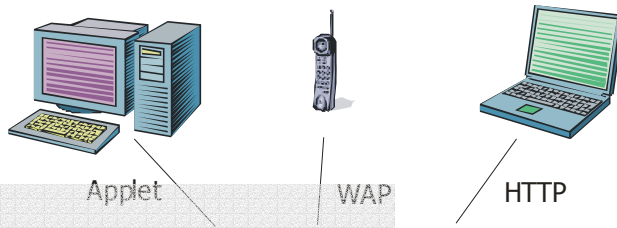


Datenbanken und XML

Wassilios Kazakos (FZI)

- Ein paar Grundgedanken
 - ▶ Wie können DB-Technologien und XML zusammenspielen
- Datenbanken und XML: Die XML-Perspektive
 - ▶ XML als Datenmodell
 - ▶ XML-Anfragesprachen
- Datenbanken und XML: Die RDBMS-Perspektive
 - ▶ Probleme der Abbildung von XML auf Tabellen
 - ▶ Systembeispiel Microsoft SQL Server



Applet

WAP

HTTP

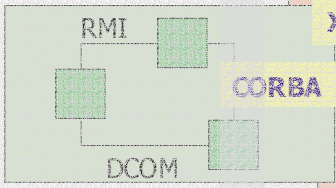
Web-Technologien im Überblick

www.klick-and-bau.com

CGI JSP ASP Servlet XSL-FO

Komponentenframeworks

Verarbeitung von XML-Daten



XSL-T

SQL

Verteilte Transaktionen

JDBC

CICS

Datenaustausch und -zugriff mit XML

XML-basierter Datenbankzugriff

XML Schema

XML Schema XQuery

Architekturen und Systeme zur Informationsintegration

Mediator

OEM

OEM

OEM

Semantische Integration

Wrapper

Wrapper

Wrapper

RDBMS

HOST

RDBMS

DBMS

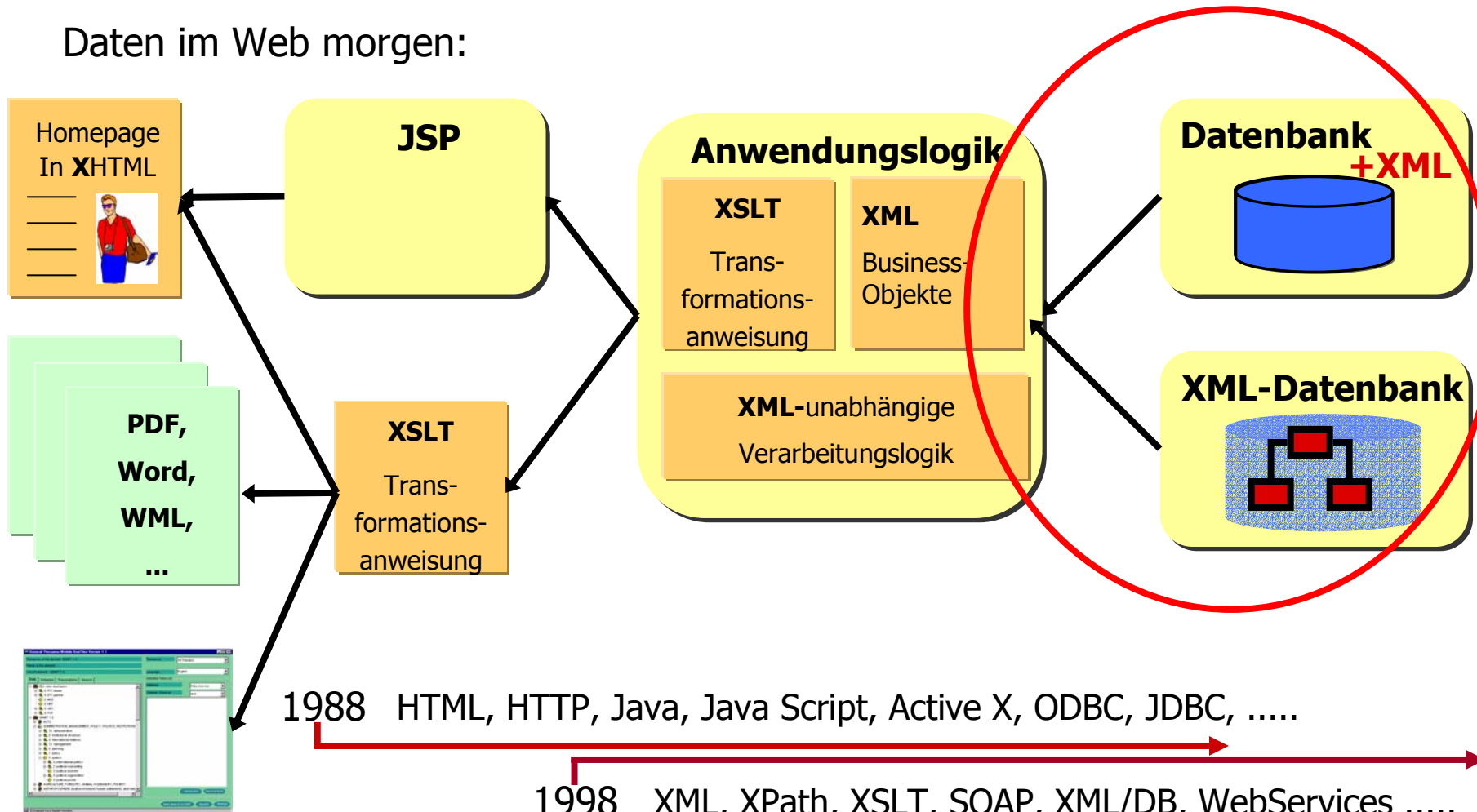
RDBMS

OODBMS

XML-DBMS

Übersicht

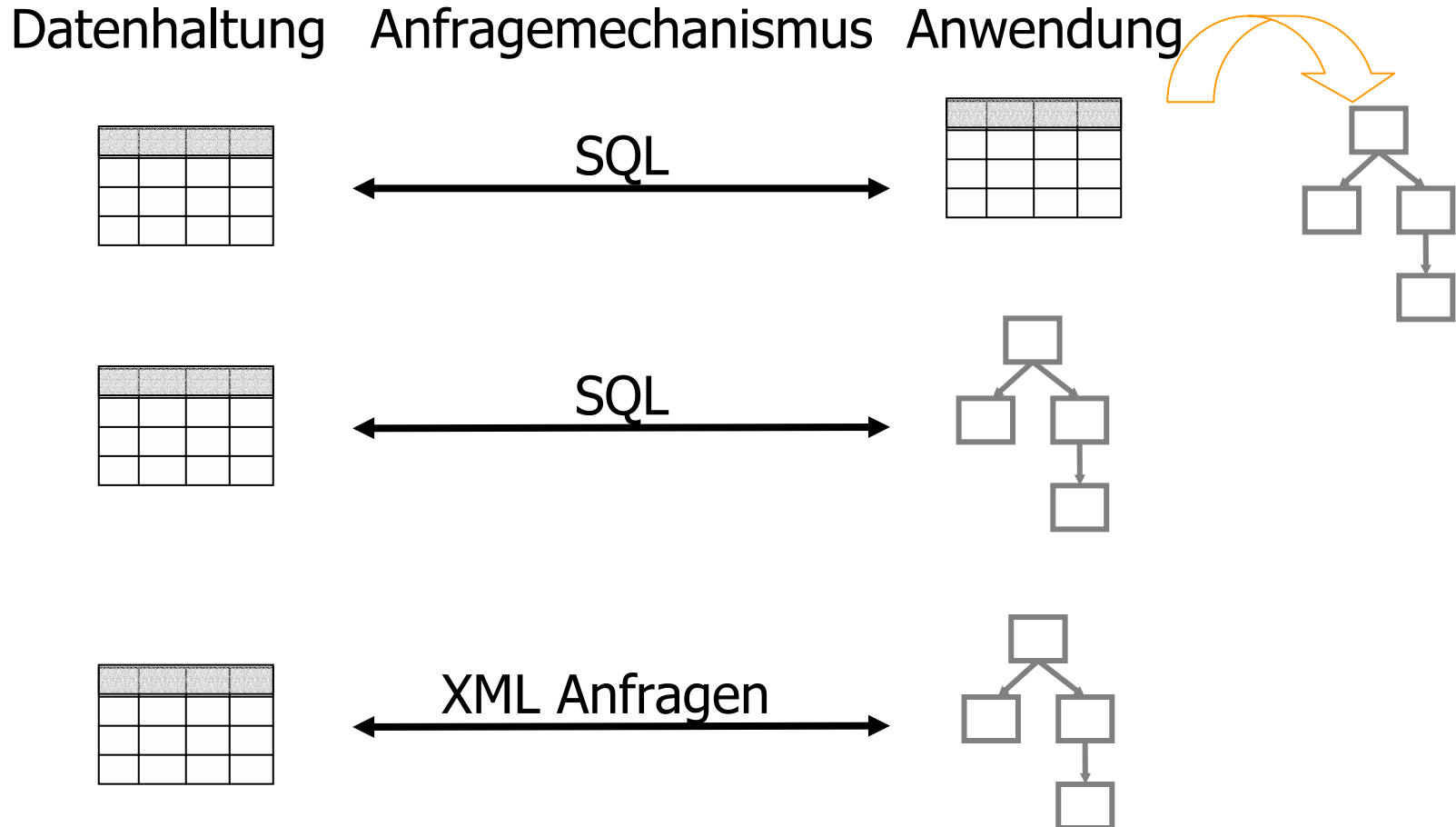
Daten im Web morgen:



1988 HTML, HTTP, Java, Java Script, Active X, ODBC, JDBC,

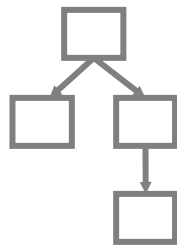
1998 XML, XPath, XSLT, SOAP, XML/DB, WebServices

Grundgedanken

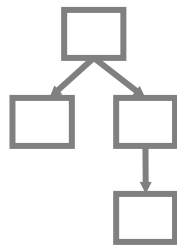
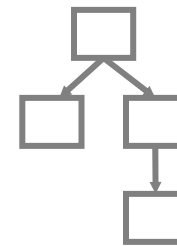


Grundgedanken 2

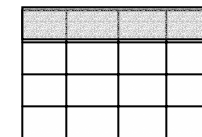
Datenhaltung Anfragemechanismus Anwendung



XML Anfragen



SQL



- Unterschiedliche Anforderungen
 - ▶ Lesender und schreibender Zugriff
 - ▶ Wiederherstellbarkeit der Daten
- Stärke der Kopplung
 - ▶ Datenbank nur als Speicher vs. Verlagerung von Funktionalität in die Datenbank
 - ▶ Wie viel XML im Verarbeitungsmodell?
 - Vergleichbar mit OO-Datenbanken vs. Relationale Datenbanken
- Dokumente vs. Daten
 - ▶ Buchtexte werden anders verarbeitet als Adresslisten
 - ▶ Grad der Strukturierung

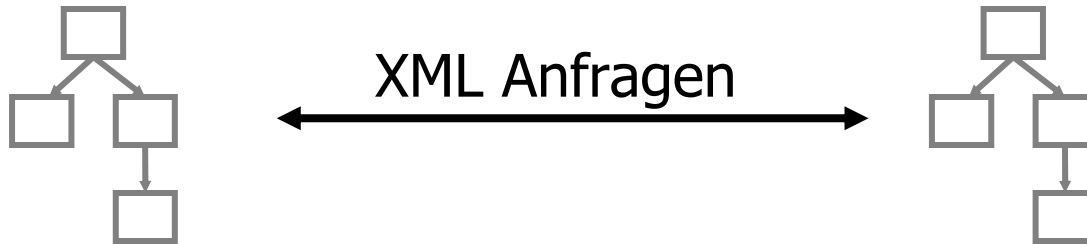
Datenbanken und XML

Die XML Perspektive

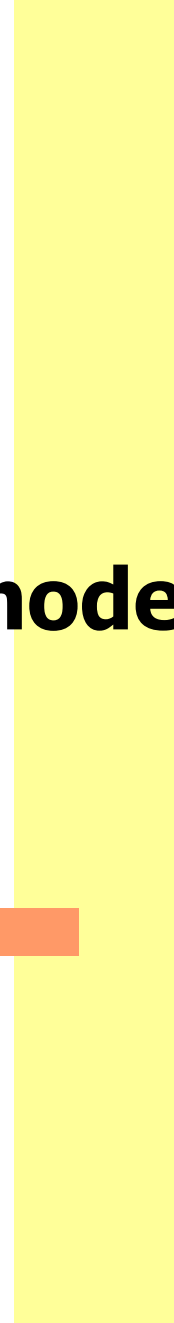


- Wir befinden uns in einer XML-Welt
 - ▶ XML als Datenmodell
 - ▶ XML-Anfragesprachen
 - ▶ Schreibender Zugriff auf XML

Datenhaltung Anfragemechanismus Anwendung



XML als Datenmodell



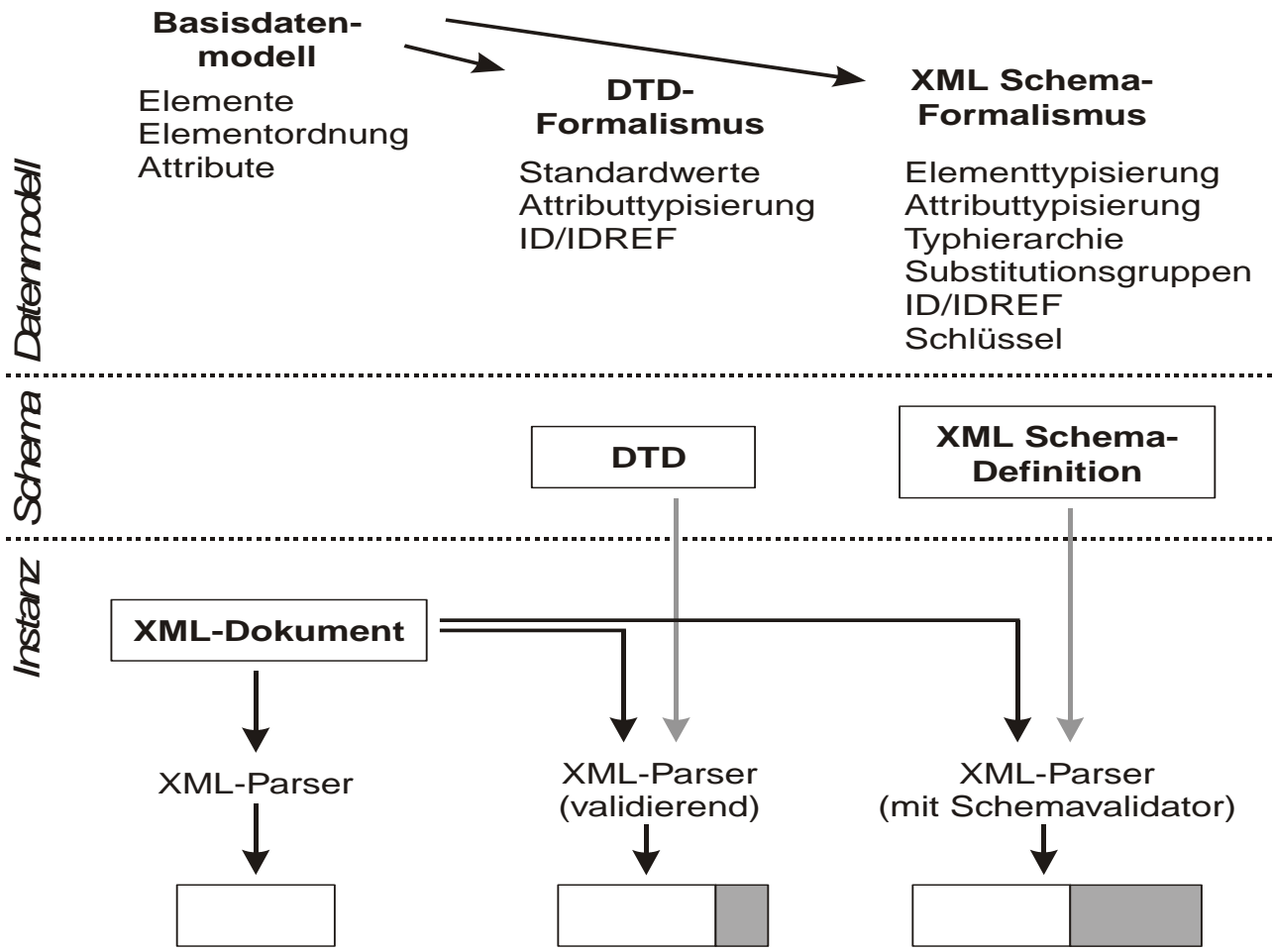
Was ist ein Datenmodell?

- Aus Datenbankeinsatz:
 - ▶ "Eine hinreichend breit akzeptierte Datenbankfunktionalität aus Strukturierungsregeln und Dienstfunktionen"
 - ▶ "Ein Datenbasisverwaltungssystem realisiert ein (einziges) Datenmodell"
- Ein Datenmodell gibt die Strukturierungsregeln vor, mit Hilfe derer die Integrität der Daten (beispielsweise durch ein DBMS) überprüft werden kann.

Ein Datenmodell für XML

- Das XML-Datenmodell hängt von der Validierung ab (!)
 - ▶ Basis-Datenmodell ohne Validierung
 - Elemente, Elementanordnungen und Attribute
 - ▶ DTD-Formalismus
 - Standardwerte, Attributtypisierung, ID/IDREF
 - ▶ XML Schema-Formalismus
 - Elementtypisierung, Attributtypisierung, Typhierarchie, Substitutionsgruppen, ID/IDREF, Schlüssel
- XML-Information Set
 - ▶ Definiert die Informationen die über ein XML-Dokument verfügbar sind.
 - ▶ Standard seit Oktober 2001

Datenmodell Zusammenhang



Beispiel

```
<warenkorb>
  <artikel aid="xy1234" >
    <name>Billy Regal</name>
    <beschreibung> Das ist ein Regal </beschreibung>
    <preis>98</preis>
  </artikel>
  <artikel aid="xy1235" artikel_ref="xy1234">
    <name>Billy Brett</name>
    <beschreibung>Zusatz zu Billy Regal</beschreibung>
    <preis>98</preis>
  </artikel>
</warenkorb>
```

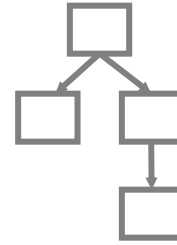
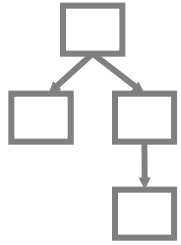
gleich?

Integer? String?

Datenhaltung

Anfragemechanismus

Anwendung



XML-Anfragesprachen



XML-Anfragesprachen: Motivation

- Deskriptiver Zugriff auf XML-Daten
 - ▶ derzeit meist Zugriff entweder auf der Basis von XSL-T
 - ▶ oder auf der Basis von Programmier-APIs (DOM)
- Was macht XML so anders?
 - ▶ hierarchische Strukturierung (im Gegensatz zu relational)
 - ▶ lose Kopplung an Schemata (im Gegensatz zu objektorientiert)
- Es können aber viele Konzepte aus existierenden relationalen und objektorientierten Anfragesprachen übernommen werden!

XML Anfragesprachen: Grundprinzip

■ Selektionsaspekt

- ▶ Formulierung von Anfragebedingungen
- ▶ meist mittels (Baum-)Mustern
- ▶ Ergebnis sind Variablenbindungen

■ Transformationsaspekt

- ▶ Wie soll das Ergebnis aussehen?
- ▶ Erzeugung einer XML-Struktur aus diesen Variablenbindungen

XML Anfragesprachen – Überblick

- XQL
 - ▶ Erweiterung von XPath
 - ▶ keine Berücksichtigung des Transformationsaspektes (soll von XSL-T erledigt werden)
 - ▶ einfach und bereits implementiert (MS SQL, Tamino)
- XML-QL
 - ▶ Entwurf aus der Datenbankwelt mit CONSTRUCT und WHERE-Teil
 - ▶ basiert auf Baummustern und Variablen (wie XQuery)
- XQuery
 - ▶ »offizielle« Standardisierung einer XML-Anfragesprache durch das W3C

Überblick über XQuery (1)

- Ende 1998
 - ▶ begann der Prozeß zur Entwicklung einer W3C-Empfehlung für eine XML-Anfragesprache
- Februar 2001
 - ▶ erster Entwurf auf der Basis von »Quilt«
- April 2001
 - ▶ erste XQuery-Prototypen
 - ▶ Syntaxebene: Microsoft
 - ▶ Algebraebene: Lucent/GMD-IPSI
- Abschluss frühestens Q1 2004
 - ▶ Synchronisation mit anderen Standards

Überblick über XQuery (2)

- XQuery-Spezifikation gliedert sich in die folgenden Teile:
 - ▶ Anfragesprache (*surface syntax, XQuery*)
 - ▶ XML-Repräsentation dieser Anfragesprache (*XQueryX*)
 - ▶ Datenmodell (zusammen mit XPath 2.0)
 - ▶ Formale Semantik («Algebra»)
 - ▶ Standardbibliothek an Funktionen und Operatoren (zusammen mit XPath 2.0 und XSLT 2.0)

XQuery: Übersicht

FOR

Iteration über Knotenmengen
<=> Kreuzprodukt

Variablenbindungen

LET

Bindung von
Einzel-Knoten an Variablen

Variablenbindungen

WHERE

Überprüfen von Bedingungen
über den Variablenbindungen

Variablenbindungen

RETURN

Konstruktion des Ergebnisses

Beispiel für Anfragebearbeitung

<kunden>

FOR \$k in //kunde

LET \$b = count(\$k//bestellung
[bestelldatum>2001-01-01])

WHERE \$b > 0

RETURN

<kunde>

\$k/name,

\$k/vorname,

<anzahl>\$b</anzahl>

</kunde>

</kunden>

\$k	\$b
Hans Mayer	0
Stefanie Bauer	1
Andreas Schmidt	5
Jochen Adam	0

\$k	\$b
Stefanie Bauer	1
Andreas Schmidt	5

```
<kunden>
  <kunde>
    <name>Bauer</name>
    <vorname>Stefanie</vorname>
    <anzahl>1</anzahl>
  </kunde>
  <kunde>
    <name>Schmidt</name> ...
```

XQuery – Beispiele

Joins

<produkte>

```
FOR $b IN document("www.handwerkermarkt.de/angebote.xml")//artikel,  
    $a IN document("www.verbraucherportal.de/produkte.xml")//produkt  
WHERE $b/name = $a/bezeichnung  
RETURN
```

<produkt>

\$b/name,

<preis> \$a/preis/text() </preis> ,

<hersteller> \$a//herstellername/text() </hersteller> ,

<bewertung> \$b/rating/text() </bewertung>

</produkt>

```
SORTBY ($b/name)
```

</produkte>

XQuery – Beispiel

Quantoren

<kunden>

FOR \$k in //kunde

WHERE SOME \$p **IN** \$k//bestellung//produkt **SATISFIES**
\$p/kategorie = "Bohrmaschine"

RETURN

<kunde>

\$k/name,

\$k/vorname,

</kunde>

</kunden>

- analog auch mit **EVERY ... IN ... SATISFIES** für Allquantor

XQuery – Beispiele

Filter

<toc>

```
LET $b := document("bauvorschriften.xml")  
RETURN filter($b,  
    $b//abschnitt | $b//paragraph | $b//paragraph/überschrift)
```

</toc>

- Es bleiben nur die Elemente *abschnitt*, *paragraph* und *überschrift* übrig; der Baum wird »komprimiert«

XQuery – Funktionsdefinitionen

- XQuery erlaubt auch die Definition eigene Funktionen

```
FUNCTION anzahlBestellungen($k, $d)  
    count($k//bestellung[bestelldatum>$d])  
END FUNCTION
```

- können unterschiedliche Datentypen zurückliefern
- können rekursiv sein
 - ▶ damit sehr große Mächtigkeit
 - ▶ Einschränkung der Rekursion wird diskutiert wegen Komplexität der Anfrageauswertung

XQuery: Weitere Möglichkeiten

- Funktionen können definiert werden
- XQuery-Anfragen sind beliebig schachtelbar
- Mengenoperatoren (UNION, INTERSECT, EXCEPT)
- IF ... THEN ... ELSE - Konstrukt

XQuery – Bewertung

- Mächtige und konzeptionell einfach gehaltene XML-Anfragesprache
 - ▶ gute Integration mit anderen XML-Standards, wie z.B. XPath
 - ▶ Schemaunterstützung (u.a. *instanceof*)
- aber:
 - ▶ Kritisiert wird die Überlappung der Transformationsfunktionalität mit XSL-T

Schreibzugriffe auf XML-Daten



Updates für XML

- derzeit:
 - ▶ schreibender Zugriff nur über Programmier-APIs (DOM o.ä.)
- W3C
 - ▶ Notwendigkeit erkannt, aber noch kein Vorschlag
 - ▶ für die zweite Version von XQuery geplant
- MS SQL Server 2000
 - ▶ UpdateGrams
 - ▶ stark an relationalen Datenbanken orientiert
- XML:DB-Initiative
 - ▶ XUpdate
 - ▶ berücksichtigt auch die Ordnung von Elementen

- Vorschlag der XML:DB-Initiative
- Grundprinzip
 - ▶ Selektiere den Kontext der Änderung mittels XPath
 - ▶ spezifiziere die Änderungen auf diesem Knoten
- Methoden zur Änderung
 - ▶ insert-after, insert-before, append
 - ▶ update
 - ▶ remove
 - ▶ u.a.

XUpdate Änderung von Daten

```
<xupdate:modifications version="1.0"
  xmlns:xupdate="http://www.xmldb.org/xupdate">

  <xupdate:update
    select="//kunde[name='Schmidt' AND
      vorname='Andreas']/telefon" >
    0721/9654-732
  </xupdate:update>
</xupdate:modifications>
```

XUpdate: Neue Datensätze anfügen

```
<xupdate:modifications version="1.0"  
  xmlns:xupdate="http://www.xmldb.org/xupdate">
```

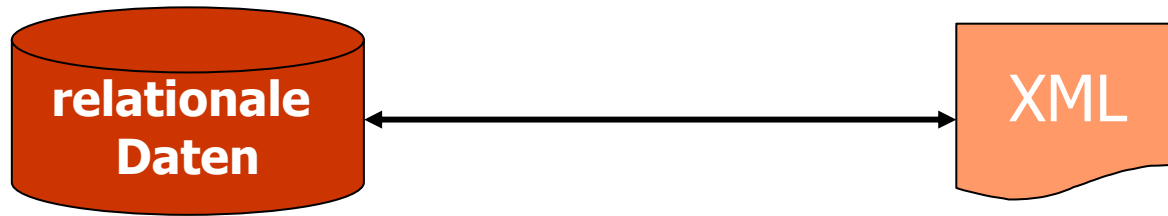
```
<xupdate:append  
  select="//kunden" >  
  <xupdate:element name="kunde">  
    <name>Schmidt</name>  
    <vorname>Andreas</vorname>  
    <telefon>0721/9654-732</telefon>  
  </xupdate:element>  
</xupdate:update>  
</xupdate:modifications>
```

Datenbanken und XML

Die RDBMS-Perspektive



- Relationale DBMS haben sich durchgesetzt
 - ▶ Investitionsschutz
 - ▶ Wiederverwendung durch andere Systeme
 - ▶ Leistungsfähigkeit bewiesen
- Bekannte und mächtige Anfragesprache SQL
- Daten sind bereits vorhanden!
 - ▶ Zusammenspiel mit existierenden Datenbeständen
- Aber:
 - ▶ Mögliche Probleme durch Modellwechsel



Abbildungsaspekte

Mögliche Probleme bei der Abbildung von XML auf relationale Datenbanken

- Schema muss nicht vorab bekannt sein
 - ▶ Optimierungsprobleme bei Speicherung
 - ▶ In realen Systemen wird diese Flexibilität oft fallengelassen (Schema muss also bekannt sein)
- Verschachtelung von Elementen
 - ▶ Verschachtelung beliebiger tiefe
 - ▶ Zerlegung in relationalen ("shredding") und Verknüpfung über Fremdschlüssel
- Wiederholung von Elementen
 - ▶ Beliebige Wiederholung von Elementen möglich
 - ▶ Spalten in Relationen einmalig

Abbildungsaspekte (2)

- Alternativen erlaubt
 - ▶ Elemente eines Elementtyps können unterschiedliche Kinderelemente enthalten
- Optionale Elemente
 - ▶ Kann auf NULL Werte abgebildet werden (ist aber semantisch nicht ganz das gleiche)
- Mixed Content
 - ▶ Abwechselnd Zeichen und Kinderelemente
 - ▶ Widerspruch zur ersten Normalform im relationalen Modell
- Attributwerte

Abbildungsaspekte (3)

- Elementreihenfolge
 - ▶ Elementare Eigenschaft von XML
 - ▶ Wird bei Validierung berücksichtigt
- Datentypen
 - ▶ In DTD nicht vorhanden (nur in XML Schema)
- Nullwerte
 - ▶ Abbildung von nicht vorhandenen Elementen auf NULL-Werte
 - ▶ Unterschied zwischen "" und null
- Binärdaten
- Processing Instructions

Microsoft SQL Server

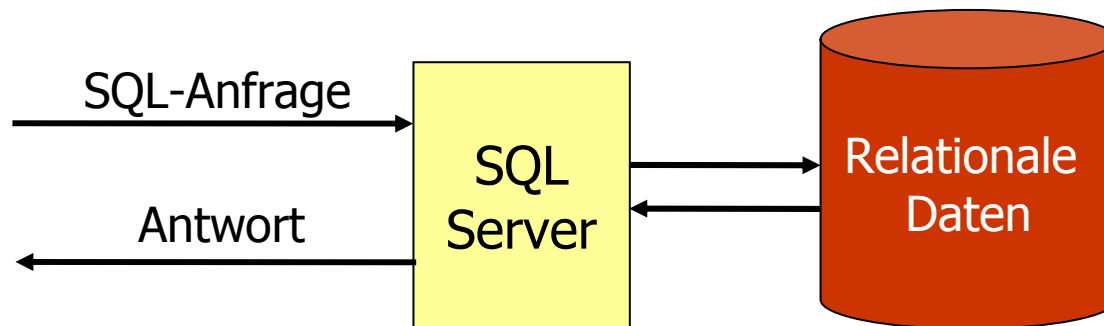


Microsoft SQL 2000 Server und XML

- XML-Tags für alle Erweiterungen im Namespace `xml-sql` definiert.

```
<root xmlns:sql="urn:schemas-microsoft-com:xml-sql">
```

- Ursprüngliche Vorgehensweise der SQL-Anfrage ohne Verwendung von XML-Techniken :



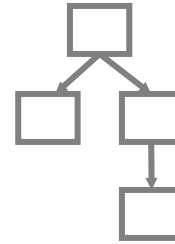
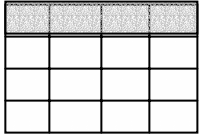
Vorstellung der Werkzeuge

- Erzeugen von XML aus Relationen
 - ▶ FOR XML
- Verändern einer Datenbank mit XML-Techniken:
 - ▶ Update Grams
- XML-basierte Suche
 - ▶ XPath über Annotierte Schemata
- SQL-Anfragen auf XML-Dokumente
 - ▶ OpenXML zur Herstellung einer relationalen Sicht eines XML-Dokuments. Darauf ist Standard-SQL-Anfrage möglich.

Datenhaltung

Anfragemechanismus

Anwendung

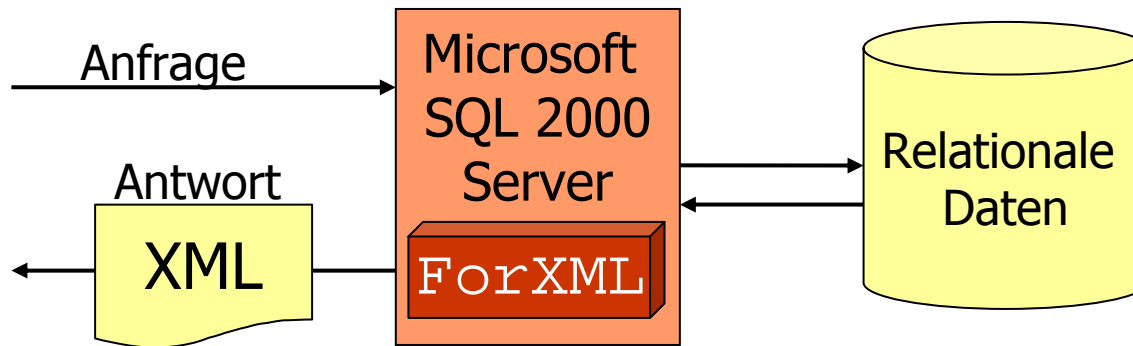


FOR XML



Relationale Daten nach XML

- Verwendung der SQL-Erweiterung For XML



- Transformierungsmodi von FOR XML:

- RAW
- AUTO
- EXPLICIT

■ SQL-Anfrage:

```
SELECT Artikel.katalogelementId, Artikel.name
       Bestellung.beschreibung
FROM Artikel
INNER JOIN Bestellung
ON Artikel.katalogelementId = Bestellung.
   katalogelementId
WHERE Bestellung.katalogelementId = '47'
```

- ▶ Durch Anwendung von `FOR XML` und den Transformierungsmodi erfolgt Generierung der Ergebnisrelation zu XML-Dokument

Transformierungsmodi RAW & AUTO

■ RAW

- Ausgabebetupel als <row> Tag
- Spaltenname als Attributname
- Spaltenwert als Attributwert
 - `<row katalogelementId =î47î name=îSack Zementî beschreibung=îAchtung! schnell bindendî/>`

■ AUTO

- Ergebnisrelation als verschachtelte XML
- Tabellename aus FROM-Anweisung als Element ausgegeben
- Gemäß in SELECT-Anweisung ausgewählte Ergebnisspalten werden Elementattributen zugeordnet

Elementverschachtelung

- ▶ Attributnamen gemäß Reihenfolge der Spaltendeklaration in SELECT-Anweisung:

```
<artikel katalogelementId =î47î name=îSack Zementî>
```

Tabellenname

Spaltennamen

- ▶ Durch Einsatz von ELEMENTS -Direktive hinter FOR XML, Spalten als Kindelemente:

```
<artikel>
```

```
<katalogelementId>47</katalogelementId>
```

```
</artikel>
```

Spaltennamen

Aufruf von FOR XML

■ Direkte Anfrage über url

- ▶ HTTP-get-Aufruf mit SQL-Anfrage als CGI-Parameter:
 - mit `root` Wurzelement angeben
 - Formatierung der Ergebnisdokument über XSL-Stylesheet mit `xsl-` Parameter
 - `contenttype` zur MIME-Typ Festlegung. Ohne Angabe des Typs, `text/html` als Defaultwert

▶ Beispiel:

```
http://IISServer/vroot?sql=SELECT+name,beschreibung+  
FROM+Artikel+FOR+XML+AUTO&xsl=katalog.xsl&root=kata  
alog
```

xsl-stylesheet



Aufruf von FOR XML über Schablonen

- XML-Dokument mit definierter SQL-Anfrage
- Einbettung in XML über `<sql:query>` -Tag.
- Zusätzlicher Parameterübergabe möglich
 - ▶ Angabe der Elemente `<sql:header>` und `<sql:param>`
- Aufruf der *templates* über url

Update-Grams

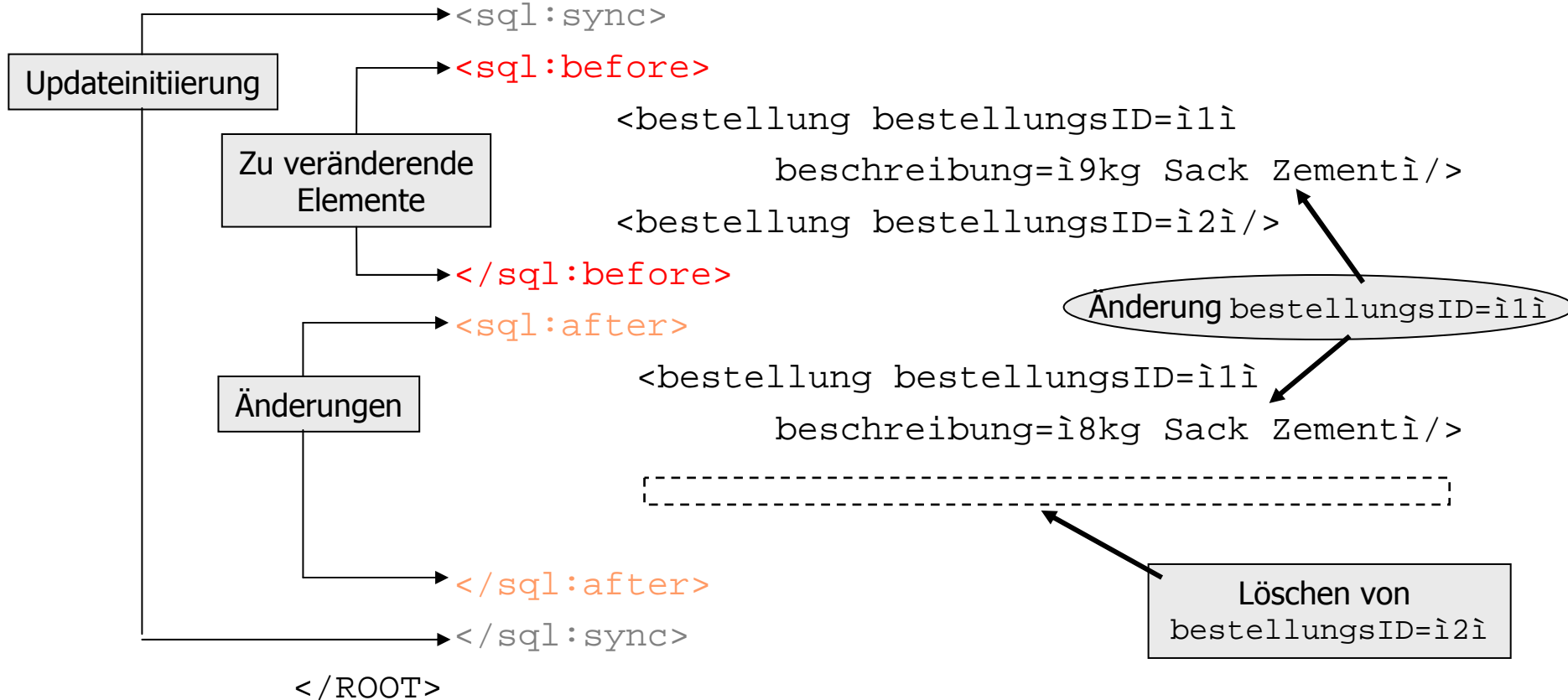


Update-Grams-Aktualisierung mit XML

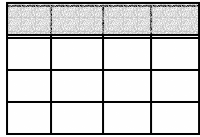
- Spezifizierung des zu ändernden Elements in Baumschablone (XML-Struktur) mit Änderungsbefehlen:
 - ▶ `<sql:sync>` gibt die zu ändernden Elemente an
 - ▶ `<sql:before>` beschreibt aktuelle Eigenschaft und Inhalt
 - ▶ `<sql:after>` beschreibt, wie Element nach Änderung auszusehen haben
- ⇒ Vorher-Nachher-Abbildung des spezifizierten Elements

Beispielmodifikation

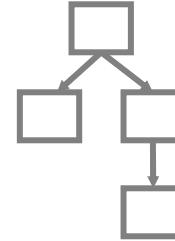
```
<ROOT xmlns:sql=urn:schemas-microsoft-com:xml-sql>
```



Datenhaltung Anfragemechanismus Anwendung



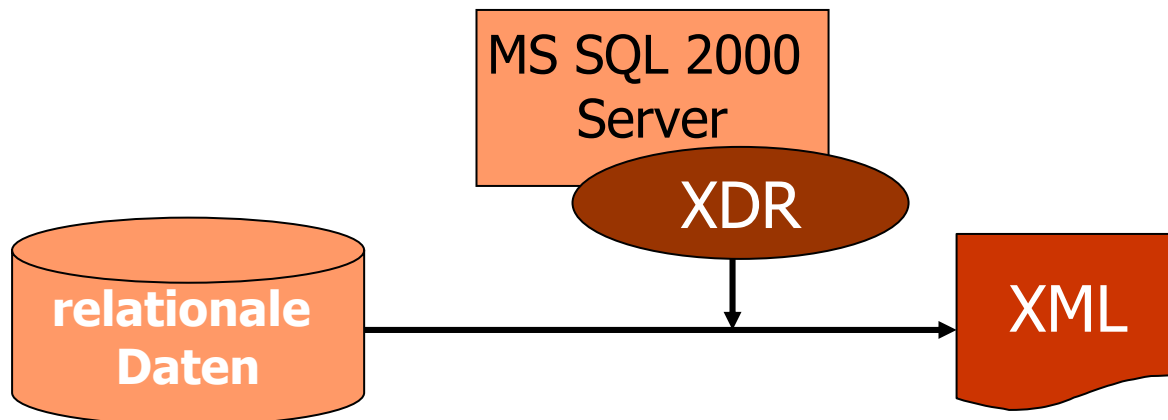
← XML Anfragen →



Schemaannotationen

Mit XPath auf SQL Server zugreifen

- Bei ausschließlicher Verwendung von XML, Datenbankzugriff über `XPath` auf Relationen
- Virtuelle XML-Sicht der relationalen DB erforderlich
 - ⇒ Zuordnung von Tabellen und Spalten nach Elemente und Attribute
 - ⇒ Zuordnung in annotierten Schemata Dokumenten gespeichert



Schemaannotation

- MS SQL 2000 Server führt diese Erweiterungen (Annotationen) in Schemadefinition ein
- Bidirektionale Zuordnung von Elementnamen zu Tabellennamen
- Schemadefinition
 - ▶ Zunächst in XML Data Reduced (XDR)
 - ▶ neuerdings auch XML Schema
- Definierte Annotationen:
 - ▶ `sql:relation` Abbildung von Elementen auf Relationen
 - ▶ `sql:field` Abbildung von Elementen und Attributen auf Spalten
 - ▶ `sql:overflow-field` zur Speicherung überschüssiger Daten aus dem XML-Quelldokument in separater Spalte

XDR-Mapping Schema

```
<?xml version='1.0' encoding='UTF-8' ?>
```

```
<Schema xmlns="urn:schemas-microsoft-com:xml-data" ...>
```

```
  xmlns:dt="urn:schemas-microsoft-com:datatypes"
```

```
  xmlns:sql="urn:schemas-microsoft-com:xml-sql">
```

```
<ElementType name="title" sql:relation="titles">
```

```
  <AttributeType name="title" dt:type="string"/>
```

```
  <AttributeType name="type" dt:type="string"/>
```

```
  <attribute type="title" sql:field="title"/>
```

```
  <attribute type="type" sql:field="type"/>
```

```
</ElementType>
```

```
<ElementType name="author" order="many" sql:relation="authors">
```

```
  <element type="title" maxOccurs="*" >
```

```
    <sql:relationship key="au_id" key-relation="authors" foreign-key="au_id" foreign-  
      relation="titleauthor" />
```

```
    <sql:relationship key="title_id" key-relation="titleauthor" foreign-key="title_id" foreign-  
      relation="titles" />
```

```
  </element> <AttributeType name="au_lname" dt:type="string"/>
```

```
  <AttributeType name="au_fname" dt:type="string"/> <attribute type="au_lname"/>
```

```
  <attribute type="au_fname"/>
```

```
</ElementType>
```

```
</Schema>
```

Angabe des Tabellennamens

Angabe der
Beziehung
Spalte-Attribut

Beispielanfrage

- Schemaannotation erzeugter XML-Sicht relationaler Daten
- MS SQL 2000 Server unterstützt eine Teilmenge von XPath zur Anfrage
- XPath-Anfragen über url oder Schablonen
 - ▶ Angabe des Schemas, auf dem Anfrage durchgeführt wird
 - ▶ Beispiel- Schablone mit XPath-Anfrage:

```
<?xml version="1.0" ?>
```

```
<root>
```

```
<sql:xpath-query mapping-schema="authorschema.xdr"  
  xmlns:sql="urn:schemas-microsoft-com:xml-sql">  
  /author
```

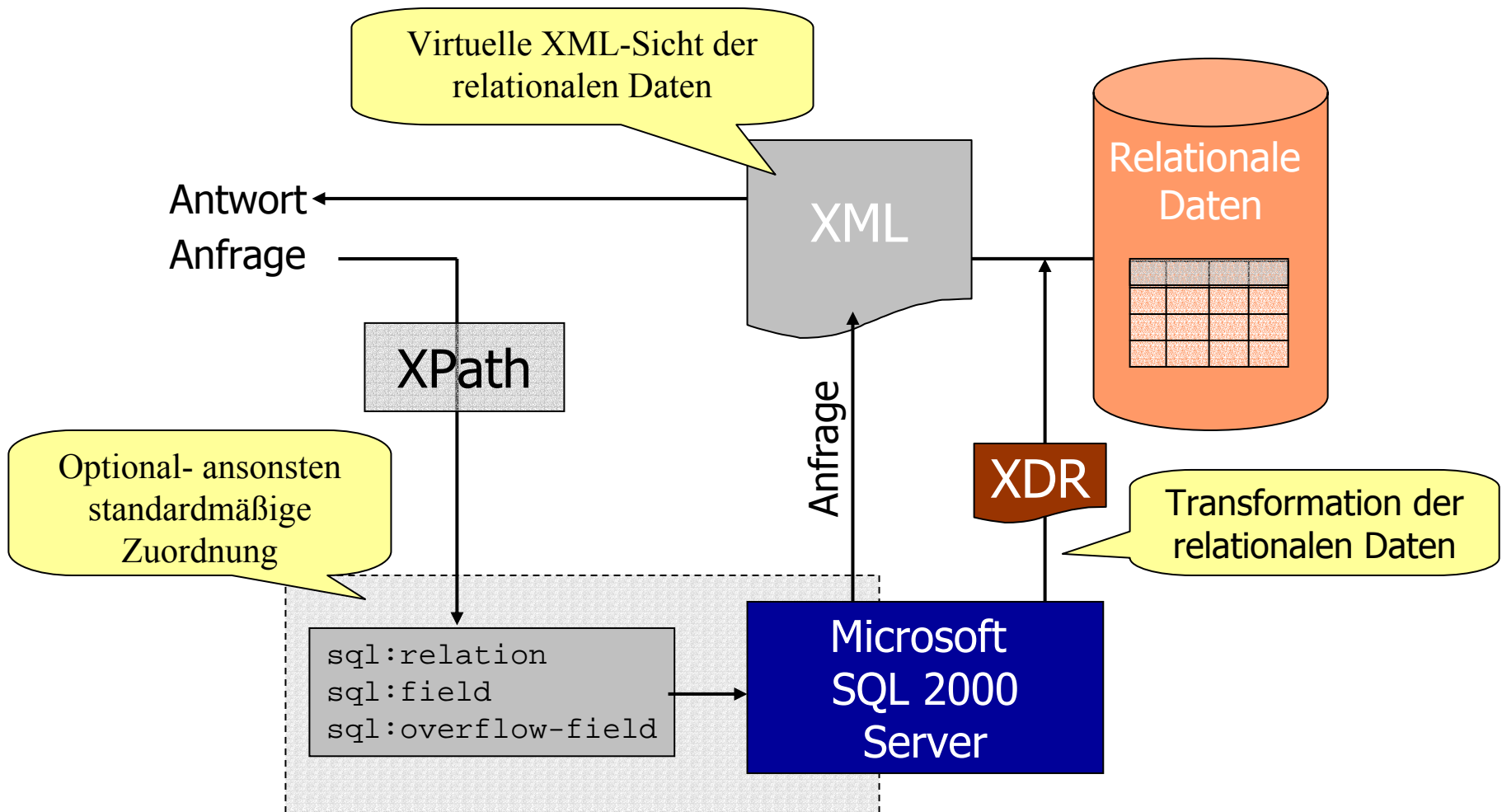
```
</sql:xpath-query>
```

```
</root>
```

Angabe des Pfades

Angabe der XDR-Datei

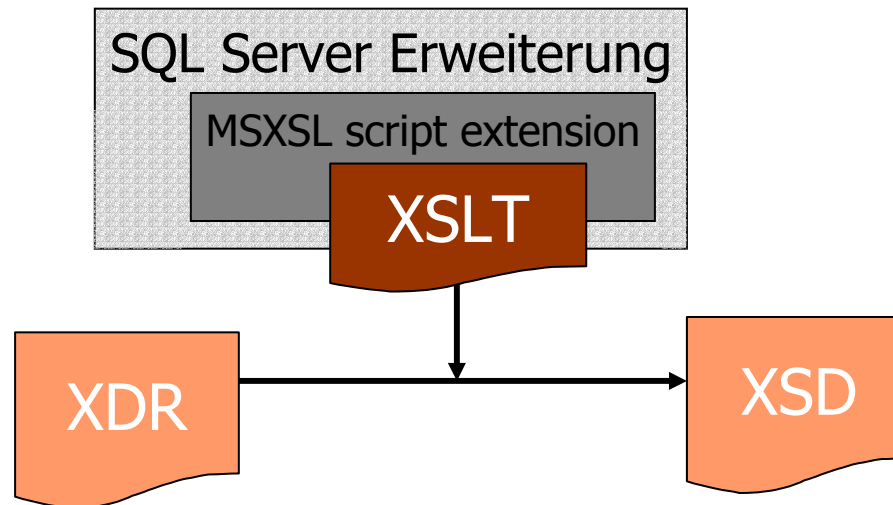
XPath über XDR



XSD anstelle XDR


■ Etablierung von XML-Schema *XSD*:

- Javabasiertes Konvertierungstool XDRtoXSD von alphaWorks (IBM)
 - Abfangen von Errors
 - Erkennen von Semantiklücken in XDR für die XSD-Konvertierung
- Transformation von XDR zu XSD mit Hilfe von XSLT-sheets:
 - (Benötigt wird hierzu SQL Server 2000 Web Release 2 (WR2) Beta2



XSD-Mapping Schema [1-2]

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
<element name="author" sql:relation="authors">
  <complexType>
    <sequence>
      <element name="title" sql:relation="titles">
        <annotation>
          <appinfo>
            <sql:relationship parent-key="au_id"
              parent="authors"
              child-key="au_id" child="titleauthor"/>
            <sql:relationship parent-key="title_id"
              parent="titleauthor"
              child-key="title_id" child="titles" />
          </appinfo>
        </annotation>
      </element>
    </sequence>
  </complexType>
</element>
```



Angabe des Tabellennamens

XSD-Mapping Schema [2-2]

```
<complexType>
  <attribute name="title" type="string"/>
  <attribute name="type" type="string"/>
</complexType>
</element>
</sequence>
<attribute name="au_lname" type="string"/>
<attribute name="au_fname" type="string"/>
</complexType>
</element>
</schema>
```

XPath-Anfrage

```
<?xml version="1.0" ?>
```

```
<root>
```

```
  <sql:xpath-query mapping-schema="authorschema.xsd" ←  
    xmlns:sql="urn:schemas-microsoft-com:xml-sql">
```

```
    /author ←
```

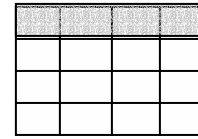
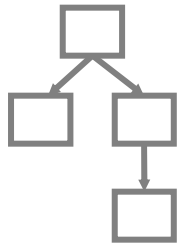
```
  </sql:xpath-query>
```

```
</root>
```

Angabe des Pfades

Angabe der XSD-Datei

Datenhaltung Anfragemechanismus Anwendung



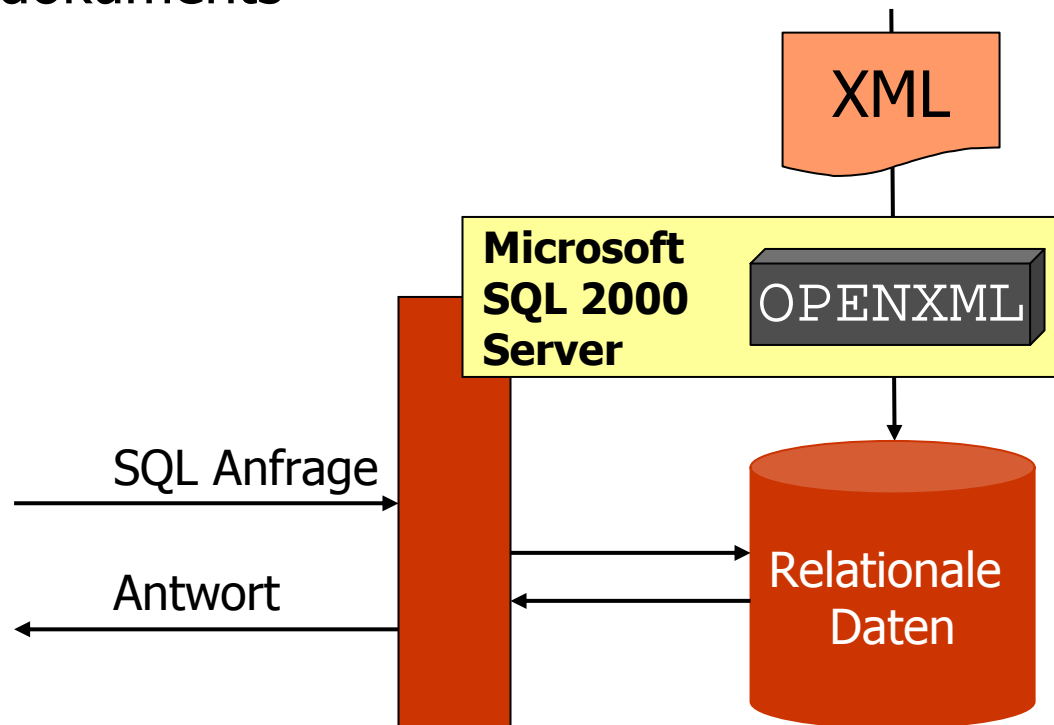
Open XML



Mit SQL auf XML-Dokumente zugreifen



- Zugriff auf XML-Dokumente mittels SQL
- Erzeugung einer relationalen Sicht eines XML-Quelldokuments



Beispiel XML-Quelldokument

```
DECLARE @idoc int
DECLARE @doc varchar(1000)
SET @doc =ë
<ROOT>
  <person marktteilnehmerID = ì6ì nachname =ìhuberì>
    <bestellung bestelungsID = ì1234ì>
      <artikel/>
      <artikel/>
    </person>

  <person marktteilnehmerID = ì7ì nachname =ìmeierì>
    <bestellung bestelungsID = ì1235ì>
      </artikel>
    </person>
</ROOT>ë
```

Erzeugung einer relationalen Sicht

- über Operationsbefehl `sp_xml_preparedocument` wird relationale Sicht des XML-Quelldokuments erstellt

```
EXEC sp_xml_preparedocument @idoc OUTPUT, @doc
```

- ⇒ Auf erzeugter Repräsentation Standard-SQL-Abfragen mit Einfüge-, Aktualisierungs- und Löschoperationen möglich

```
SELECT *  
FROM OPENXML (@idoc, '/root/person',1)  
WITH ( marktteilnehmerID varchar(10), nachname  
      varchar (20) )
```

Ergebnisrelation

marktteilnehmerID	nachname
6	huber
7	meier

- mit `sp_xml_removedocument` kann das XML-Dokument aus Arbeitsspeicher wieder entfernt werden

- Datenbanken und XML
 - ▶ Zwei unterschiedliche Welten müssen zusammenspielen
 - ▶ Das Beste aus jeder Welt nutzen
 - ▶ Kaum Standards
 - Anbietergesteuert (Microsoft, Oracle, IBM)
- Nicht behandelt, aber interessant:
 - ▶ Native XML-Datenbanken
 - Tamino, Xindice, Birdstep, Natix,....
 - ▶ Oracle, IBM
- Nicht behandelt, da separate Vorlesung
 - ▶ XML Schema

- Wassilios Kazakos, Andreas Schmidt, Peter Tomczyk:
Datenbanken und XML, Springer, April 2002
- Alternativ: Literaturverzeichnis zum Thema:
 - ▶ www.datenbanken-und-xml.de