



XML-Speicherung in Datenbanksystemen

Dr. Harald Schöning



Überblick

- **Motivation: XML und e-Commerce**
- **XML in relationalen Datenbanken**
- **Das XML-Datenbanksystem Tamino**

XML

- Ist eine Metasprache
- entstanden aus SGML
- *die* neue Sprache für das Web
- ist recht einfach aufgebaut
- `<?XML version="1.0"?>`
 - `<Vortrag Uhrzeit="17:15 Uhr">`
 - `<Thema>XML-Speicherung in Datenbanken</Thema>`
 - `<Vortragender>Dr. Harald Schöning`
 - `<Firma>Software AG</Firma>`
 - `</Vortragender>`
 - `</Vortrag>`

Wichtige Eigenschaften von XML

- Dokumente haben eine Baumstruktur, einen optionalen Prolog und ggf. eine DTD
- Die wichtigsten Bausteine sind *Elemente* und *Attribute*
- Dokumente sind wohlgeformt (*well-formed*), wenn sie den Syntaxregeln genügen (im wesentlichen, wenn sie die geforderte Baumstruktur haben)
- Dokumente *können* ein Schema haben. Wenn sie wohlgeformt sind und dem Schema entsprechen, nennt man sie gültig (*valid*)
- Kommentare und Processing Instructions sind an jeder Stelle erlaubt

Wichtige Eigenschaften von XML - 2

- Elemente können auch rekursiv geschachtelt sein
- Elemente gleichen Namens dürfen mehrfach vorkommen
- Reihenfolge und Position sind wichtig
- `<Satz><Subjekt>Mixed content</Subjekt>` ist möglich `</Satz>`
- Elemente können leer sein
- Attribute können optional sein
- Bei Attributen ist Reihenfolge unwichtig
- Es gibt spezielle Attribute zur Referenzierung (ID und IDREF)

Datenaustausch - Schlüssel zum e-Commerce

- **Company to company (supply chain)**

- **EDI**

- Punkt-zu-Punkt Lösung
- hoher Investitionsaufwand
 - für Mittelstand ungeeignet

- **Ablösung durch XML**

- Standardisierte DTDs (OASIS)
- Können auch unilateral erweitert werden
- Standardwerkzeuge

Flexible Formatierung

- XML impliziert keine bestimmte Präsentationsform
- Unterschiedliche Präsentationen von XML-Dokumenten wird durch Style-Sheets möglich
- Hohe Flexibilität
- Generischer Ansatz

Andere Anwendungen

- **Dokumentenverwaltung / Content Management**
- **Website-Management**
- **electronic mall**
- **intelligente Suchmaschinen**
- **Katalogverwaltung**
- **Knowledge management**
- **e-government**
- **mobile Anwendungen**
- **Metadatenverwaltung**
- **Multimediatdaten**
- **...**

Warum XML in Datenbanken?

- **In diesen Anwendungen besteht der Bedarf der Speicherung**
 - effizient
 - sicher
 - mit Abfragemöglichkeiten
 - Mehrbenutzerbetrieb
 - ...
- **Aber auch Anbindung an vorhandene DBS!**

Daten oder Dokumente?

■ Dokumente

- sind selten genau gleich strukturiert
- Reihenfolge ist wichtig
- Volltextsuche ist unabdingbar
- “mixed content” ist typisch
- Beispiele:
 - Zeitschriftenbeiträge
 - Verträge
 - Bücher

Daten oder Dokumente? (2)

- **Daten (z.B. in relationalen Systemen)**

- haben oft keine feste Ordnung
- sind einheitlich und meist einfach strukturiert
- haben Datentypen
- Beispiele:
 - Flugpläne
 - Bestellungen

- **XML kann beides repräsentieren! Und Mischformen!**

- Beispiel:
 - Produktkataloge

Anfragetypen

■ Wertorientiert:

- @attribute > 5
 - <element attribute="4" />
- element < 7.1
 - <element>6.1</element>
- element = "Hugo"
 - <element>Hugo</element>
 - <element><subelement>Hugo</subelement></element>
 - <element>Hugo</element>
- element > "Otto"

Anfragetypen (2)

■ Textorientiert:

- documents containing „XML“
- documents containing "XML" OR "HTML" but not "SGML"
- documents containing "XML" within two words of "database"
- documents with words similar to "XML" (ranking)

Anfragetypen (3)

■ Strukturorientiert:

- //Buch

Dokumente, in denen ein Element "Buch" vorkommt

- //Buch/@ISBN

Dokumente, in denen ein Element "Buch" ein Attribut "ISBN" hat

- /Buch/Titel

- /Buch/Autor[1]

- Titel AFTER /Buch/Autor

Anfragetypen (4)

■ Linkorientiert

- Dokumente, die auf eine bestimmte Stelle zeigen
- Dokumente, die aus einem Dokument referenziert werden

■ Kombinationen aus

- wertorientiert
- textorientiert
- strukturorientiert
- linkorientiert
 - //Buch[Preis < 50 AND Titel CONTAINS "XML"]

XML in Datenbanken - Optionen zur Realisierung

■ relational

- inhaltsorientiert zerlegt

 - generisch

 - definatorisch

- strukturorientiert zerlegt

- opak

■ objektorientiert

■ XML (Tamino)

Relational inhaltsorientiert

```
<Lieferung Datum="7.6.2001">  
  <Lieferadresse>  
    <Name>Harald Schöning</Name>  
    <Stadt>Dieburg</Stadt>  
  </Lieferadresse>  
  <Teil Nummer="4711" Menge="10"/>  
  <Teil Nummer="4712" Menge="11"/>  
</Lieferung>
```

Lieferung	Teilnr	Menge
17	4711	10
17	4712	11

Lieferung

Nr	Datum	Kunde
17	7.6.2001	K19

Kunden

Kundennr	Name	Ort
K19	Harald Schöning	Dieburg

Relational inhaltsorientiert - generisch

■ Darstellung von relationalen Tabellen als XML-Dokumente

■ Columns als Elemente oder Attribute

■ <employees>

```
<employee> <emplno>17</emplno>
```

```
<name>Egon Maier</name> </employee>
```

```
<employee> <emplno>18</emplno>
```

```
<name>Hugo Maier</name> </employee>
```

```
<employees>
```

■ <employees>

```
<employee emplno=„17“ name=„Egon Maier“/>
```

```
<employee emplno=„18“ name=„Hugo Maier“/>
```

```
<employees>
```

Relational inhaltsorientiert - generisch (2)

- **Konvertierung einer Tabelle (d.h. jedes SQL-Anfrageergebnisses) nach XML ist einfach**
- **kann außerhalb der Datenbank geschehen**
 - z.B. Bluestone's XML Suite (generierte Java-Klassen)
- **Umgekehrter Weg:**
 - Abspeichern von XML-Dokumenten in vorgegebene Tabellen
 - Setzt eine veränderbare Sicht oder Basistabelle voraus
 - Die XML-Dokumente müssen in der Struktur der Tabelle entsprechen

Relational inhaltsorientiert - generisch (3)

- **Dokumentstruktur starr,**
- **Rekursion, mixed content nicht definierbar**
- **Schema durch relationales Schema gegeben**
- **“round trip” möglich?**
 - Kommentare, processing instructions?
 - XML-Prolog
 - Reihenfolge der Elemente
 - Element vs. Attribut
- **offensichtlich keine beliebigen (nicht vordefinierten) XML-Dokumente speicherbar**

Relational inhaltsorientiert - definitorisch

- **Definition einer Abbildungsvorschrift zwischen XML-Dokument und Datenbanktabellen**
- **Elemente und Attribute können auf Zeilen / Spalten verschiedener Tabellen abgebildet werden**
- **Abbildungsvorschrift wird vor dem ersten Abspeichern erstellt und in der Datenbank gespeichert**

Relational inhaltsorientiert - definatorisch (2)

```
<Xcolumn>
  <table name="Person_names">
    <column name="fname" type="varchar(50)"
      path = "/person/firstName" multi_occurrence="NO"/>
    <column name="lname" type="varchar(50)"
      path = "/person/lastName" multi_occurrence="NO"/>
  </table>
  <table name="person_phone_number">
    <column name="pnumber" type="varchar(20)"
      path="/person/phone/number" multi_occurrence="YES"/>
  </table>
</Xcolumn>
```

Relational inhaltsorientiert - definatorisch (3)

- Nur Dokumente mit a priori bekanntem Schema
- Rekursion?
- Reihenfolgeerhaltung?
- Prolog, Kommentare, processing instructions gehen verloren
- kein mixed content
- Anfragesprache ist SQL
- wertebasierte Anfrage ist einfach, textorientierte nur bedingt, strukturorientierte schwierig
- Konvertierung SQL-Ergebnis → XML notwendig

Relational inhaltsorientiert - definitorisch (4)

- **Nur Dokumente mit a priori bekanntem Schema**
- **Unterschiedliche Schemamächtigkeit**
 - Rekursion?
 - kein mixed content
- **Keine vollständige Abbildung von Dokumenten**
 - Reihenfolgeerhaltung?
 - Prolog, Kommentare, processing instructions gehen verloren
- **Anfragesprache ist SQL**
 - Anfragen aus XML-Sicht müssen immer in SQL übersetzt werden
 - Werkzeuge?

Relational inhaltsorientiert - definatorisch (5)

■ Anderes Datenmodell

- Konvertierung SQL-Ergebnis → XML notwendig
- Auflösen synonyme Darstellungen notwendig:
'' ' '
- beim Speichern andere Validierungsanforderungen:
 - was als DATE deklariert ist, muß dem auch syntaktisch entsprechen
- Konvertierung wegen Datentypen kann round-trip verhindern
 - DATE Format

■ wertebasierte Anfrage ist einfach, textorientierte nur bedingt, strukturorientierte schwierig

Beliebiges XML in (objekt-)relationalen DB

- **Ziel: Abspeichern beliebig strukturierter XML-Dokumente in relationalen Datenbanken**
 - Elemente **und** Attribute
 - multiple Elemente (z.B. mehrere Vornamen)
 - rekursive Struktur
 - reihenfolgeerhaltend
 - mixed content
 - Kommentare und processing instructions
 - XML Prolog

Strukturorientiert zerlegt - Beispiel

```

    2
1 <Lieferung Datum="7.6.2001">
  3 <Lieferadresse>
    4 <Name>
      Harald Schöning
    </Name>
    5 <Stadt>
      Dieburg
    </Stadt>
  6 <Teil Nummer="4711" 7
    8 Menge="10" />
  9 <Teil Nummer="4712" 10
    11 Menge="11" />
</Lieferung>

```

Quelle	Nr	Tag	Ziel	Daten
0	1	Lieferung	1	
1	0	@Datum	0	7.6.2001
1	1	Lieferadresse	3	
3	1	Name	4	Harald Schöning
3	2	Stadt	5	Dieburg
1	2	Teil	6	
6	0	@Nummer	0	4711
6	0	@Menge	0	10
1	3	Teil	9	
9	0	@Nummer	0	4712
9	0	@Menge	0	11

Relational strukturorientiert zerlegt

- z.B. Xbase oder Florescu & Kossmann:
- Speicherung in generischen Tabellen
 - Abspeichern der Kanten des zum XML-Dokument gehörigen Strukturbaumes
- Aus 1 XML-Dokument werden n Sätze
- Kommentare, mixed content?
- Verwendete relationale Strukturen für Benutzer unbrauchbar

Relational strukturorientiert zerlegt (2)

- **Anfragesprache: nur SQL**
 - keine adäquaten Anfragekonstrukte
- **Schlechte Performance**
 - Anfragen beinhalten komplexe Joins
 - Anfragen enthalten Sortierung (wegen der Reihenfolgeerhaltung)
 - Locking
 - viele Tupel pro Dokument \Rightarrow viele Sperren pro Änderung

Relational opak

- XML-Dokument als Inhalt einer Spalte einer Tabelle (BLOB, CLOB) oder als externe Datei

```
CREATE TABLE xmlTable (  
                xml      BLOB)
```

- schematische Beschreibung des XML-Dokumentes nicht Voraussetzung
- Anfragen mittels Extenders/Data Blades
- Zugriffspfade / Indizes über Datenbankerweiterungen?
- Locking auf Dokumentebene
 - feineres Locking unmöglich

Anfragen über Extensions

■ Anfragen über

- UDF, die eine eigene Anfragesprache verstehen können

```
SELECT xml
```

```
FROM xmlTable
```

```
WHERE QUERY(xml, “/uni/fachbereich[@name=“Informatik”]”)
```

■ Optimierung von Anfragen über mehrere Dokumente?

■ Sortierung des Ergebnisses?

■ UPDATE xmlTable

```
SET xml = TransformXml(xml, “replace <a> by <b>”)
```

- ersetzt den kompletten Attributwert, d.h. das gesamte Dokument!

Anfragen über Extensions

■ Behandlung der Daten-Sicht

- Beispiel: `<document importance="5"> ...`
- *Finde alle Dokumente, deren Wichtigkeit mindestens 3 ist*
- Text-Retrieval-Funktionen helfen nicht weiter
- Kann es Zugriffsstrukturen (Indizes) dafür geben?
- Voraussetzung ist Erkennung als numerisches Feld
- DB2: side tables
 - Konsistenz nicht garantiert!
 - Auch syntaktisch getrennt
 - ggf. nested tables (multiplicity > 1)

Anfragen über Extensions

■ Textsuche

- separate Software? (so bei DB2)
- Speicherung des XML-Dokumentes muss entsprechendes Format haben
 - character entities
 - encoding
- Kombination aus struktureller Suche und Textsuche?
 - Z.B. “Alle Dokumente, in denen ein Element *Firma* vorkommt, das die Zeichenfolge “AG” im Inhalt hat.

XML-Unterstützung in relationalen DBMS

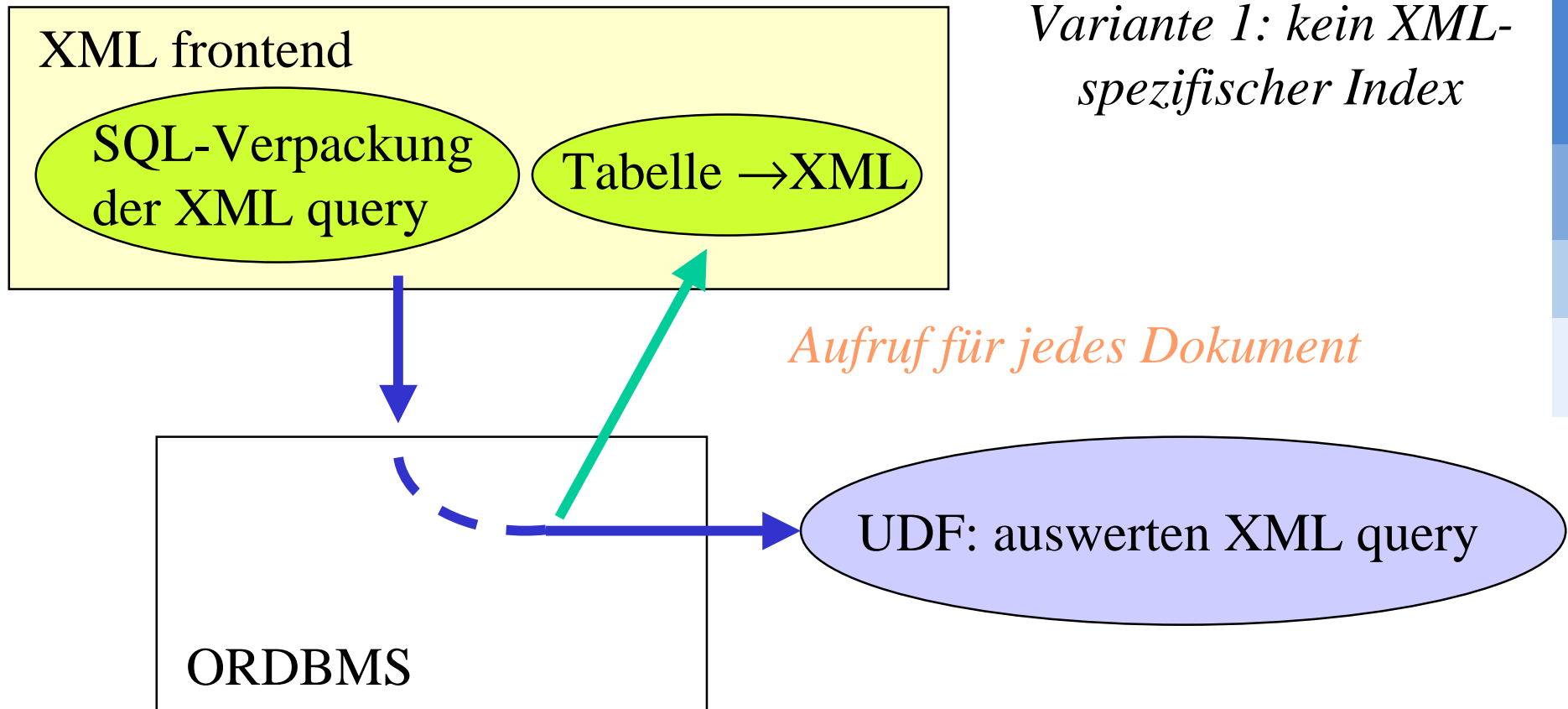
■ Generelles Problem:

- Für eine Anfrage muß bekannt sein
 - Speicherungsmethode des Dokumentes
 - bei inhaltsorientierter Zerlegung: Struktur des Dokumentes
- Änderungen der Speicherungsform (zerlegt, opak) zwingt zur Änderung aller Anwendungsprogramme

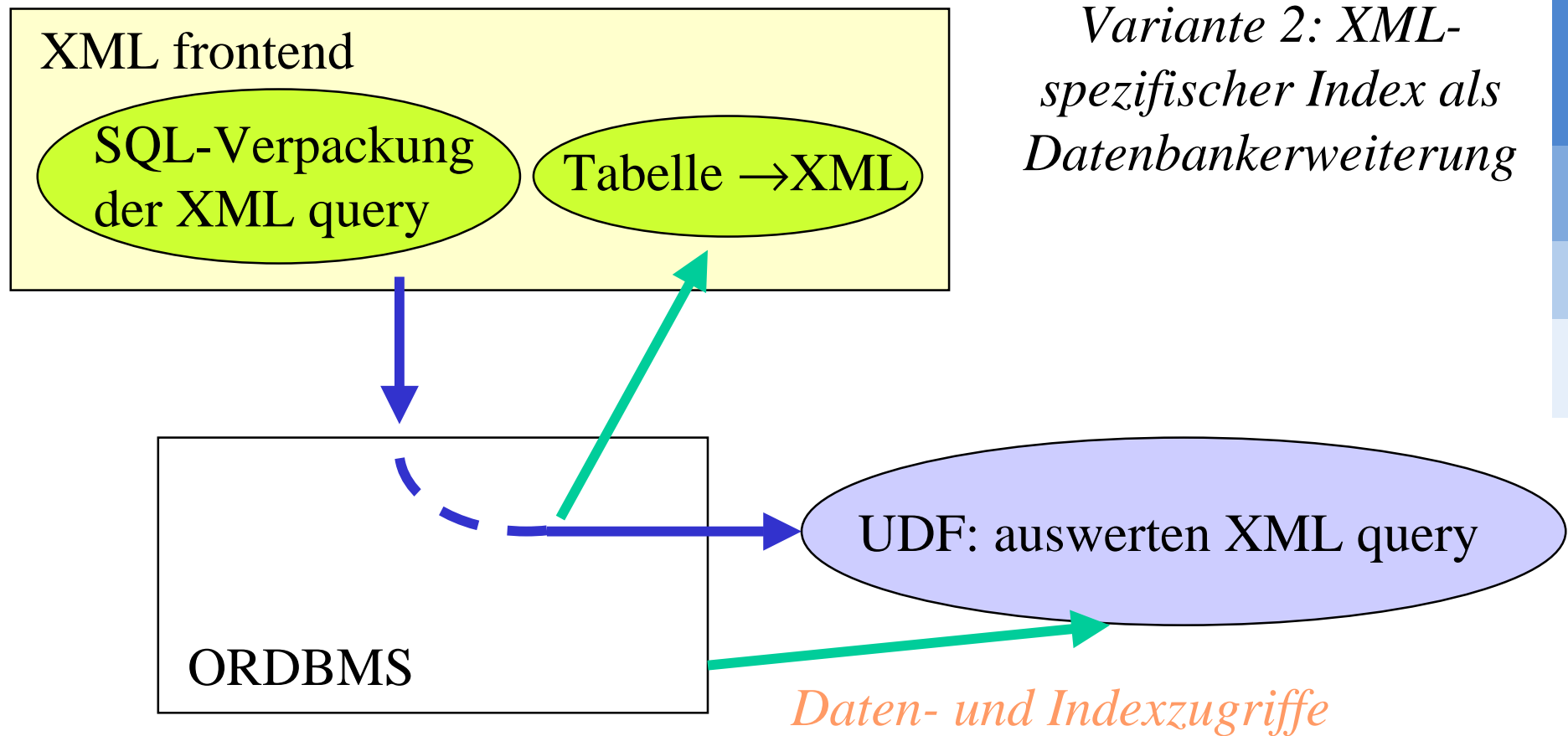
■ Ergebnis ist immer eine Tabelle, die nach XML konvertiert werden muß

■ Anfrage muß in SQL übersetzt oder verpackt werden

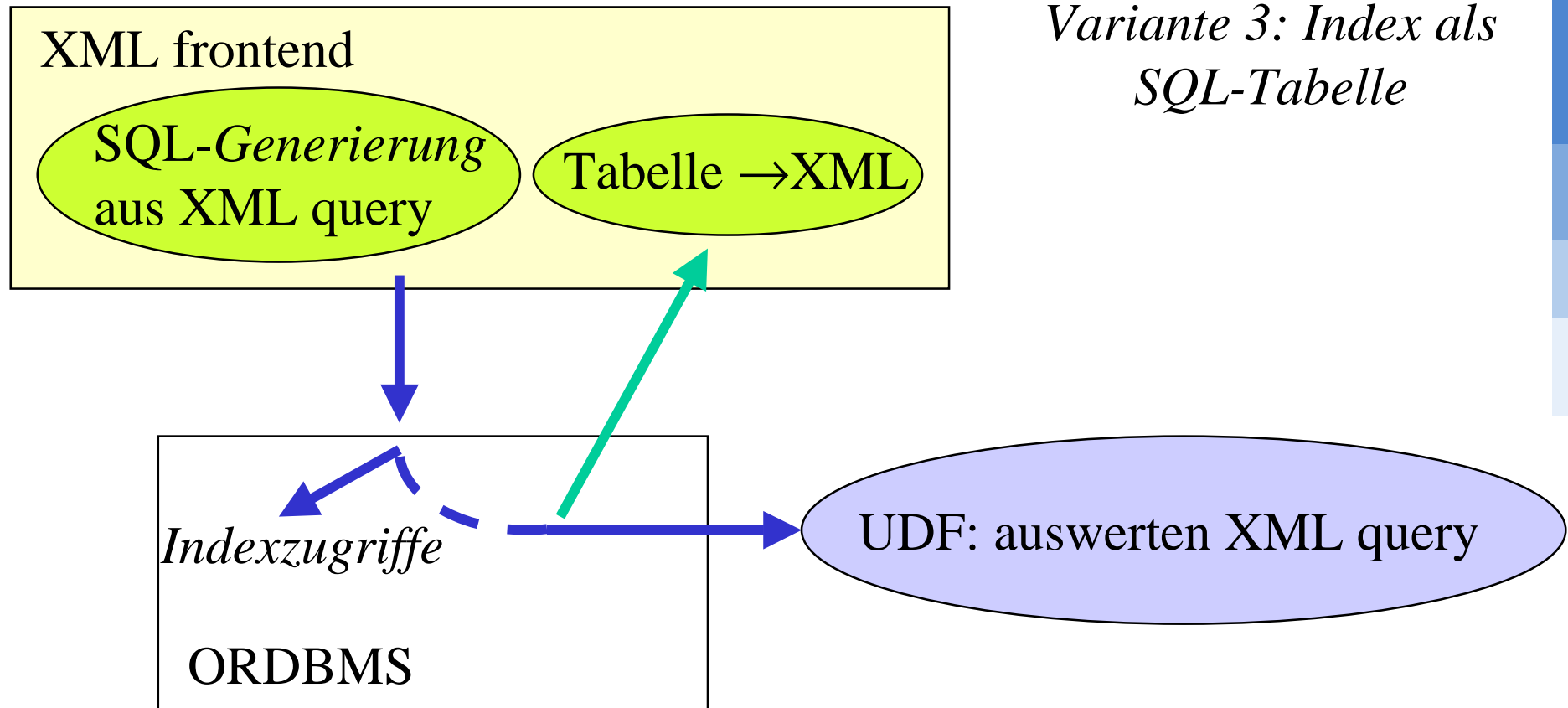
XML-Anfragen (opak): Kontrollfluß in ORDBMS



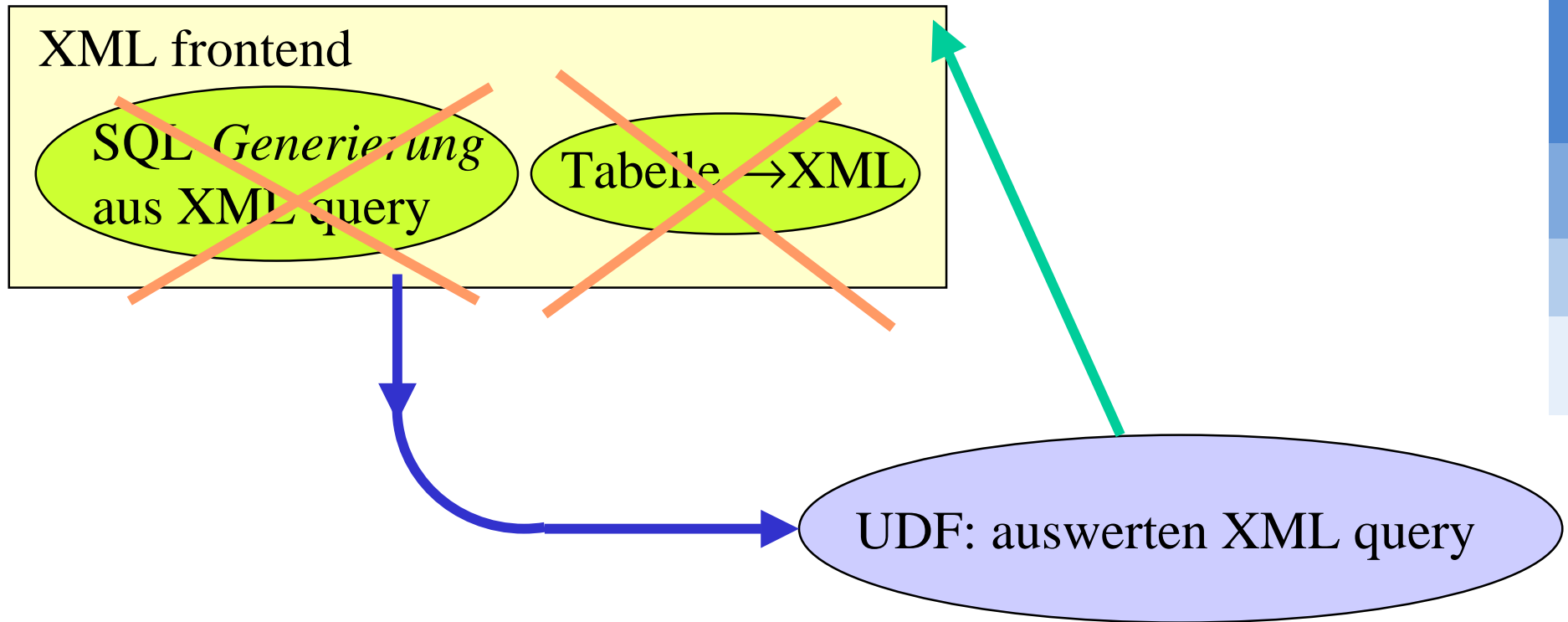
XML-Anfragen (opak): Kontrollfluß in ORDBMS (2)



XML-Anfragen (opak): Kontrollfluß in ORDBMS (3)



Wozu das ORDBMS?



XML-Datenbanksystem

- **Kann XML-spezifische Anfragesprache verarbeiten**
 - z.B. XPATH, XQuery
- **kann XML-Dokumente als solche speichern**
- **liefert XML als Ergebnis**
- **unterstützt struktur-, text- und wertebasierte Anfragen effizient**
- **unterstützt Daten- und Dokumentsicht (mixed content, Kommentare etc.)**
- **erlaubt eine schematische Beschreibung der Dokumente**

The logo for Tamino, featuring the word "Tamino" in a black, handwritten-style font. The letters are partially obscured by a thick, orange brushstroke that runs horizontally across the middle of the text.

ein XML- Datenbanksystem

- Seit 1999 verfügbar
- Anfragesprache: Xpath-Erweiterung, XQuery
- Integration heterogener Datenquellen
- Erweiterbar durch "Server extensions"
- speichert XML-Dokumente und "non-XML"
- schematische Beschreibung ist möglich, aber nicht zwingend
- http ist primärer Zugang
- Indizierung gemäß XML-Struktur

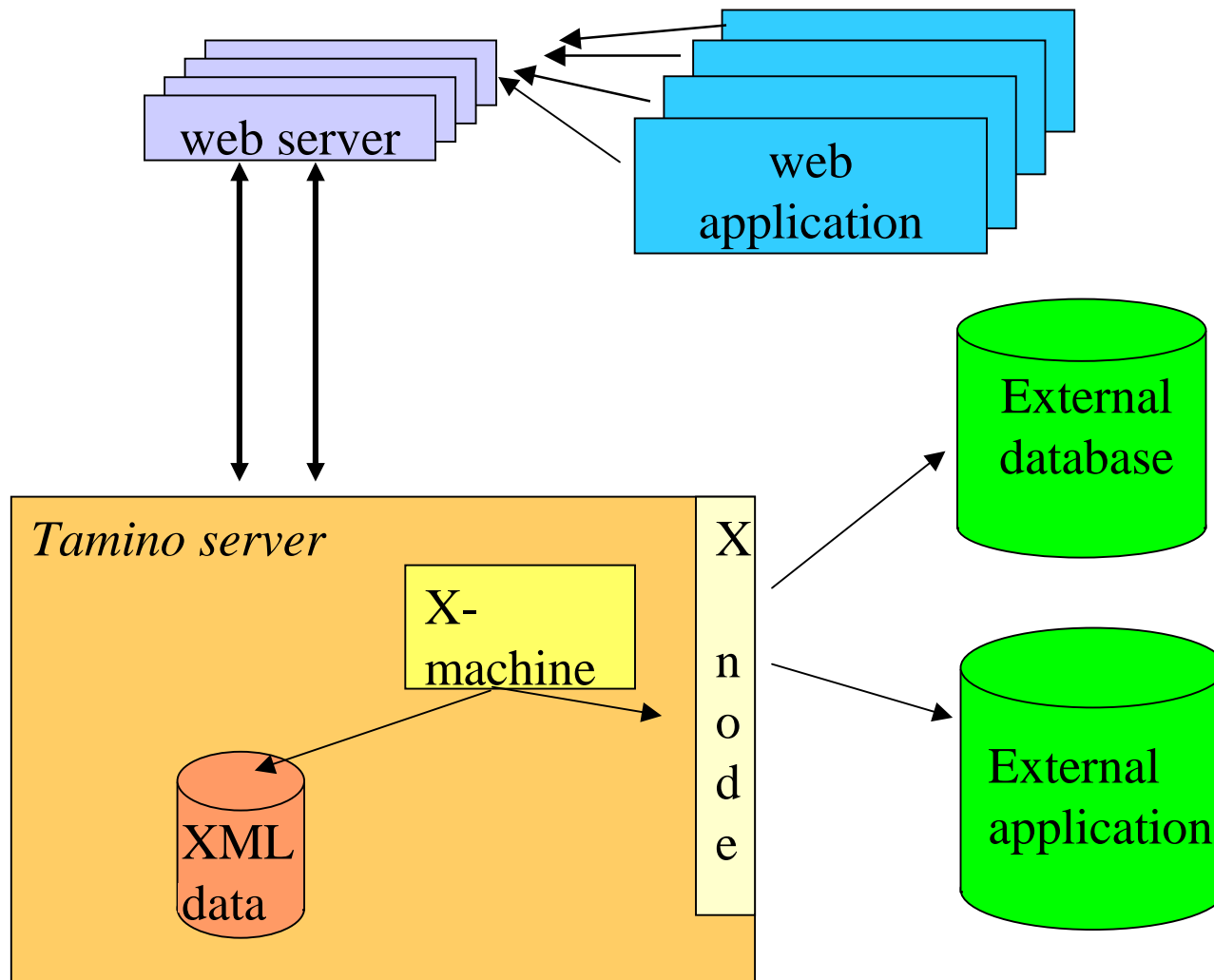
XML als Ergebnis

- **Mengenorientierung erfordert result wrapping**
 - kann entfallen, wenn Ergebnis nur 1 Element hat
- **Ebenso: Möglichkeit der Attributabfrage**
- **Ziel: well-formed XML**
- **Beispiel: //Firma**

Ergebnis:

```
<xql:result>  
<Firma>Software AG</Firma>  
<Firma>IBM</Firma>  
</xql:result>
```

Architekturüberblick



Integration anderer Datenquellen

- **Warum: legacy Daten sollen per Web zugänglich gemacht werden**
- **Daten anderer Quellen sollen mit den in Tamino gespeicherten Daten ein gemeinsames Dokument ergeben**
- **Als XML gelieferte Daten sollen (teilweise) in operative Datenbanken fließen**
- **Zugang z.B. zu ERP-Systemen**

Erweiterbarkeit

- **Integration benutzerdefinierter Funktionen in die Anfragesprache**
 - server extensions
- **Abbildungsfunktionen, die beliebige Abbildungen zulassen (z.B. auf Email oder ein ERP-System)**
 - lesend (d.h. bei Anfragen)
 - schreibend (d.h. bei Einfügungen, Änderungen, Löschungen)
- **Server extensions sind COM oder Java-basiert**

Der Zweck bestimmt die Speicherung

```
<Bestellung Datum="29.09.1999">  
  <Ware Menge=5>Bleistift</Ware>  
  <Ware Menge=2>Schreibblock</Ware>  
</Bestellung>
```

- *vollständig intern:*

```
<Bestellung Datum="29.09.1999"><Ware ...
```

- *extern: Datenbank AuftragsDb, Tabelle Bestellung,...*

- *prozedural: Abbildungsfunktion(<Bestellung...)*

Externe Speicherung

- **“middleware”- Ansatz**
- **Dokument wird geparst, entsprechende Informationen werden in externen Datenhaltungssystemen gespeichert**
- **bei Anfrage entsprechendes Lesen**
- **Kenntnis der Dokumentstruktur erforderlich**
- **verteilte Transaktionen**
- **Dokumentenänderung an Tamino vorbei**
 - Indizierung im Fremdsystem
 - Suchmöglichkeiten nicht durch Fremdsystem beschränkt

Prozedurale Speicherung

- **Aufruf einer Prozedur für (Teil-) Baum statt Speicherung / Lesen aus der Datenbank**
- **Verhalten der Prozedur offen**
 - Verschicken von Mail
 - Speicherung in ERP-System
 - ...
- **round trip**
 - vom Verhalten der Prozedur abhängig

Kombination

- **Die vorgestellten Speicherungsvarianten lassen sich untereinander und mit der internen Speicherung auf Dokumentenebene kombinieren**
- **Dokumentstruktur wird ohnehin abgebildet**
 - ermöglicht strukturbasierte Suche
- **vollständige Speicherung für nicht weiter spezifizierte Teilbäume möglich**
 - Schema-Evolution wird unterstützt

Indizierung

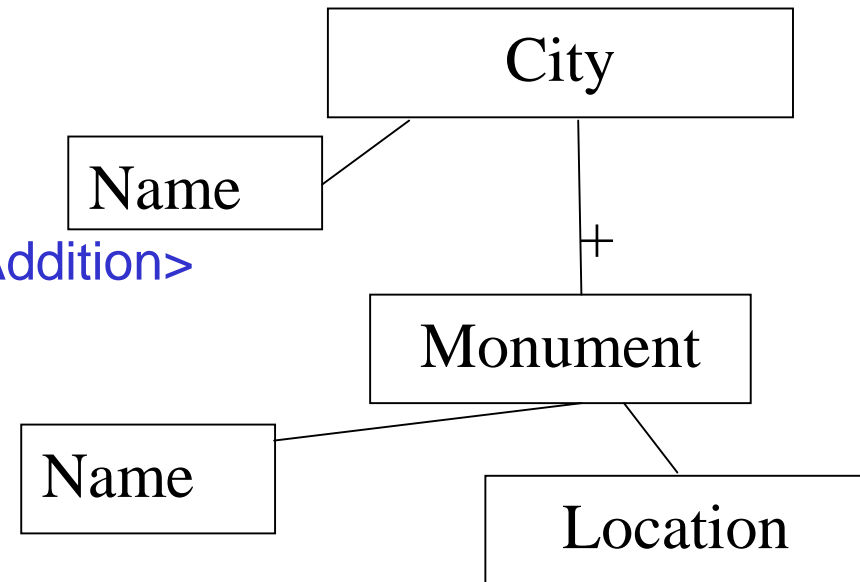
- **klassische Indizes für Daten**
 - Zahlen
 - Zeichenketten
 - Datentypdefinition erforderlich
- **Textindizes für dokumentartige Teile**
 - mit Wildcards
- **Struktur**
- **Frei kombinierbar**
 - innerhalb eines Dokumententyps

Schema ist optional

- **Ein XML-Datenbanksystem darf kein bekanntes Schema voraussetzen**
 - DTD-lose / schemalose Dokumente müssen verarbeitet werden
- **Kommentare / processing instructions müssen gespeichert werden**
 - obwohl nicht im Schema deklariert
- **Offenheit des Schemas von Anwendungsszenarien gefordert**
 - das ist nicht das ANY aus DTDs
- **Tamino**
 - Schema-Sprache: XML Schema
 - *open content* oder *closed content*

Open content in Tamino - Beispiel

```
<City Inhabitants="138000">
  <Name>Darmstadt</Name>
  <Addition>The city of art nouveau</Addition>
  <Monument Height="39m">
    <Name>Langer Ludwig</Name>
    <Location>
      <Name>Luisenplatz</Name>
      <MapIndex>M5</MapIndex>
    </Location>
  </Monument>
</City>
```



```
<!ELEMENT City (Name, Monument+)>
<!ELEMENT Monument (Name, Location)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Location (#PCDATA)>
```

Open content in Tamino

■ Strikte Schema-Validierung

- *closed content* definieren

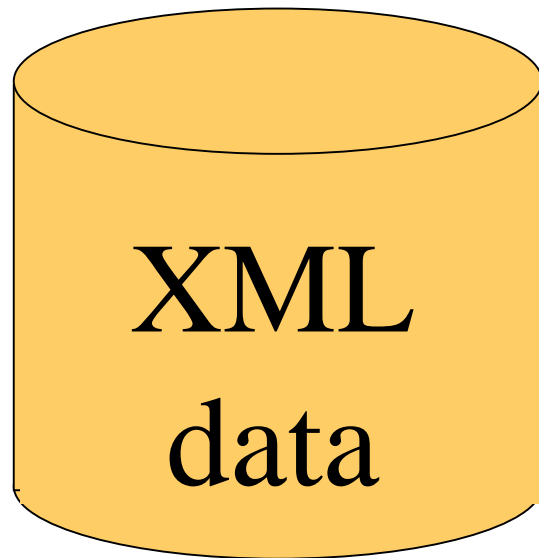
■ Ansonsten: Elemente und Attribute definieren, wenn

- sie indiziert werden sollen
- wenn sie auf externe Datenquellen oder server extensions abgebildet werden sollen
- Zugriffsrechte darauf definiert werden sollen
- Eigenschaften (z.B. Multiplicity) erzwungen werden sollen
- einer dieser Punkte für ein Kindelement gilt

Tamino Schemabehandlung

- **Dokumente ohne bekanntes Schema können gespeichert werden**
 - keine schemabasierte Gruppierung
 - Tamino collections als benutzerdefinierte Gruppierung
- **Kommentare und processing instructions werden gespeichert**
 - Unabhängig vom content Modell

XML Datenorganisation



- **Collection**
 - Dokument-Typ
 - dokumente
- **eigene Collection für “unbekannte Dokumenttypen” (well-formed XML)**
- **Auch nicht-XML-Objekte (*.gif etc)**
- **physische Organisation**
 - Dokumenteninhalt
 - Struktur
 - Indizes

Existierende *Tamino* Anwendungen

- **Datenaustausch, z.B.**
 - mobile Anwendungen
 - Gesundheitswesen
- **Dokumentenverwaltung / content management**
- **Dokumentgenerierung aus Bausteinen**
- **B2C, webshops , B2B, business integration (electronic mall)**
- **intelligente Suchmaschinen**
- **Katalogverwaltung komplexer Teile (web2CAD)**
- **Knowledge management**
- **e-government (UK)**
- **persistente Speicherung von XML (Messaging)**
- **Metadatenverwaltung / Servicebeschreibung**

Zusammenfassung

- **XML-Speicherung ist eine wichtige Anforderung an Datenbanksysteme im e-Commerce**
- **Zugriff auf vorhandene Daten (RDBMS) im XML-Format ist einfach**
- **direkter Anschluss an das Web**
- **Speicherung von XML-Dokumenten in (objekt-) relationalen Datenbanksystemen ist eine Notlösung**
- **Speziell auf XML zugeschnittene Datenbanksysteme bieten bessere Unterstützung**

Mehr Information....

- Mein Buch „XML und Datenbanken“
erscheint im November 2002
im Carl Hanser Verlag
- Tutorientage im Rahmen der
Fachtagung Datenbanksysteme in
Business, Telecommunication and Web
(BTW 2003) in Leipzig.
<http://www.btw2003.de/>
25.2.2002: XQuery und SQL/XML
- Seminar der Deutschen Informatik Akademie
XML und Datenbanken
4. – 5. 6. 2003
<http://www.dia-bonn.de/>

