverarbeitung

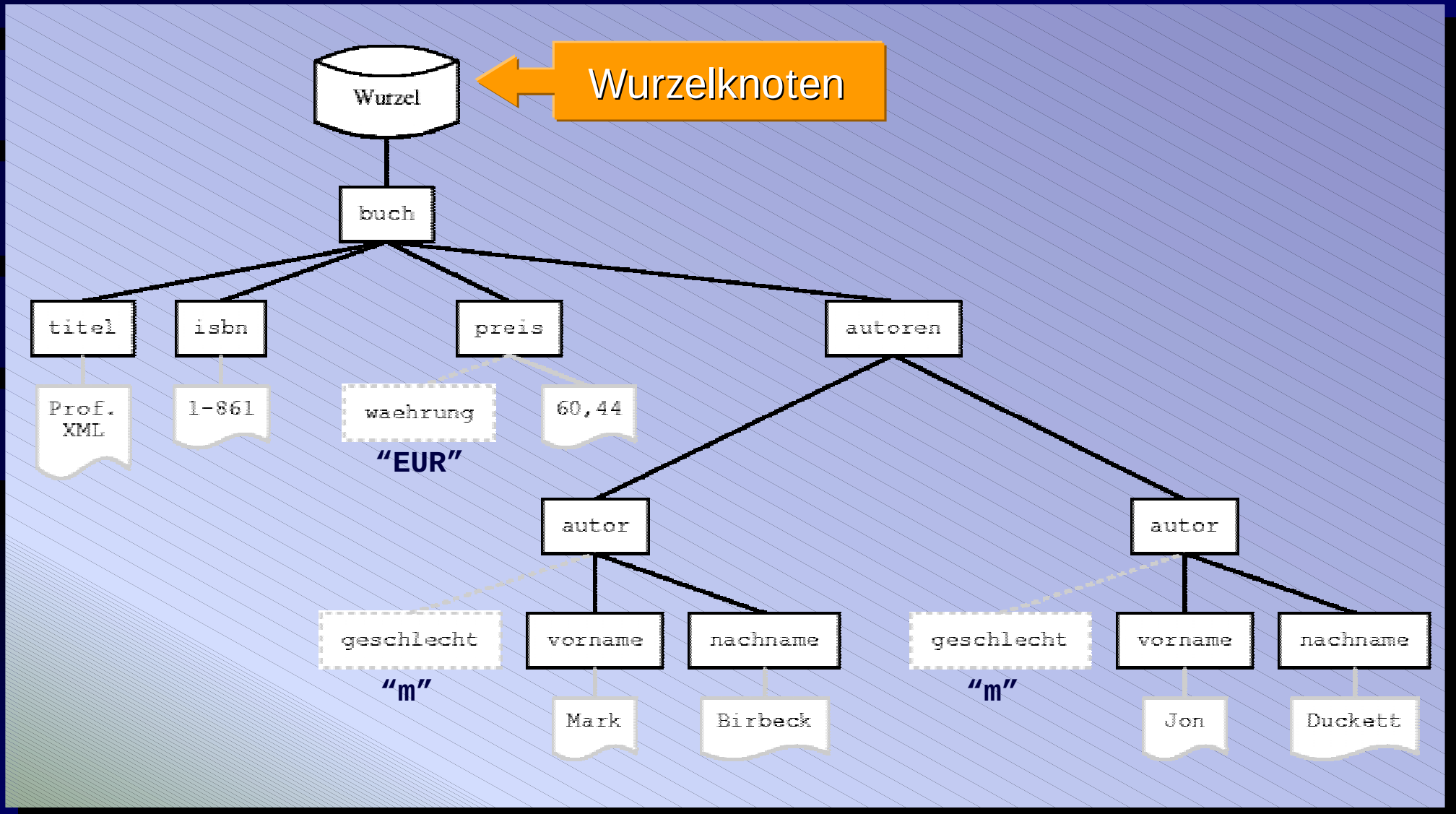
Beispieldokument

```
<buch>
   <titel>Professional XML</titel>
   <isbn>1-861005-05-9</isbn>
   <preis waehrung="EUR">60,44</preis>
   <autoren>
      <autor geschlecht="m">
         <vorname>Mark</vorname>
         <nachname>Birbeck</nachname>
      </autor>
      <autor geschlecht="m">
         <vorname>Jon</vorname>
         <nachname>Duckett</nachname>
      </autor>
   </autoren>
</buch>
```

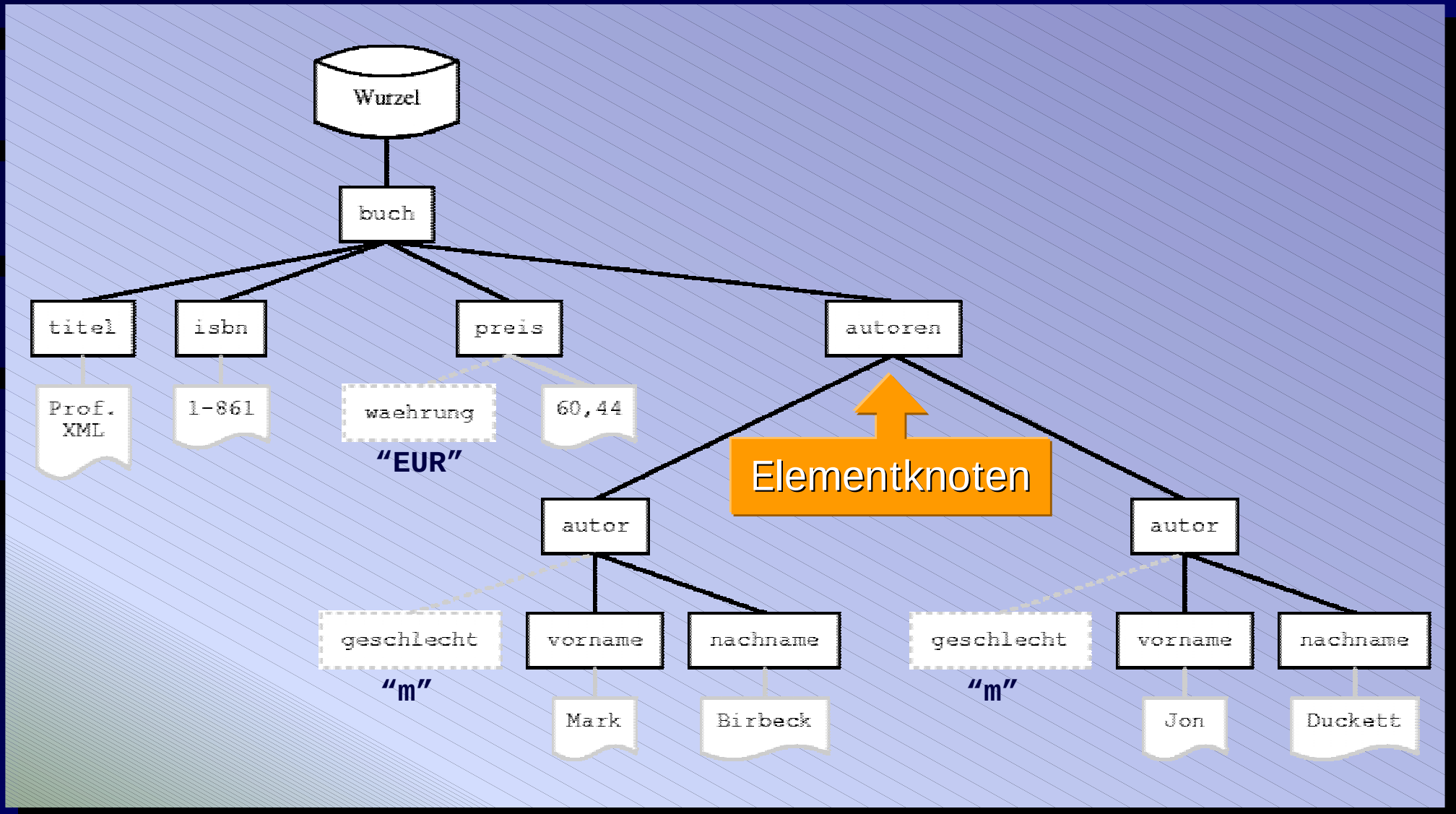
Baumdarstellung



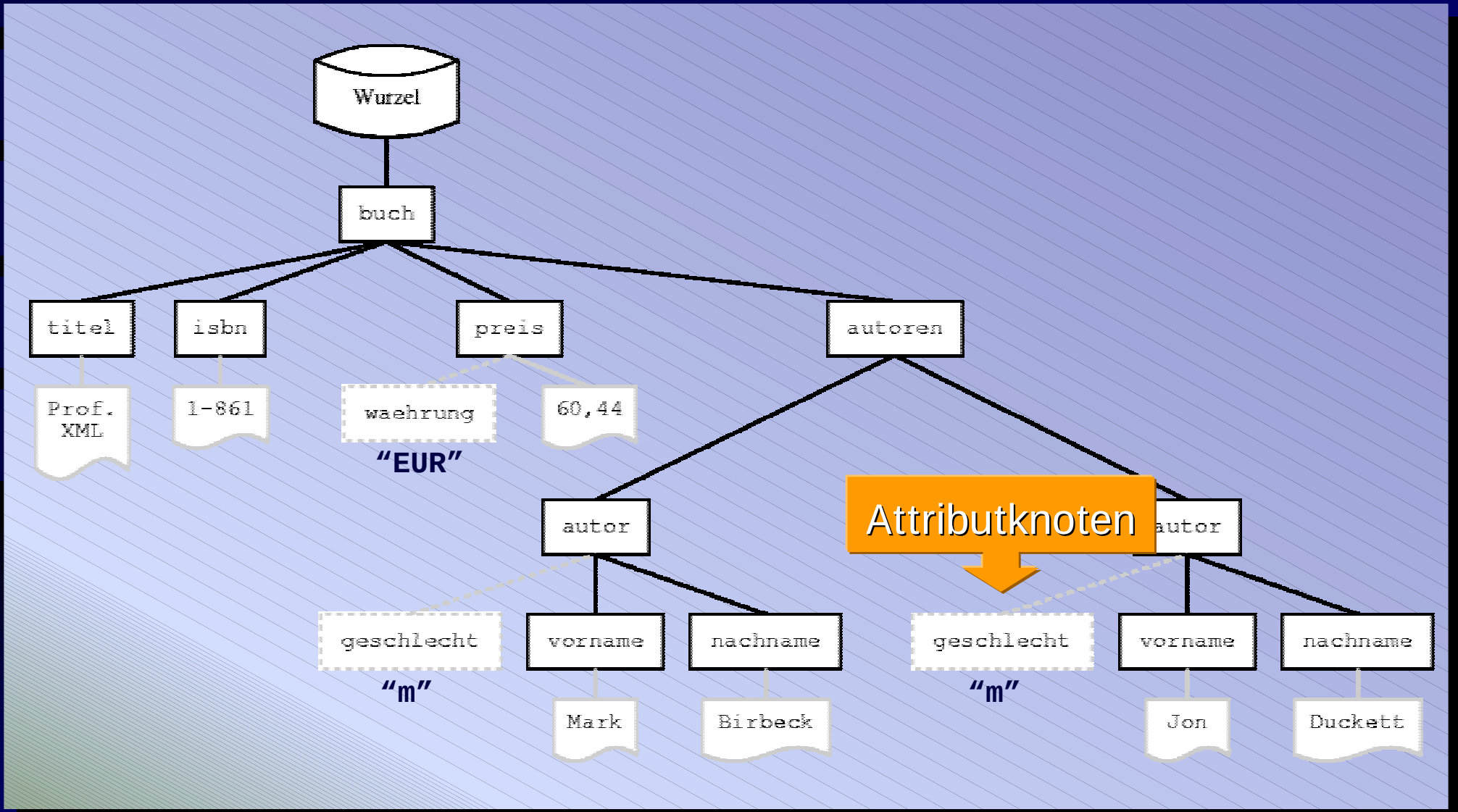
Baumdarstellung



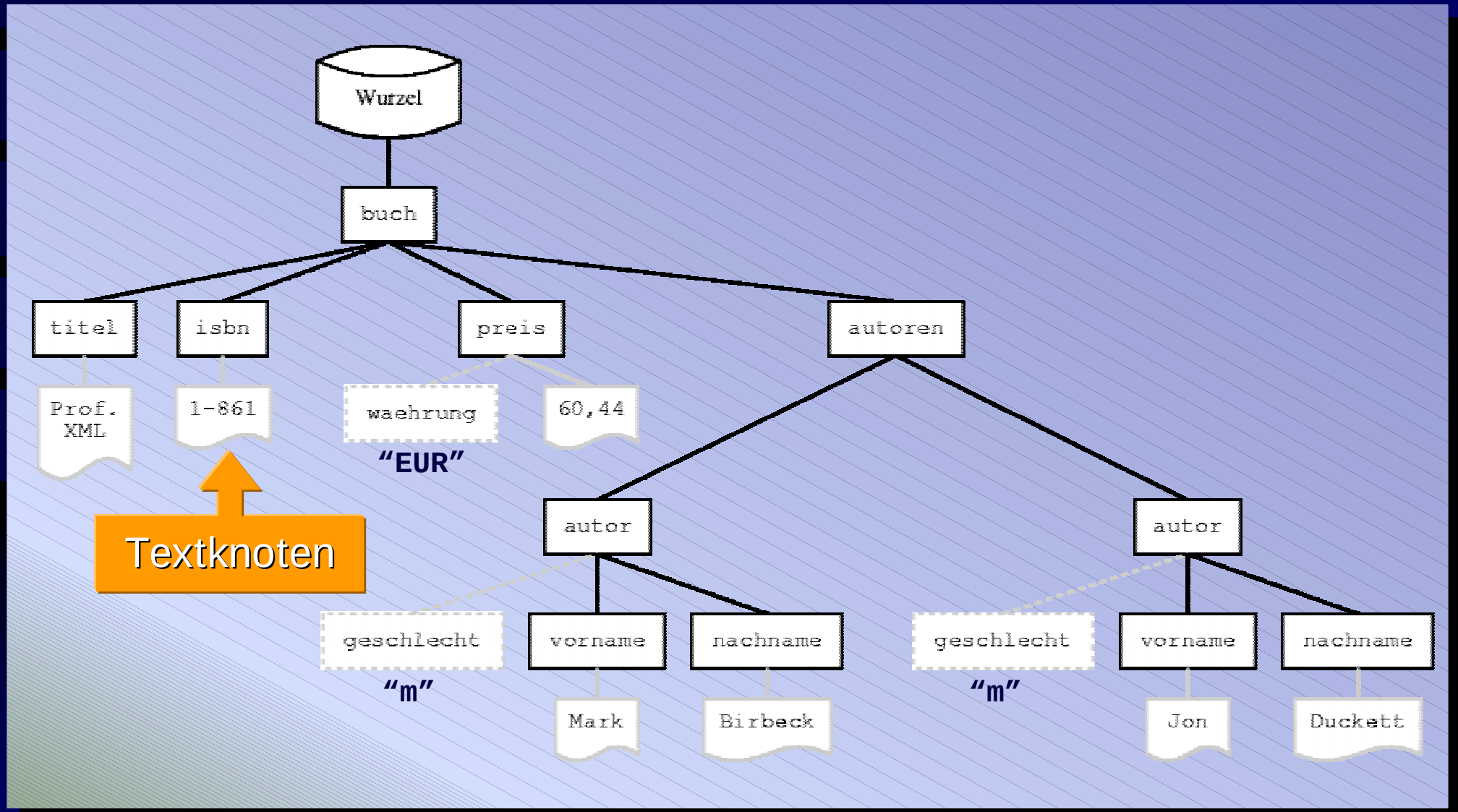
Baumdarstellung



Baumdarstellung



Baumdarstellung



Modellorientierte Ansätze

- Ausnutzung der hierarchischen Struktur von Dokumenten

- Adressierung von Werten über Pfad im Dokumentenbaum
- Unabhängig von DTD oder Schema

XRel: Überblick

- Charakterisierung eines Dokuments über Knoten des Baums
- Ablage eines Pfadausdrucks zusammen mit dem Wert zur Beschreibung eines Knotens
- Unterstützung von XPath-Anfragen

XRel: Pfadausdrücke

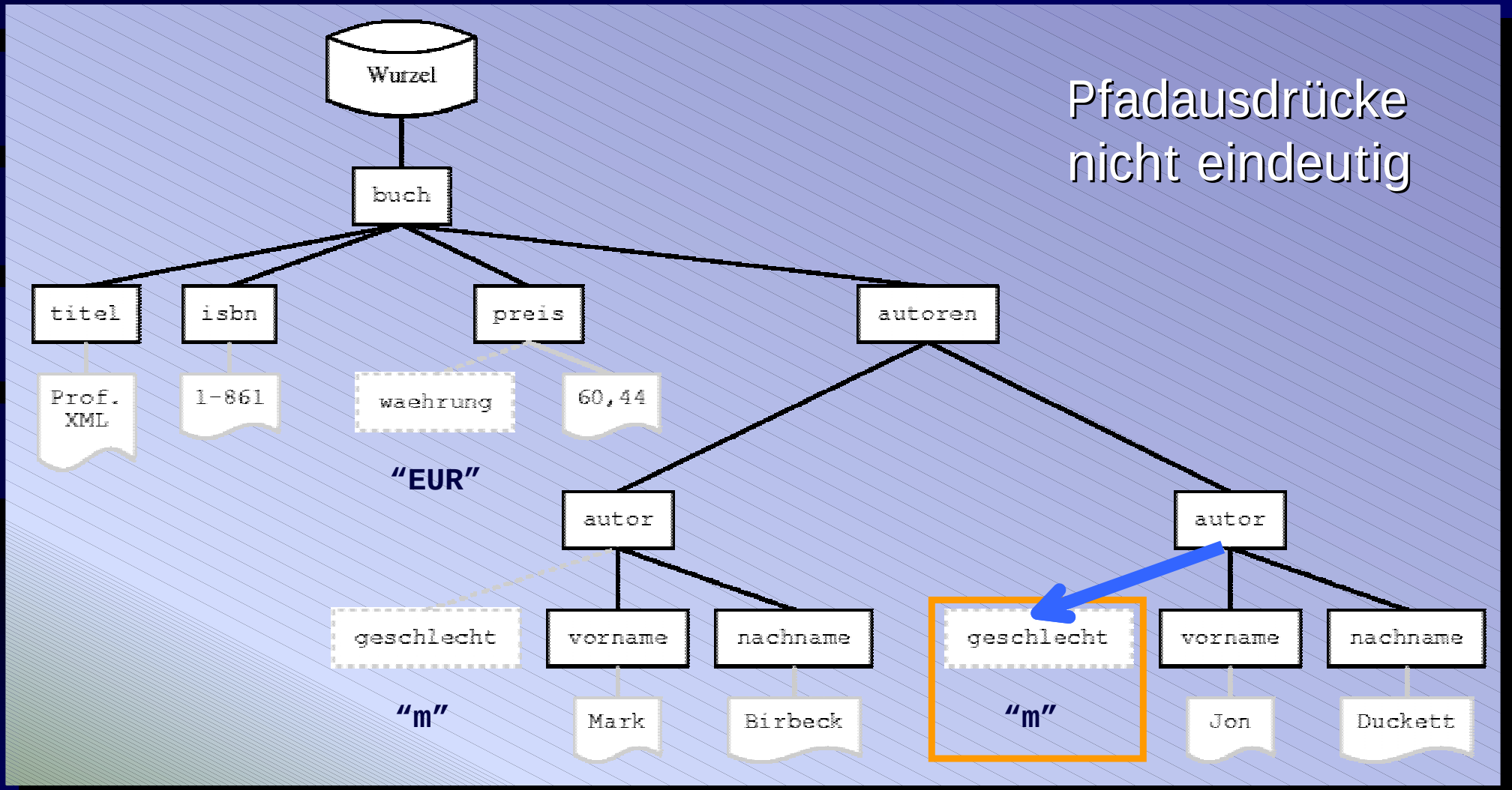
- Beschreibung der Position eines Knotens im Dokumentbaum
- Syntax an XPath angelehnt

XRel: Pfadausdrücke

- Beschreibung der Position eines Knotens im Dokumentbaum
- Syntax an XPath angelehnt

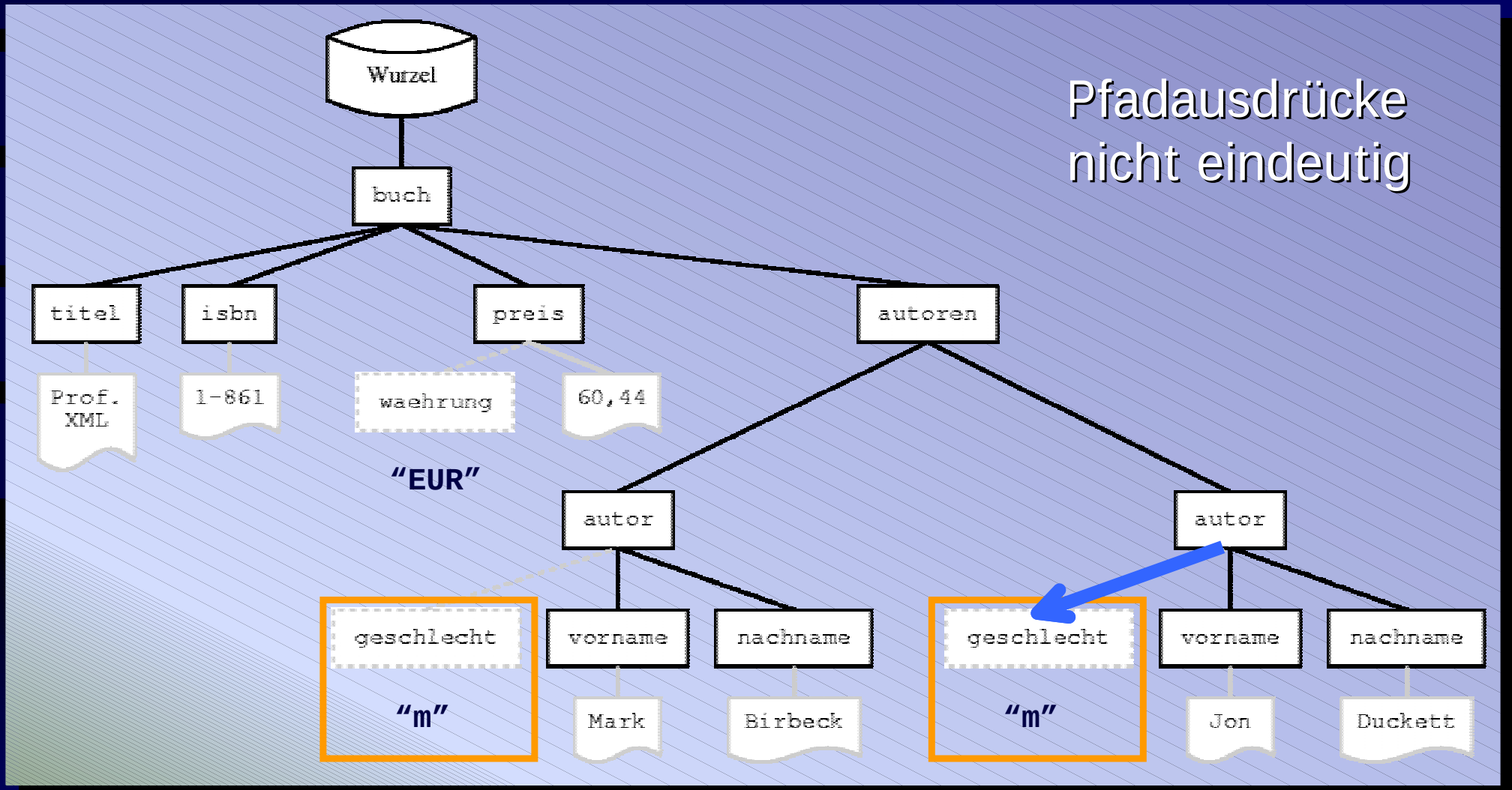
```
PfadAusdruck ::= '#/' Schritt
                 | PfadAusdruck '#/' Schritt

Schritt      ::= Name
                 | '@' Name

Name         ::= XMLName
```

XRel: Pfadausdrücke



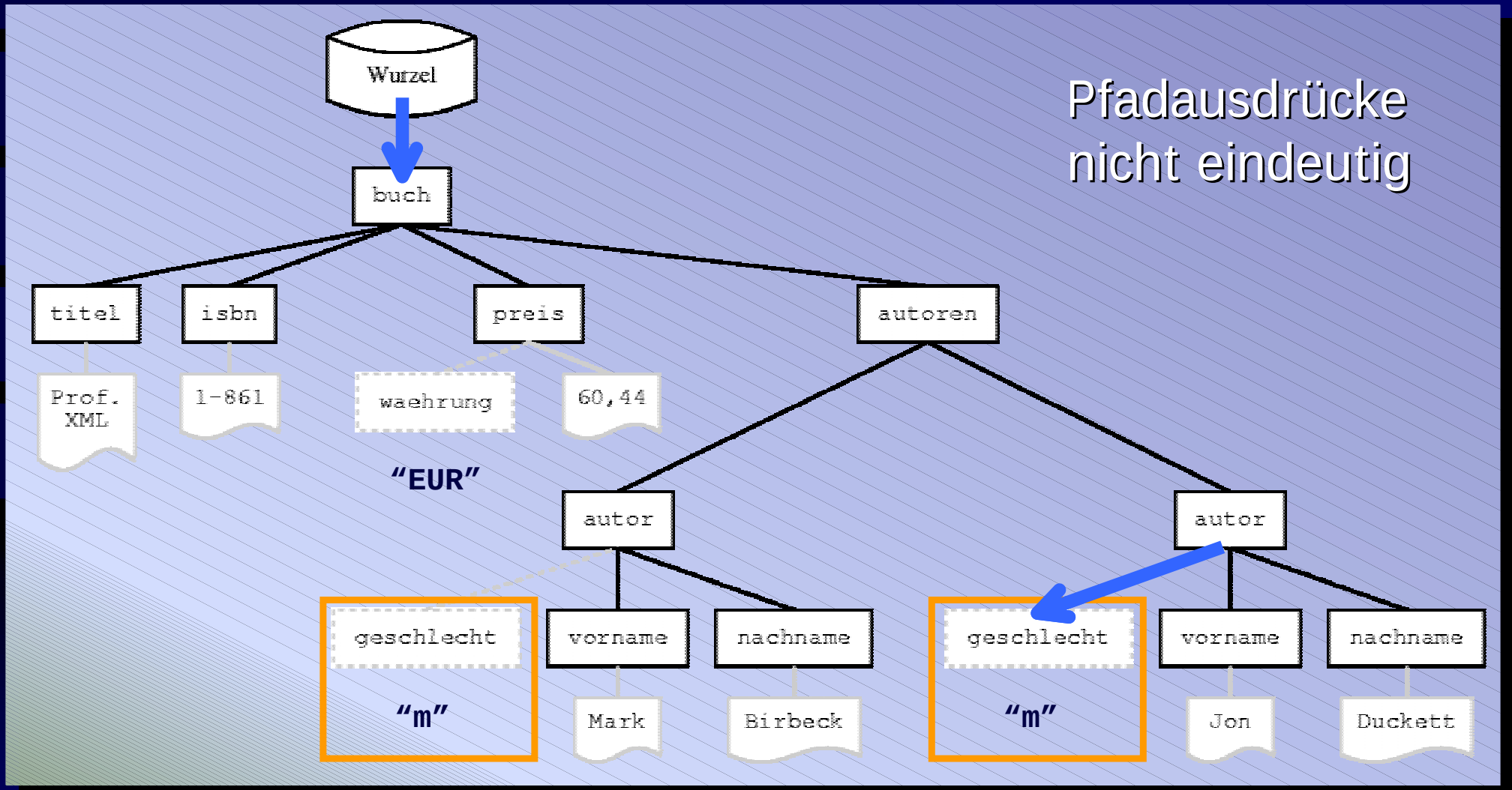
Beispiel: `#/buch#/autoren#/autor#/@geschlecht`

XRel: Pfadausdrücke



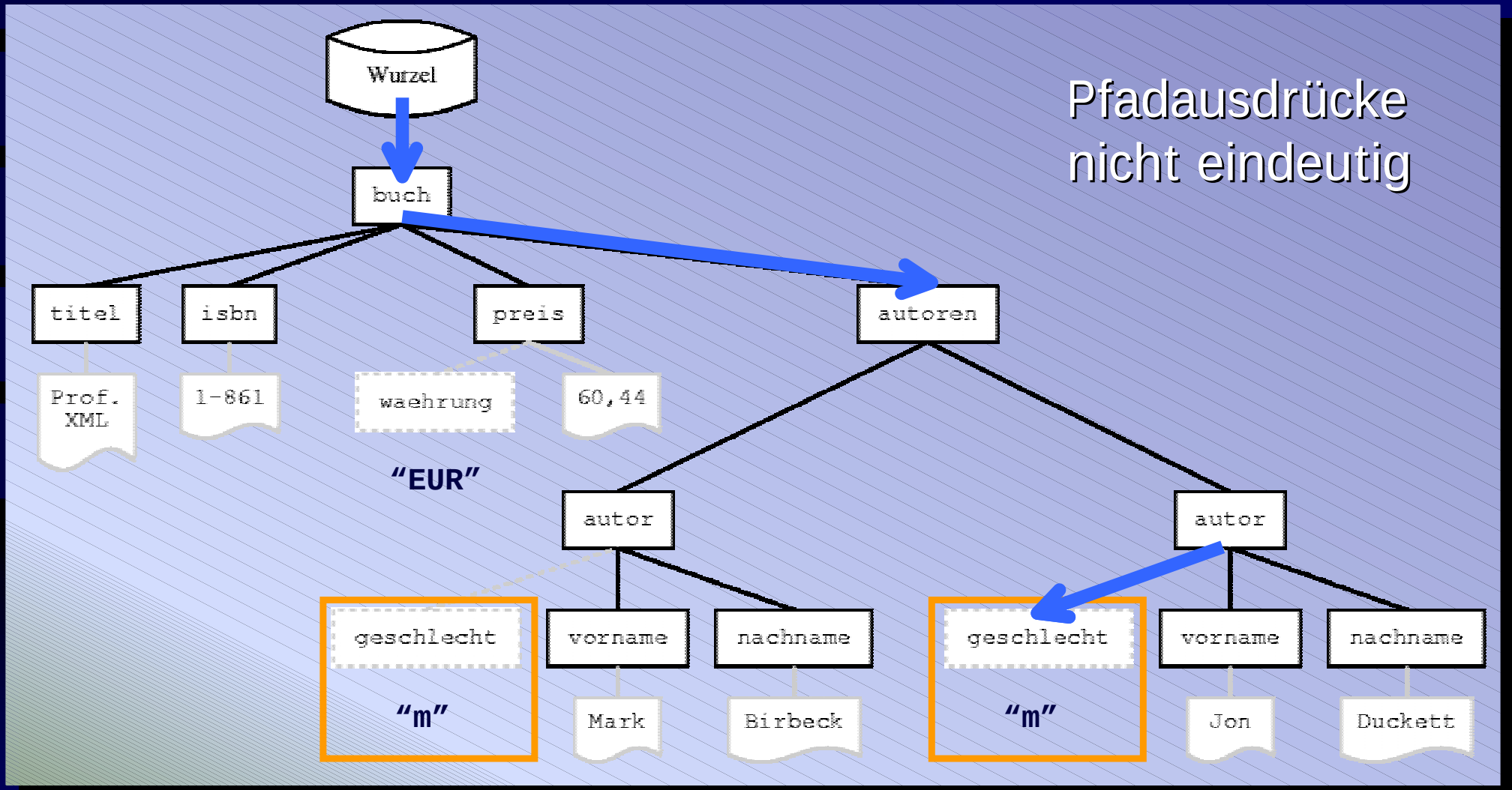
Beispiel: `#/buch#/autoren#/autor#/@geschlecht`

XRel: Pfadausdrücke



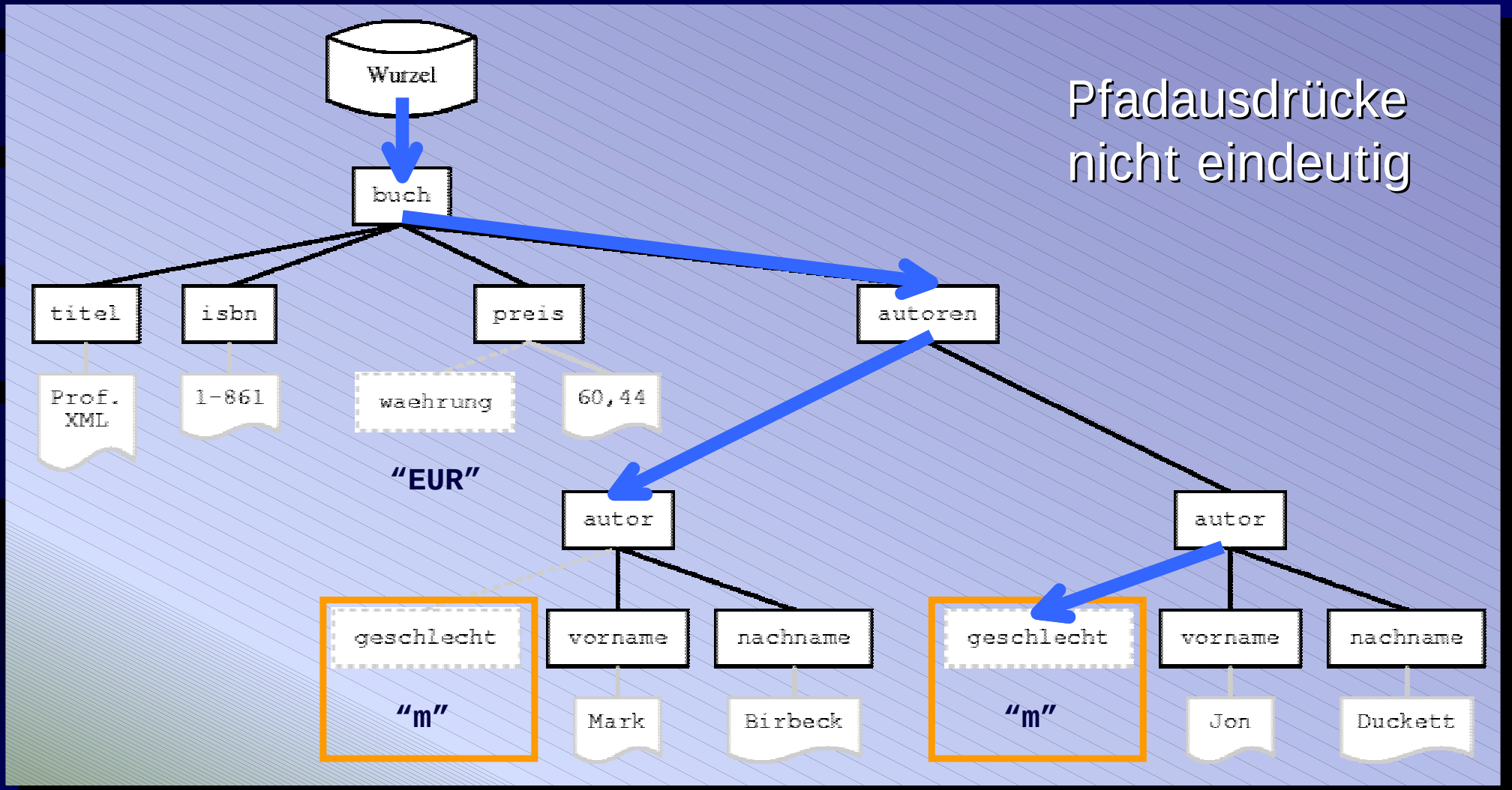
Beispiel: `#/buch#/autoren#/autor#/@geschlecht`

XRel: Pfadausdrücke



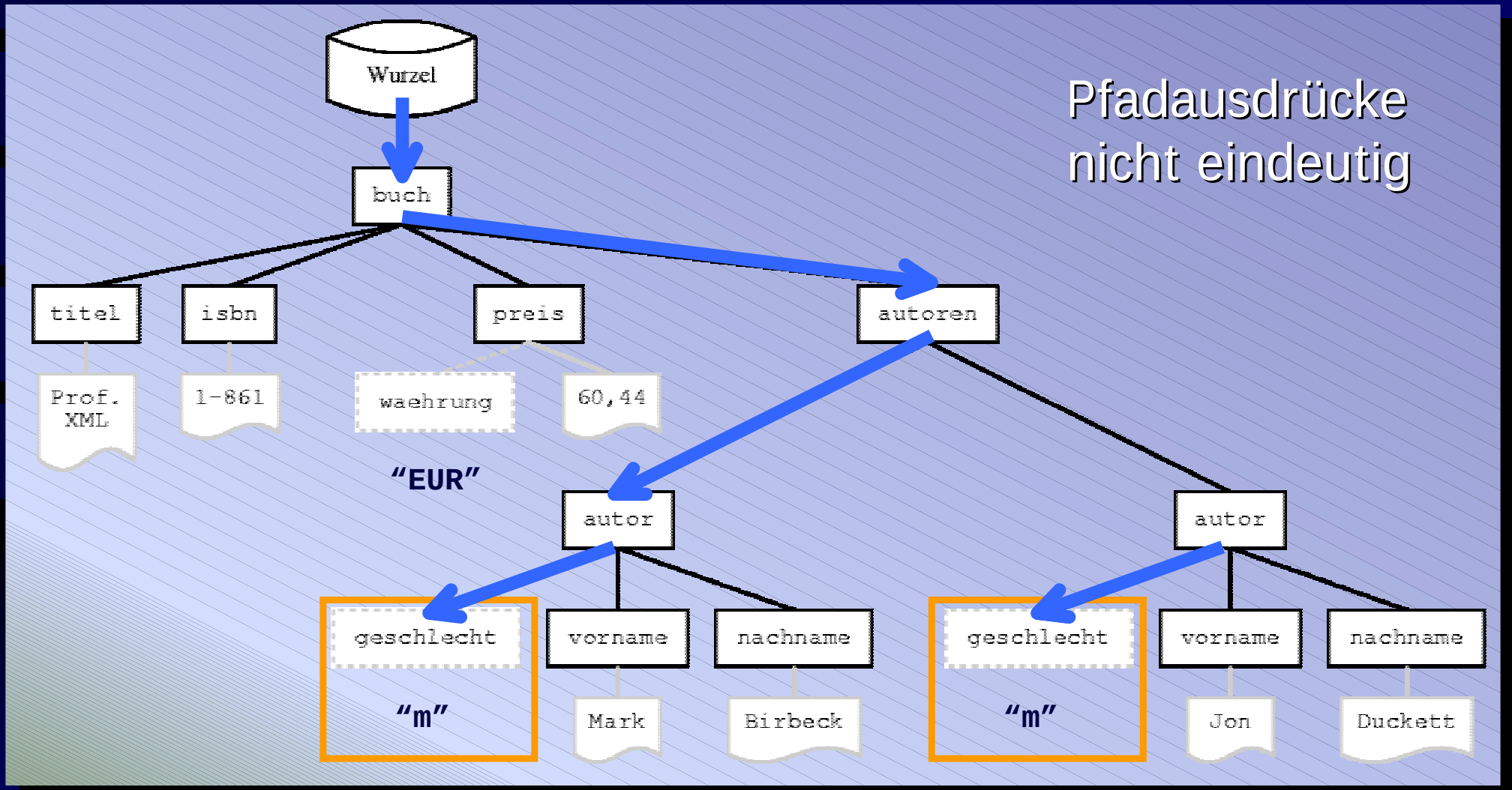
Beispiel: `#/buch#/autoren#/autor#/@geschlecht`

XRel: Pfadausdrücke



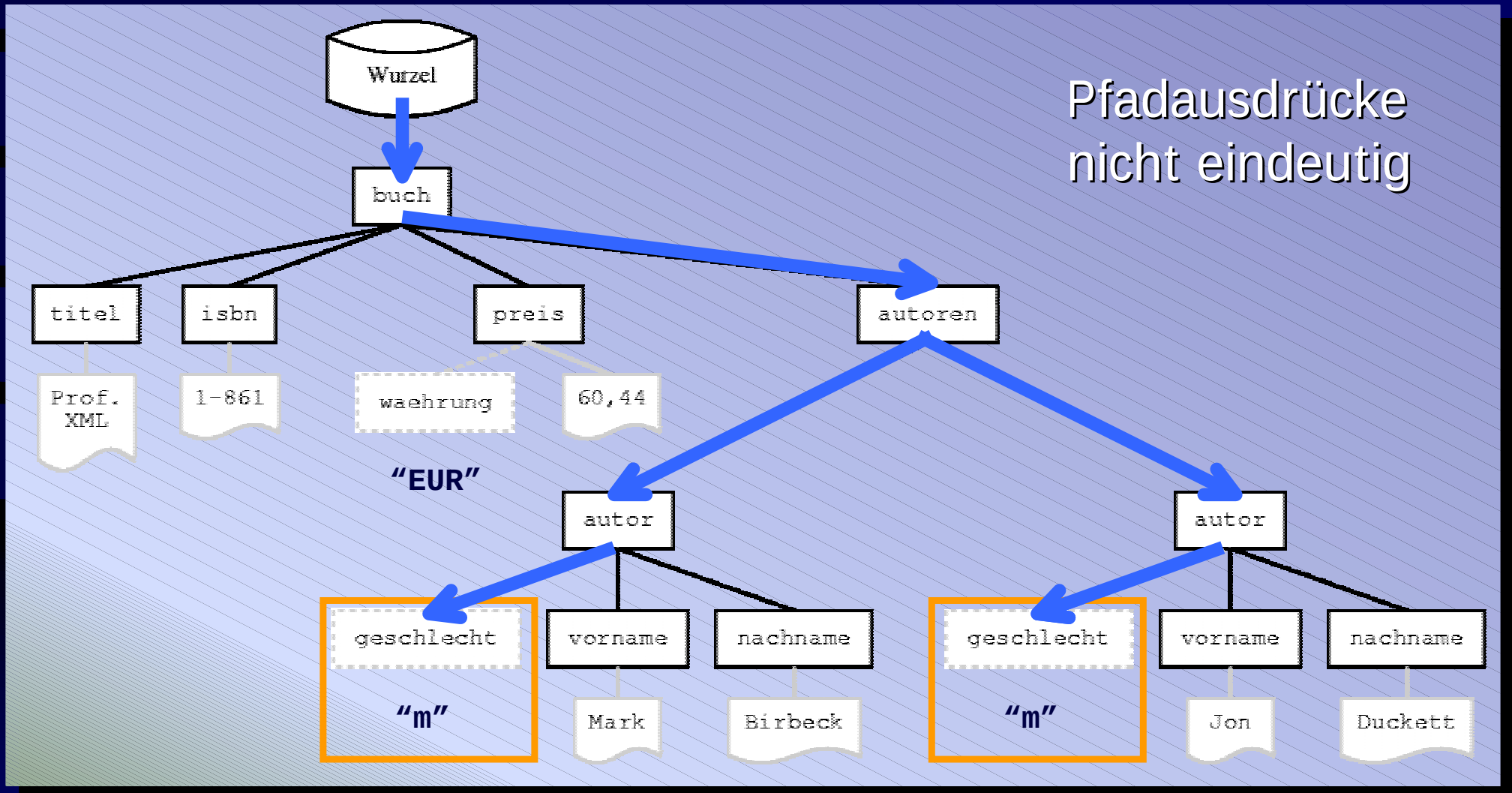
Beispiel: `#/buch#/autoren#/autor#/@geschlecht`

XRel: Pfadausdrücke



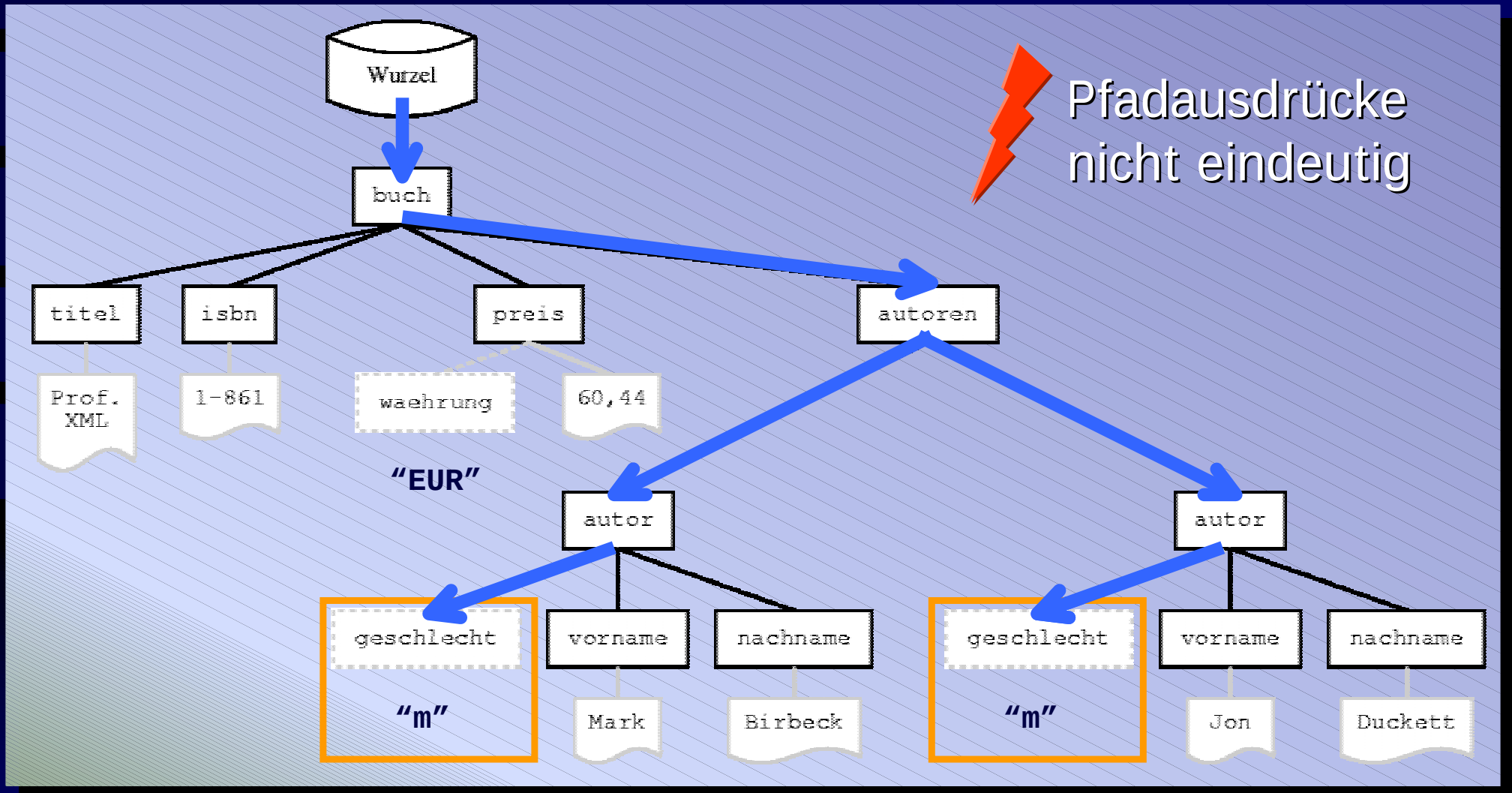
Beispiel: `#/buch#/autoren#/autor#/@geschlecht`

XRel: Pfadausdrücke



Beispiel: `#/buch#/autoren#/autor#/@geschlecht`

XRel: Pfadausdrücke



Beispiel: `#/buch#/autoren#/autor#/@geschlecht`

XRel: Datenbankschema (1)

- Eine relationale Tabelle pro Knotentyp
- Ein Tupel pro Knoten

XRel: Datenbankschema (1)

- Eine relationale Tabelle pro Knotentyp
- Ein Tupel pro Knoten

```
Element   (docID, path, start, end)
Attribute (docID, path, start, end, value)
Text      (docID, path, start, end, value)
```

XRel: Datenbankschema (1)

- Eine relationale Tabelle pro Knotentyp
- Ein Tupel pro Knoten

```
Element   (docID, path, start, end)
Attribute (docID, path, start, end, value)
Text      (docID, path, start, end, value)
```

➔ Pfadinformation redundant

XRel: Datenbankschema (2)

- Eine relationale Tabelle pro Knotentyp
- Ein Tupel pro Knoten

```
Element   (docID, pathID, start, end,
           index, reindex)
Attribute (docID, pathID, start, end, value)
Text      (docID, pathID, start, end, value)
Path      (pathID, pathexp)
```

XRel: Verfahren (1)

1. Erzeugen der Pfadtabelle

XRel: Verfahren (1)

1. Erzeugen der Pfadtabelle

pathID	pathexp
1	#/buch
2	#/buch#/titel
3	#/buch#/isbn
4	#/buch#/preis
5	#/buch#/preis#/@waehrung
6	#/buch#/autoren
7	#/buch#/autoren#/autor
8	#/buch#/autoren#/autor#/@geschlecht
9	#/buch#/autoren#/autor#/vorname
10	#/buch#/autoren#/autor#/nachname

XRel: Verfahren (2)

2. Erzeugen der Element-Tabelle

```
<buch>
   <titel>Professional XML</titel>
   <isbn>1-861005-05-9</isbn>
   <preis waehrung="EUR">60,44</preis>
   ...
</buch>
```

XRel: Verfahren (2)

2. Erzeugen der Element-Tabelle

```
<buch>
    <titel>Professional XML</titel>
    <isbn>1-861005-05-9</isbn>
    <preis waehrung="EUR">60,44</preis>
    ...
</buch>
```

docID	pathID	start	end	index	reindex

XRel: Verfahren (2)

2. Erzeugen der Element-Tabelle

```
<buch>
   <titel>Professional XML</titel>
   <isbn>1-861005-05-9</isbn>
   <preis waehrung="EUR">60,44</preis>
   ...
</buch>
```

docID	pathID	start	end	index	reindex

XRel: Verfahren (2)

2. Erzeugen der Element-Tabelle

```
<buch>
   <titel>Professional XML</titel>
   <isbn>1-861005-05-9</isbn>
   <preis waehrung="EUR">60,44</preis>
   ...
</buch>
```

docID	pathID	start	end	index	reindex
1	1	0	390	1	1

buch

XRel: Verfahren (2)

2. Erzeugen der Element-Tabelle

```
<buch>
   <titel>Professional XML</titel>
   <isbn>1-861005-05-9</isbn>
   <preis waehrung="EUR">60,44</preis>
   ...
</buch>
```

docID	pathID	start	end	index	reindex
1	1	0	390	1	1

buch

#/buch

XRel: Verfahren (2)

2. Erzeugen der Element-Tabelle

```
<buch>
    <titel>Professional XML</titel>
    <isbn>1-861005-05-9</isbn>
    <preis waehrung="EUR">60,44</preis>
    ...
</buch>
```

docID	pathID	start	end	index	reindex
1	1	0	390	1	1

buch

XRel: Verfahren (2)

2. Erzeugen der Element-Tabelle

```
<buch>
    <titel>Professional XML</titel>
    <isbn>1-861005-05-9</isbn>
    <preis waehrung="EUR">60,44</preis>
    ...
</buch>
```

docID	pathID	start	end	index	reindex	
1	1	0	390	1	1	*buch*
1	2	11	41	1	1	*titel*

XRel: Verfahren (2)

2. Erzeugen der Element-Tabelle

```
<buch>
    <titel>Professional XML</titel>
    <isbn>1-861005-05-9</isbn>
    <preis waehrung="EUR">60,44</preis>
    ...
</buch>
```

docID	pathID	start	end	index	reindex	
1	1	0	390	1	1	*buch*
1	2	11	41	1	1	*titel*

#/buch#/titel

XRel: Verfahren (2)

2. Erzeugen der Element-Tabelle

```
<buch>
   <titel>Professional XML</titel>
   <isbn>1-861005-05-9</isbn>
   <preis waehrung="EUR">60,44</preis>
   ...
</buch>
```

docID	pathID	start	end	index	reindex	
1	1	0	390	1	1	*buch*
1	2	11	41	1	1	*titel*

XRel: Verfahren (2)

2. Erzeugen der Element-Tabelle

```
<buch>
    <titel>Professional XML</titel>
    <isbn>1-861005-05-9</isbn>
    <preis waehrung="EUR">60,44</preis>
    ...
</buch>
```

docID	pathID	start	end	index	reindex	
1	1	0	390	1	1	*buch*
1	2	11	41	1	1	*titel*
1	3	47	72	1	1	*isbn*

XRel: Verfahren (2)

2. Erzeugen der Element-Tabelle

```
<buch>
    <titel>Professional XML</titel>
    <isbn>1-861005-05-9</isbn>
    <preis waehrung="EUR">60,44</preis>
    ...
</buch>
```

docID	pathID	start	end	index	reindex	
1				1	1	*buch*
1				1	1	*titel*
1	3	47	72	1	1	*isbn*

#/buch#/titel

XRel: Verfahren (2)

2. Erzeugen der Element-Tabelle

docID	pathID	start	end	index	reindex
1	1	0	390	1	1
1	2	11	41	1	1
1	3	47	72	1	1
1	4	78	111	1	1
1	6	117	381	1	1
1	7	134	246	1	2
1	9	168	190	1	1
1	10	203	230	1	1
1	7	255	366	2	1
1	9	289	310	1	1
1	10	323	350	1	1

XRel: Verfahren (3)

3. Erzeugen der Attribut-Tabelle

docID	pathID	start	end	value	
1	5	79	79	EUR	*waehrung*
1	8	134	134	m	*geschlecht*
1	8	255	255	m	*geschlecht*

4. Erzeugen der Text-Tabelle

docID	pathID	start	end	value
1	2	18	33	Professional XML
1	3	53	65	1-861005-05-9
1	4	99	103	60,44

XRel: Anfrageverarbeitung (1)

- Transformation von XPath-Ausdrücken in SQL-Anweisungen
  - Ersetzung von Zeichenketten:
    - `/` durch `#/`
    - `//` durch `#%/`
- Durchführung einer LIKE-Suche
- Darstellung des Ergebnisses in XML

XRel: Anfrageverarbeitung (2)

XRel: Anfrageverarbeitung (2)

- XPath-Ausdruck:

```
/buch/*/autor
```

XRel: Anfrageverarbeitung (2)

- XPath-Ausdruck:

  `/buch/*/autor`

- Ersetzungsergebnis:

  `#/buch#%/autor`

XRel: Anfrageverarbeitung (2)

- XPath-Ausdruck:

  /buch/*/autor

- Ersetzungsergebnis:

  #/buch#%/autor

- SQL-Anfrage:

```
SELECT    e.docID, e.start, e.end
FROM      Element e, Path p
WHERE     e.pathID = p.pathID
AND       p.pathexp LIKE '#/buch#%/autor'
ORDER BY e.docID, e.start, e.end
```

XRel: Bewertung

- \+ Speicherung unterschiedlich strukturierter Dokumente
- \+ XPath-Unterstützung
  - \+ einfach ausführbar
  - – String-Vergleiche langsam
  - – schlecht durch Indexstrukturen stützbar
- – Ineffizient bei gleichartigen Dokumenten

Strukturorientierte Ansätze

- Nachbildung der logischen Dokumentstruktur
  - Baumstruktur nur implizit vorhanden
- Spezialisierung auf bestimmte Klasse von Dokumenten
  - gestützt auf DTD oder XML Schema

XDatabase: Überblick

- Analyse eines XML-Schemas
- Erzeugung eines relationalen Datenbankschemas
  - Eine Relation pro Elementtyp
  - Felder für Attribute und Attributgruppen
  - Fremdschlüsselbeziehungen für Hierarchie

```
buch     (id)
autoren  (id, parentBuchId)
autor    (id, parentAutorenId, geschlecht)
```

XDatabase: Typen

- Darstellung einfacher Typen
  - Verwendung von Constraints

```
CONSTRAINT CHECK (geschlecht IN
                  (‘m’,‘w’))
```

- Darstellung komplexer Typen:
  - Einführen separater Tabellen
  - Referenz über Fremdschlüssel

XDatabase: Beispiel

- Erzeugte Tabellenstruktur:

```
buch     (id)
isbn     (id, parentBuchId, isbn)
preis    (id, parentBuchId, waehrung, preis)
autoren  (id, parentBuchId)
autor    (id, parentAutorenId, geschlecht)
vorname  (id, parentAutorId, vorname)
nachname(id, parentAutorId, nachname)
```

XDatabase: Bewertung

+ Abbildung von logischen Zusammenhängen
+ Robustheit durch starke Typisierung

– Anfrageverarbeitung sehr aufwändig
– nur Dokumente einer Klasse ablegbar

LegoDB: Überblick

- Ansatz zur automatischen Bestimmung optimierter Abbildungen
- Nutzung von Wissen über Anwendung
  - Datenstruktur
  - Typische Benutzerabläufe
  - Datenverteilung

LegoDB: Grundlagen

- Äquivalenzbegriff über XML Schema

> Zwei XML Schemata heißen genau dann äquivalent, wenn die Mengen gültiger Dokumente, die von beiden Schemata beschrieben werden, identisch sind.

LegoDB: Grundlagen

- Äquivalenzbegriff über XML Schema

> Zwei XML Schemata heißen genau dann äquivalent, wenn die Mengen gültiger Dokumente, die von beiden Schemata beschrieben werden, identisch sind.

- Äquivalenzumformungen
  - Reguläre Ausdrücke
  - Typdefinitionen

LegoDB: Eingabe

- Wissen über Anwendung
  - XML Schema
  - XQuery Lastprofil
  - Statistiken über Datenverteilung

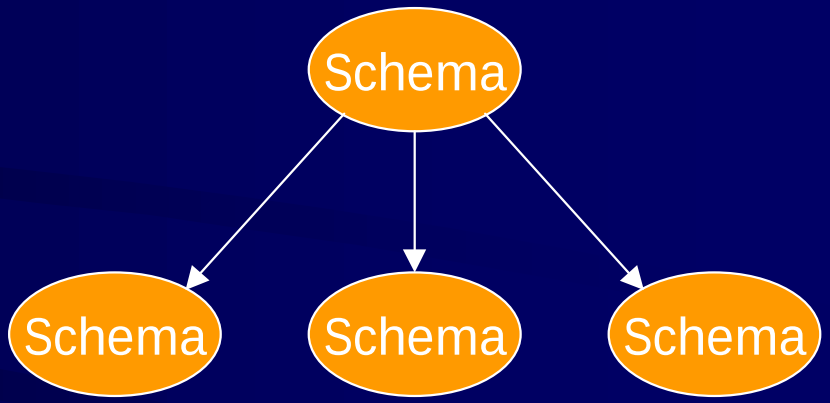
LegoDB: Verfahren (2)

- Äquivalenz-umformungen auf XML Schemata durchführen

- Suchraum aufspannen

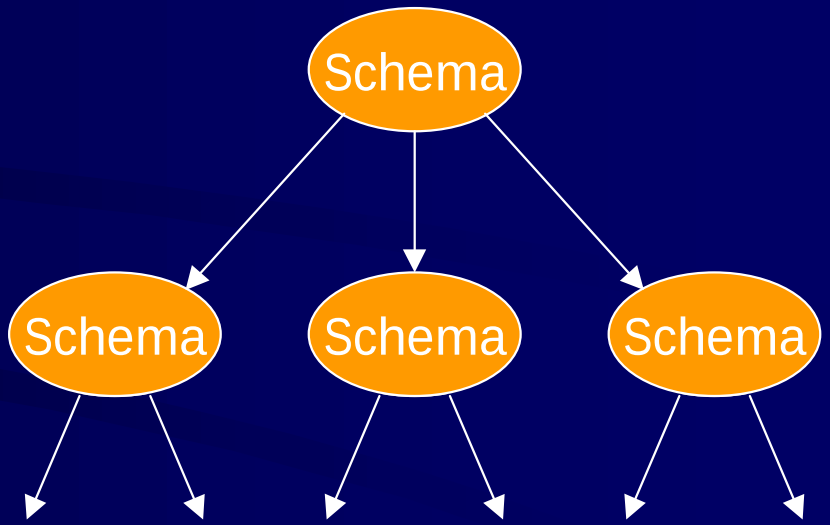
LegoDB: Verfahren (2)

- Äquivalenz-umformungen auf XML Schemata durchführen
- Suchraum aufspannen



LegoDB: Verfahren (2)

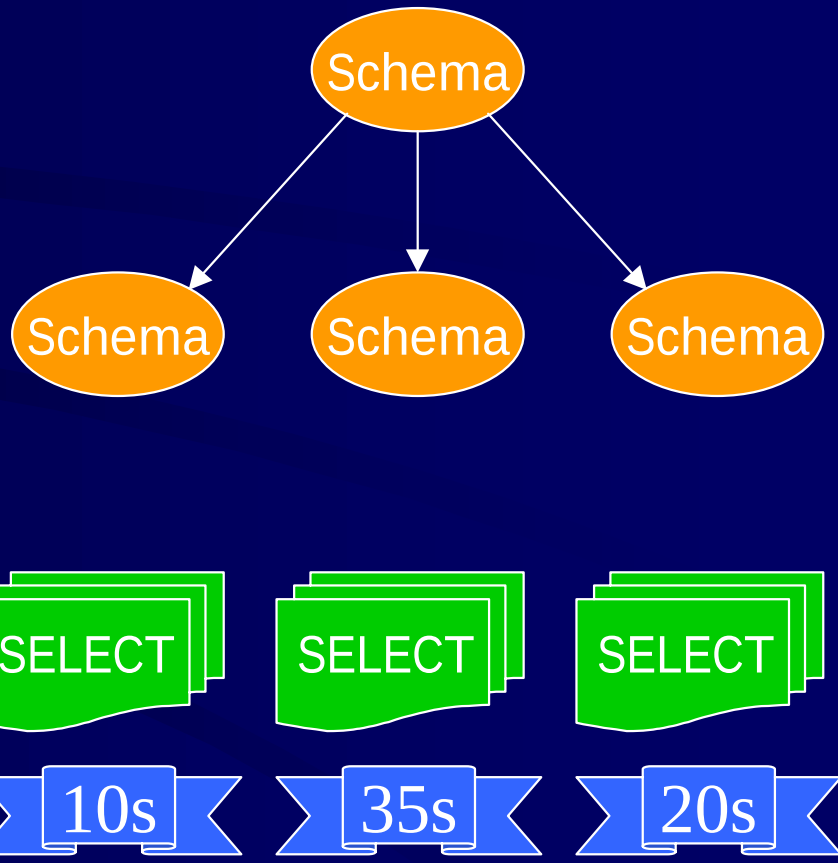
- Äquivalenz-umformungen auf XML Schemata durchführen
- Suchraum aufspannen



LegoDB: Verfahren (2)

Für jeden Knoten:

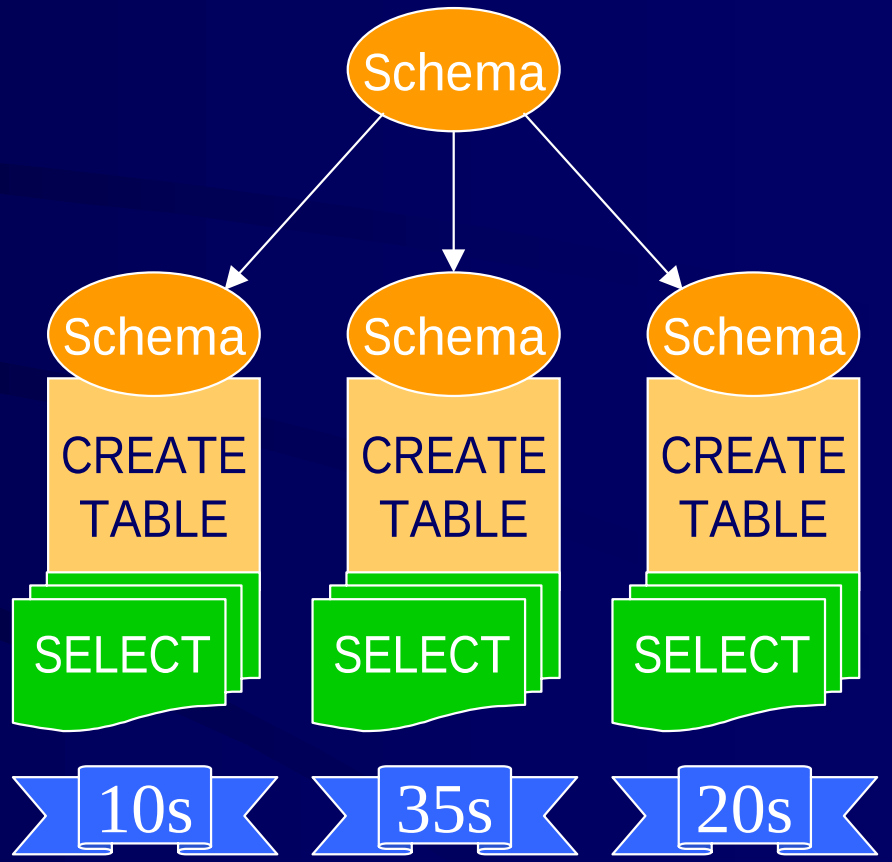
- Relationales Schema erzeugen



LegoDB: Verfahren (2)

Für jeden Knoten:

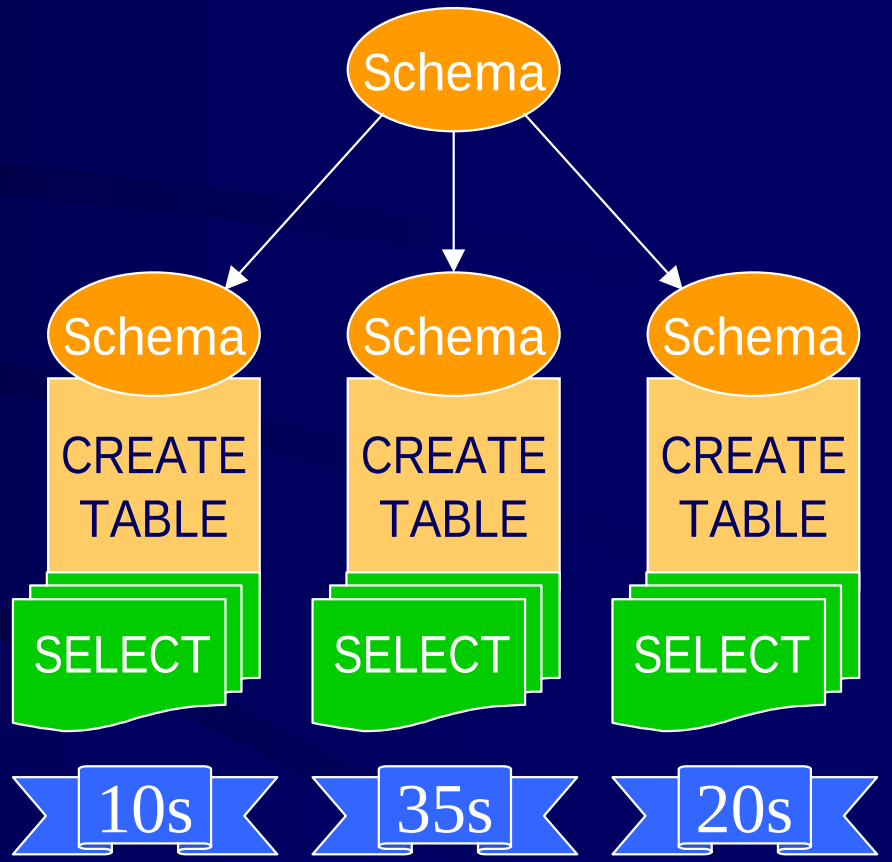
- Relationales Schema erzeugen



LegoDB: Verfahren (2)

Für jeden Knoten:

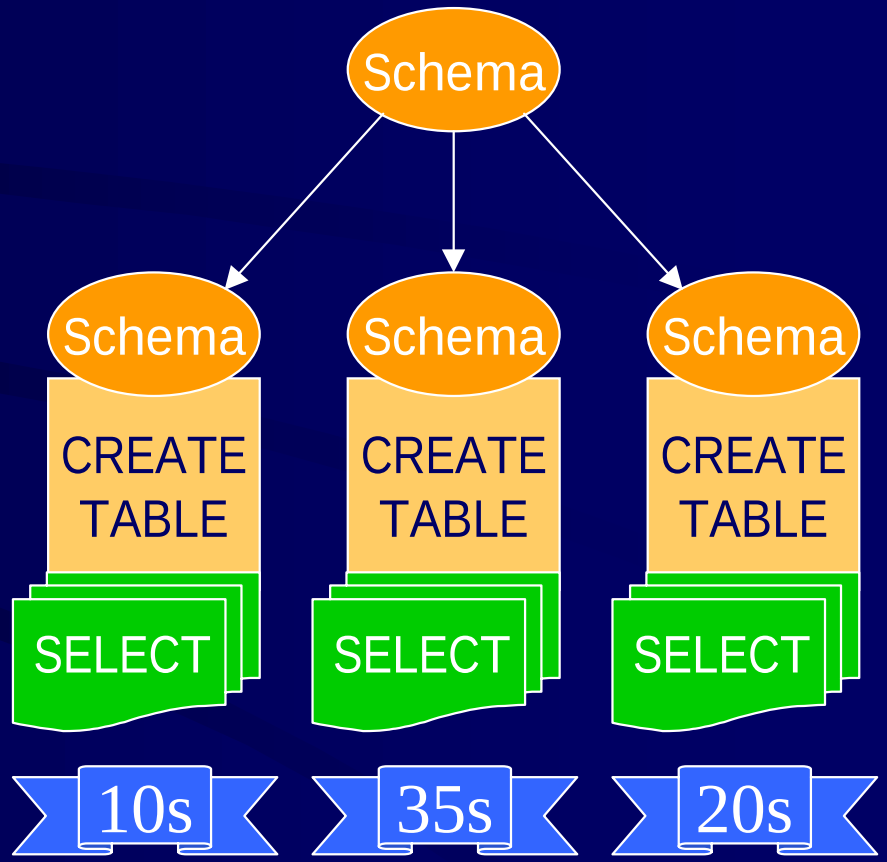
- Relationales Schema erzeugen
- XQuery-Lastprofil nach SQL umsetzen



LegoDB: Verfahren (2)

Für jeden Knoten:

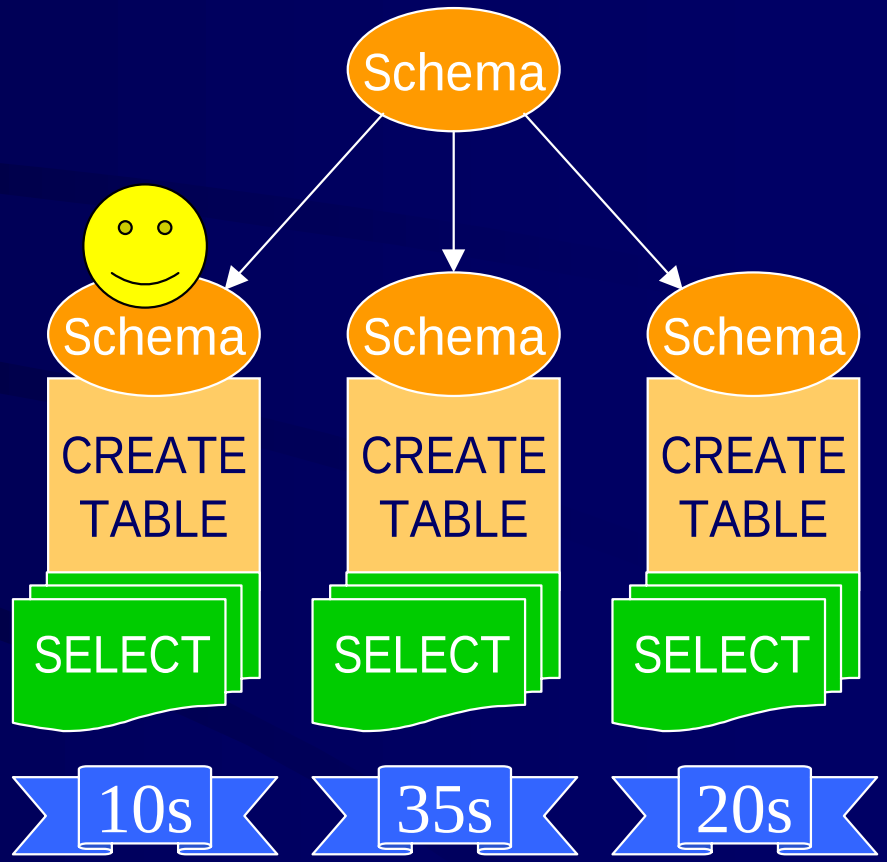
- Relationales Schema erzeugen
- XQuery-Lastprofil nach SQL umsetzen
- Abbildung mit relationalem Optimizer bewerten



LegoDB: Verfahren (2)

Für jeden Knoten:

- Relationales Schema erzeugen
- XQuery-Lastprofil nach SQL umsetzen
- Abbildung mit relationalem Optimizer bewerten



LegoDB: Bewertung

+ Leistungsfähige Abbildungen erzeugbar
– Bindung an Dokumentklasse
– Aufwändige Anfrageverarbeitung

Zusammenfassung

- Datenbanksysteme zur Speicherung von XML-Daten geeignet
- Verfügbarkeit unterschiedlicher Ansätze
  - modellorientiert, strukturorientiert
- Werkzeuge zum Vergleich von Ansätzen
- Werkzeuge zur Unterstützung von Anwendungsentwicklern

Speicherung von XML in (objekt-)relationalen Datenbanken

Burkhard Schäfer