# XML-Datenbanksysteme

Dr. Harald Schöning





## Überblick

- **Motivation: XML und e-Commerce**
- **XML in relationalen Datenbanken**
  - Prinzipien
  - konkrete Systeme
- **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>Datenbanksysteme im e-Commerce</Thema>**
  **<Vortragender>Dr. Harald Schöning**
  **<Firma>Software AG</Firma></Vortragender>**
  **</Vortrag>**

3

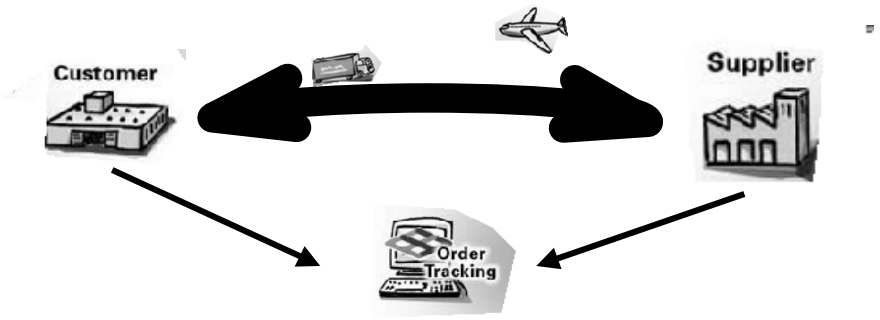


## Datenaustausch - Schlüssel zum e-Commerce

- **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

4

## Supply chain




5

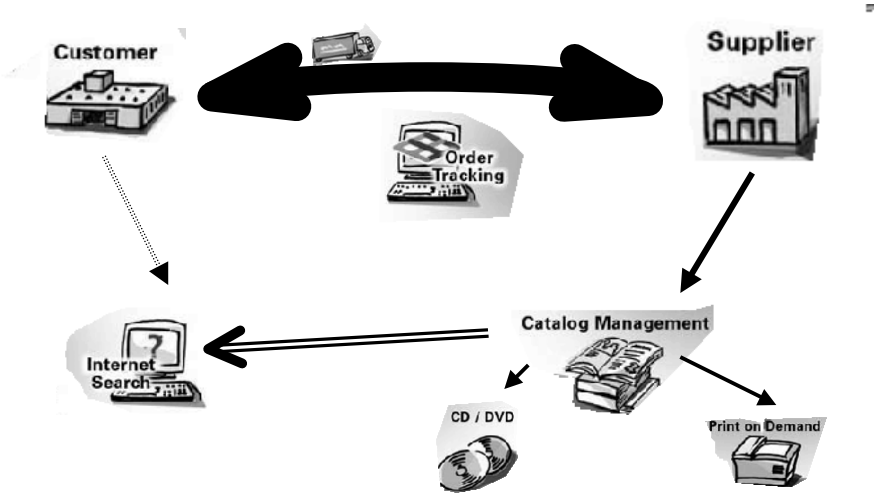


## Flexible Formatierung

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

6

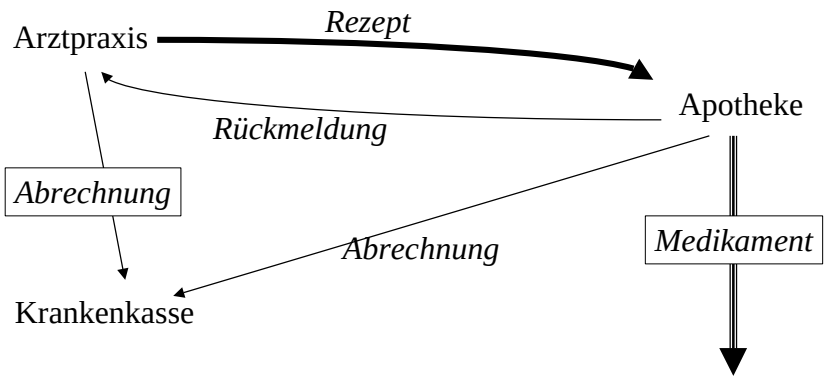
## … und integriertes Content Management




7



## Beispiel aus dem Gesundheitswesen




8

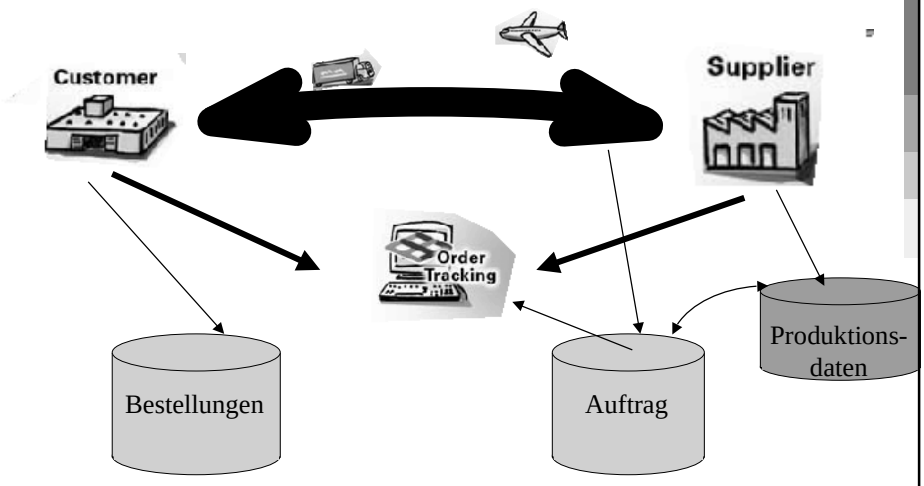
## Warum XML in Datenbanken?

- **Damit besteht auch der Bedarf der Speicherung**
  - effizient
  - sicher
  - mit Abfragemöglichkeiten
- **Aber auch Anbindung an vorhandene DBS!**

9

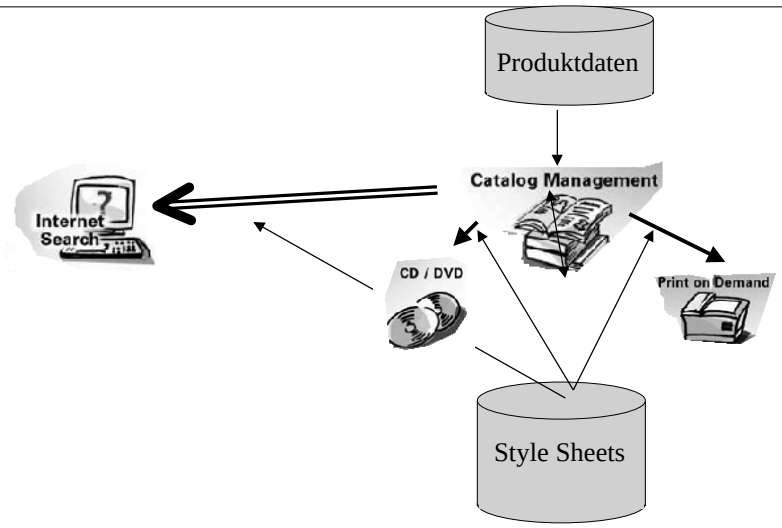


## Warum XML in Datenbanken?




10

## Warum XML in Datenbanken?








## Warum XML in Datenbanken?

- **Andere Anwendungen**
  - Dokumentenverwaltung
  - Website-Management
  - electronic mall
  - ...

## Daten oder Dokumente?

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

13



## 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

14

## Anfragetypen

- **Werteorientiert:**
  - @attribute > 5
    - <element attribute="4" />
  - element < 7.1
    - <element>6.1</element>
  - element = "Hugo"
    - <element>Hugo</element>
    - <element><subelement>Hugo</subelement></element>
    - <element><B>H</B>ugo</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"]

## Anfrageergebnisse

- **Dokument / Menge von Dokumenten**
  - Prolog!
- **Element / Menge von Elementen**
- **Attributwert / Menge von Attributwerten**
- **Elementwert / Menge von Elementwerten**
- **Funktionsergebnis(se)**
- **Processing instructions?**
- **Kommentare ?**
- **Prolog?**
- **DTD?**





## XML in Datenbanken - Optionen zur Realisierung

- **relational generisch**
- **relational inhaltsorientiert zerlegt**
- **relational strukturorientiert zerlegt**
- **relational opaque**
- **objektorientiert**
- **XML (Tamino)**

XML AK München - Jan. 2001

## 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

21

XML AK München - Jan. 2001

## 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)

22

## Relational 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>

23



## Relational 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 (generisch) - nicht nur wrapping:**
  - Abspeichern von XML-Dokumenten in vorgegebene Tabellen
  - Setzt eine veränderbare Sicht oder Basistabelle voraus
  - Die XML-Dokumente müssen in der Struktur der Tabelle entsprechen

24

## Relational 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**

25



## Relational inhaltsorientiert zerlegt

- **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**

26

## Relational inhaltsorientiert zerlegt (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>
```

27



## Relational inhaltsorientiert zerlegt (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 $\rightarrow$ XML notwendig

28

## Relational inhaltsorientiert zerlegt (4)

- **Auflösen synonymer Darstellungen: &apos &#x0027 ´**
- **Vorteil: Daten sind auch für SQL-Applikationen verfügbar**
- **verwendete relationale Strukturen reflektieren Benutzersemantik**





## 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

## 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**

31



## 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

32

## Relational opaque

- **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

33



## Anfragen über Extensions

- **Anfragen über**
  - Satz von zugeschnittenen UDF:

```
SELECT xml
FROM xmlTable
WHERE CONTAINS (xml, “Datenbanksysteme”)
```

  - 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?**

34

## 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 wäre Erkennung als numerisches Feld
  - DB2: side tables
    - Konsistenz nicht garantiert!
    - Auch syntaktisch getrennt
    - ggf. nested tables (multiplicity > 1)

35



## 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.

36

## Update bei Verwendung von Extensions

- **UPDATE xmlTable**
  **SET xml = TransformXml(xml, “replace <a> by <b>”)**
  - ersetzt den kompletten Attributwert, d.h. das gesamte Dokument!





## XML-Unterstützung in Oracle 8i

- **Generisch (XML SQL utility)**
- **inhaltsorientiert zerlegt (iFS)**
- **opaque (iFS)**
- **Kombination aus inhaltsorientiert zerlegt und opaque**
- **keine wertebasierte Suche bei opaque**
- **textorientierte Suche möglich, wenn entsprechende Spalte textindiziert ist**
- **strukturorientierte Suche nicht voll möglich**
- **kein partielles Update im opaque-Fall**
- **keine Extraktion im opaque-Fall**

## XML-Unterstützung in Oracle 8i (2)

- **XML SQL Utility for Java**
  - im Prinzip nicht DBMS-gebunden
  - generiert aus Query-Ergebnis XML (d.h. Struktur entsteht dynamisch gemäß der SQL-Anfrage)
    <ROWSET>
    <ROW>…</ROW>
    </ROWSET>
  - liefert String oder DOM-Repräsentation
  - kann auch aus Tabellen DTDs generieren, mit denen ein Java Class Generator arbeiten kann

39



## XML-Unterstützung in Oracle 8i (3)

- **XML SQL Utility for Java - Forts.**
  - kann XML in die Datenbank speichern, wenn es entsprechend formatiert ist
    - ein XML-Dokument wird zu *einer* Zeile *einer* Tabelle
    - objektrelationale Fähigkeiten von Oracle erlauben geschachtelte Tabellen, somit auch tief geschachtelte XML-Dokumente
    - Keine Werkzeugunterstützung bei der Definition der Tabelle zu einer DTD
    - Attributbehandlung?
    - Rekursion?
  - Keine XML-Query-Unterstützung

40

## XML-Unterstützung in Oracle 8i (4)

- **XSQL**
  - "dynamic XML"
  - Einbettung von SQL-Anfragen in XML-Dokumente, zur Laufzeit ausgewertet
  - Anbindung an Style-Sheets erzeugt gewünschte Struktur
- **iFS**
  - document descriptions für inhaltsorientierte Zerlegung, opaque





## XML-Unterstützung in DB2

- **Über DB2 XML-Extender**
  - inhaltsorientierte Zerlegung (XML collection)
  - oder opaque (XML column)
- **Steuerung über DAD (data access definition)**
  - kann aus DTD erzeugt werden
- **DTD-Repository**
  - optional Validierung

## XML-Unterstützung in DB2 - DAD Beispiel

```
<?xml version ="1.0">
<!DOCTYPE DAD SYSTEM "c:\dad.dtd">
<DAD><dtdid>c:\person.dtd</dtdid><validation>YES</validation>
<Xcolumn>
        <table name="person_phone_number">
            <column name="pnumber" type="varchar(20)"
            path="/person/phone/number" multi_occurrence="YES"/>
        </table>
        <table name="person_phone_type">
            <column name="ptype" type="varchar(20)"
            path="/person/phone/type" multi_occurrence="YES"/>
        </table>
</Xcolumn>
```

43



## XML-Unterstützung in DB2 (3)

- **UDFs zum Speichern und Suchen**
- **Extraktion mittels XPATH-Ausdruck möglich**
  - liefert skalaren Wert
  - oder Tabelle
- **deskriptives Update über XPATH-Ausdruck möglich**
- **Textsuche mittels Textextender**
- **auch eingeschränkte strukturorientierte Suche (z.B. keine Attribute)**

44

## XML-Unterstützung in DB2 (4)

- **wertebasierte Suche im opaque-Fall möglich, wenn Side tables definiert wurden**
  - Extraktion von Werten aus dem XML-Dokument in Tabellen
  - Trigger-basiert
    - erst nach Definition der side tables gefüllt
  - Benutzeränderungen auf side tables führen zu Inkonsistenz

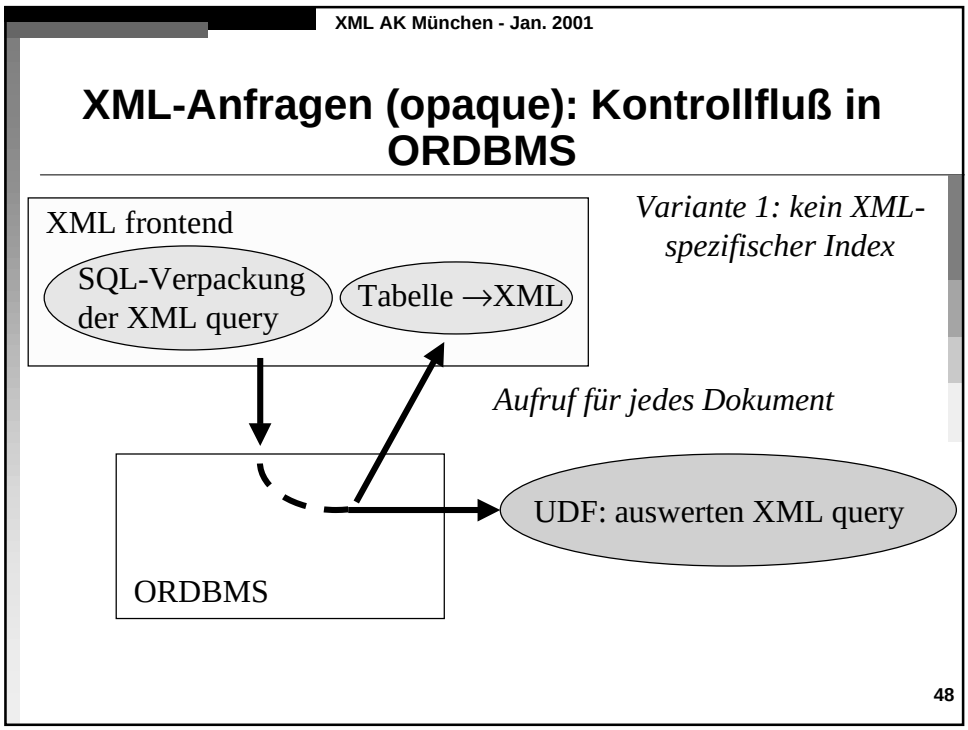
45

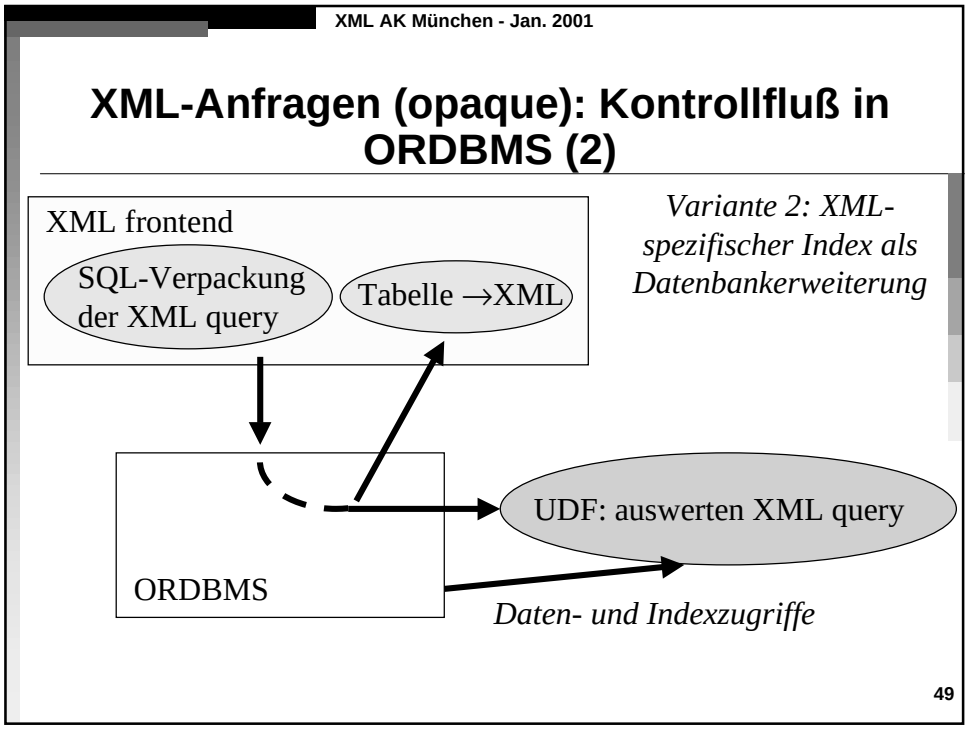


## XML-Unterstützung in relationalen DBMS

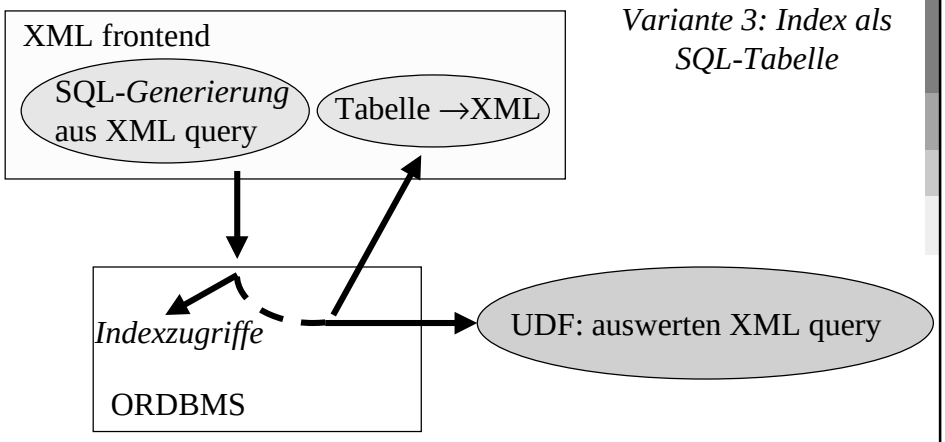
- **Generelles Problem:**
  - Für eine Anfrage muß bekannt sein
    - Struktur des Dokumentes
    - Speicherungsmethode des Dokumentes
  - Änderungen der Speicherungsform (zerlegt, opaque) zwingt zur Änderung aller Anwendungsprogramme
- **Ergebnis ist immer eine Tabelle, die nach XML konvertiert werden muß**

46


XML AK München - Jan. 2001
XML-Anfragen (opaque): Kontrollfluß in ORDBMS
XML frontend
SQL-Verpackung der XML query
Tabelle →XML
Variante 1: kein XML-spezifischer Index
Aufruf für jedes Dokument
ORDBMS
UDF: auswerten XML query
48


XML AK München - Jan. 2001
XML-Anfragen (opaque): Kontrollfluß in ORDBMS (2)
XML frontend
SQL-Verpackung der XML query
Tabelle →XML
Variante 2: XML-spezifischer Index als Datenbankerweiterung
ORDBMS
UDF: auswerten XML query
Daten- und Indexzugriffe
49

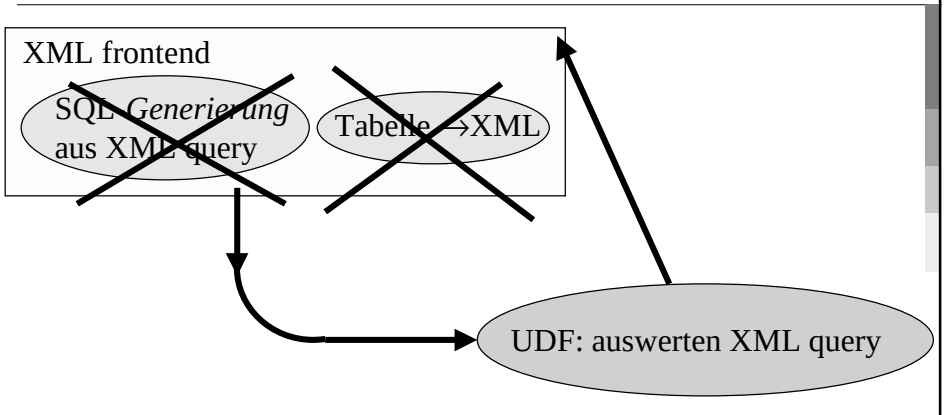
## XML-Anfragen (opaque): Kontrollfluß in ORDBMS (3)








## Wozu das ORDBMS?



## XML-Datenbanksystem

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

52



## Tamino ein XML-Datenbanksystem

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

53

## Anfragesprache

- **Kompatibel zu XPATH**
  - /talk
  - //speaker[company="Software AG"]
  - //@time
  - /talk/*/company
- **mengenorientiert**
- **Erweiterungen, z.B. für Textsuche**
  - /talk[title ~= "XML"]



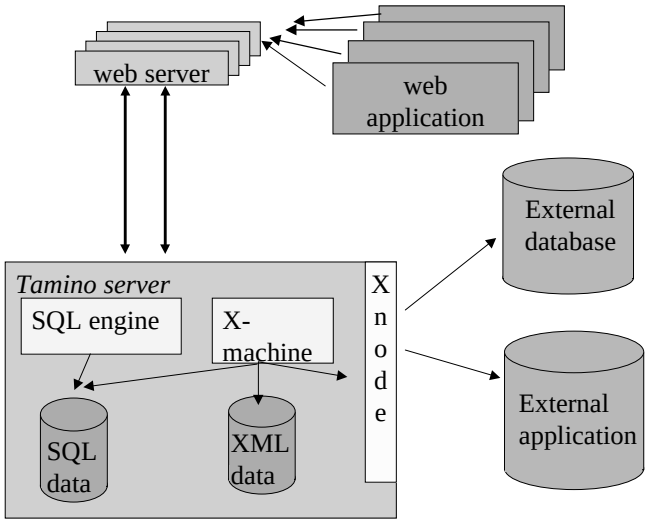


## XML als Ergebnis

- **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**

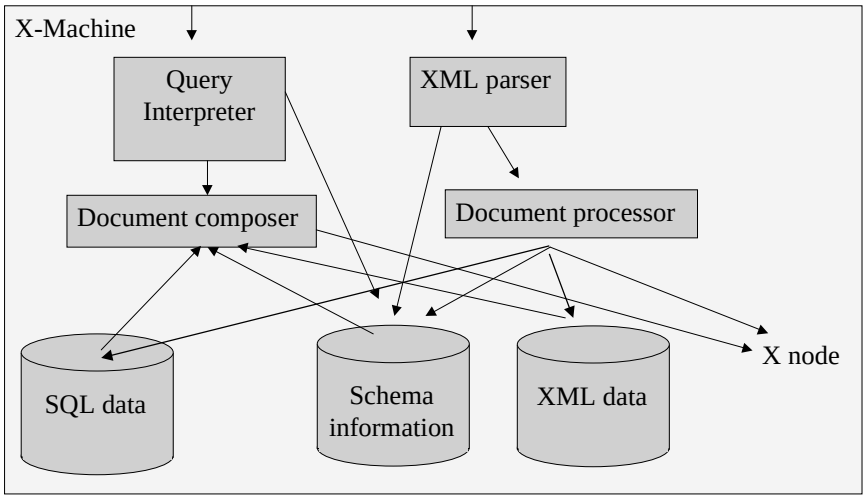
## Erweiterbarkeit

- **Integration benutzerdefinierter Funktionen in XQL**
- **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**

58

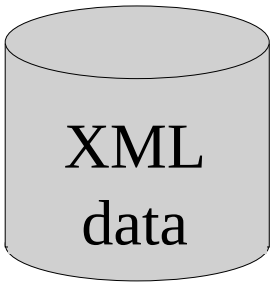


## Architektur der X-Maschine




59

## 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





## Der Zweck bestimmt die Speicherung

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

- ***vollständig*:** <Bestellung Datum="29.09.1999"><Ware ...
- ***Information*:**

```
29.09.1999
        5, Bleistift
        2, Schreibblock
```

- ***extern: Datenbank AuftragsDb, Tabelle Bestellung,...***
- ***prozedural*: Abbildungsfunktion(<Bestellung…)**

## Vollständige Speicherung

- **Dokument wird geparst**
  - zur Erstellung von Indizes
- **wird unverändert abgespeichert**
- **Kenntnis der Dokumentstruktur nicht erforderlich**
- **Markup nimmt Platz in Anspruch**
- **Anfragen, die nicht über Indizes abgehandelt werden können, erfordern Parser aller Dokumente**
- **Extraktion erfordert Parsen der Trefferdokumente**
  - //Firma[text() contains “AG”]

62



## Speichern der Information

- **Dokument wird geparst**
- **und die Informationen (ohne Markup) gespeichert**
- **erfordert Kenntnis der Dokumentstruktur**
- **Suche erfordert kein Parsen**
- **Rekonstruktion des Gesamtdokumentes aufwendiger**
  - Kommentare
  - Processing Instruktions

63

## Externe Speicherung

- **“middleware”- Ansatz**
- **Dokument wird geparst, 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 auf Dokumentebene kombinieren**
- **Dokumentstruktur wird ohnehin abgebildet**
  - ermöglicht strukturbasierte Suche
- **vollständige Speicherung für nicht weiter spezifizierte Teilbäume möglich**
  - Schema-Evolution wird unterstützt

66



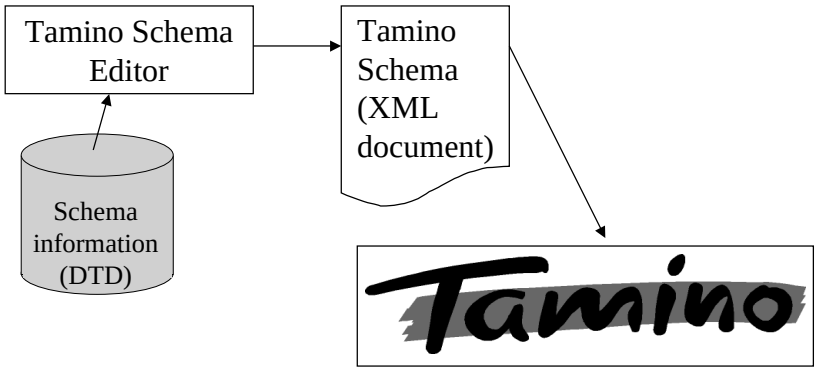
## Indizierung

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

67

## Vorgehen bei der Index-Definition

- **Definition der Abbildung**



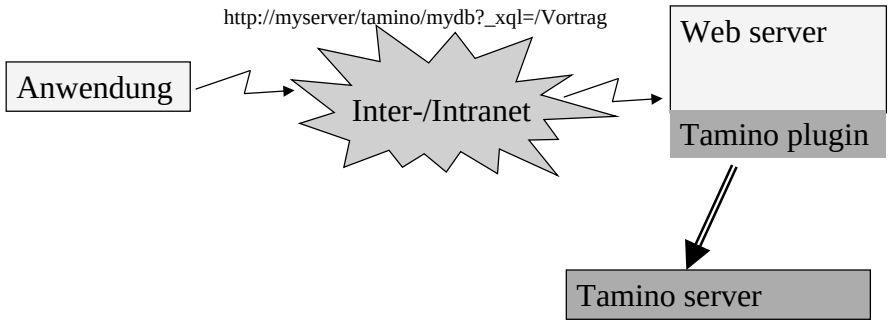





## Schema-Sprache

- **Einzige normierte XML-Schemasprache: DTD**
- **Aus Datenbanksicht unzulänglich**
  - keine Datentypen
  - keine Erweiterbarkeit (Indizierung)
  - kein XML!
- Bisher: proprietäre Tamino-Schemasprache und Werkzeug zur Erzeugung aus DTD
- **Beim W3C in Arbeit: XML Schema**
  - Basis für neue Tamino-Schemasprache

## http als primärer Zugang




- **Beschränkung der akzeptierten Web-Server**
- **Security-Check schon im Web-Server möglich**
  - wegen hierarchischer URL-Struktur

70



## Zusammenfassung

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

71