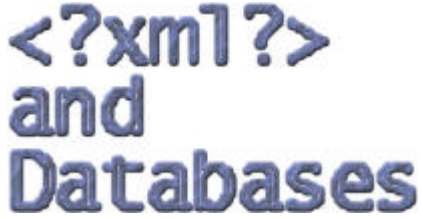
XML und Datenbanken

Meike Klettke, Holger Meyer
Universität Rostock
Fachbereich Informatik
meike@informatik.uni-rostock.de
hme@informatik.uni-rostock.de



www.xml-und-datenbanken.de

Grundlagen XML

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- XML - E**X**tensible **M**arkup **L**anguage

  ↓ Markup-Sprache

  (Daten und Informationen über die Daten in einem Dokument)

- durch das World Wide Web Consortium (W3C) entwickelt
- gut lesbar
- häufig eingesetztes Austauschformat

Grundlagen XML

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- „XML is the ASCII of the 21th century.“
- „XML is the ASCII of the Web“

  Henry Thompson (1999)

- Warum dann diese Popularität ?

Zeitliche Entwicklung XML

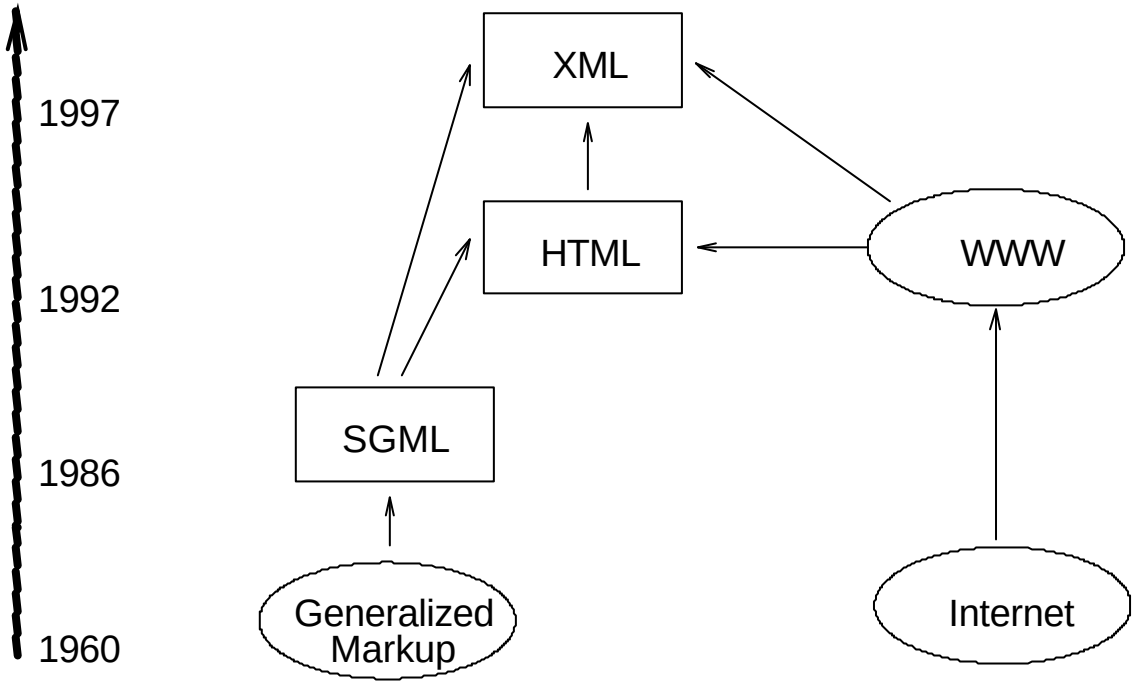
Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Aus Neil Bradley: The XML companion



Einsatzgebiete von XML

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Electronic Data Interchange (EDI)
- Austausch von Daten zwischen verschiedenen Applikationen
- Darstellung von Volltextdokumenten mit Markup
- ...



Dauerhafte und sichere Speicherung von XML-Dokumente kann dabei erforderlich sein.

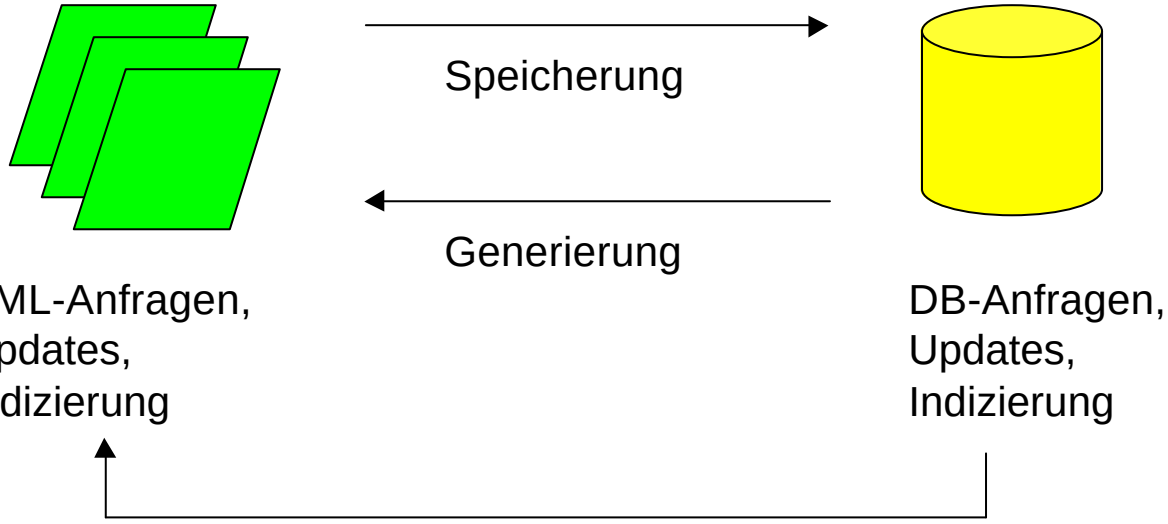
XML und Datenbanken

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



XML –
Dokumentformat

Datenbanken –
Systeme zur Speicherung
von großen Datenmengen

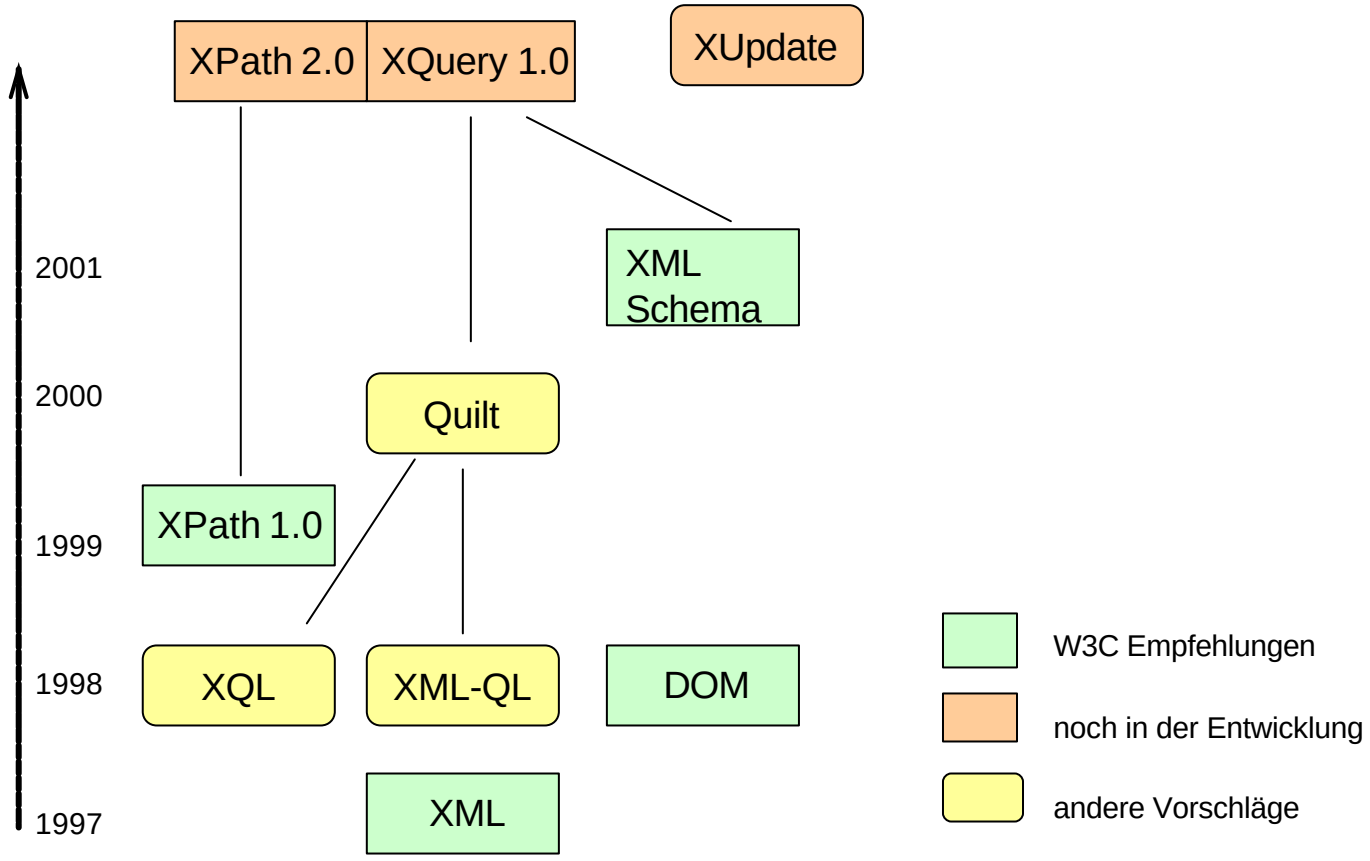
Zeitliche Entwicklung XML /2

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



Anwendungsszenarien „XML und Datenbanken“

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Vorstellung von verschiedenen Anwendungsszenarien zur Einführung und Motivation

- Betrachtung der bestehenden Teilaufgaben
- Einordnung der Teilaufgaben in das Tutorial

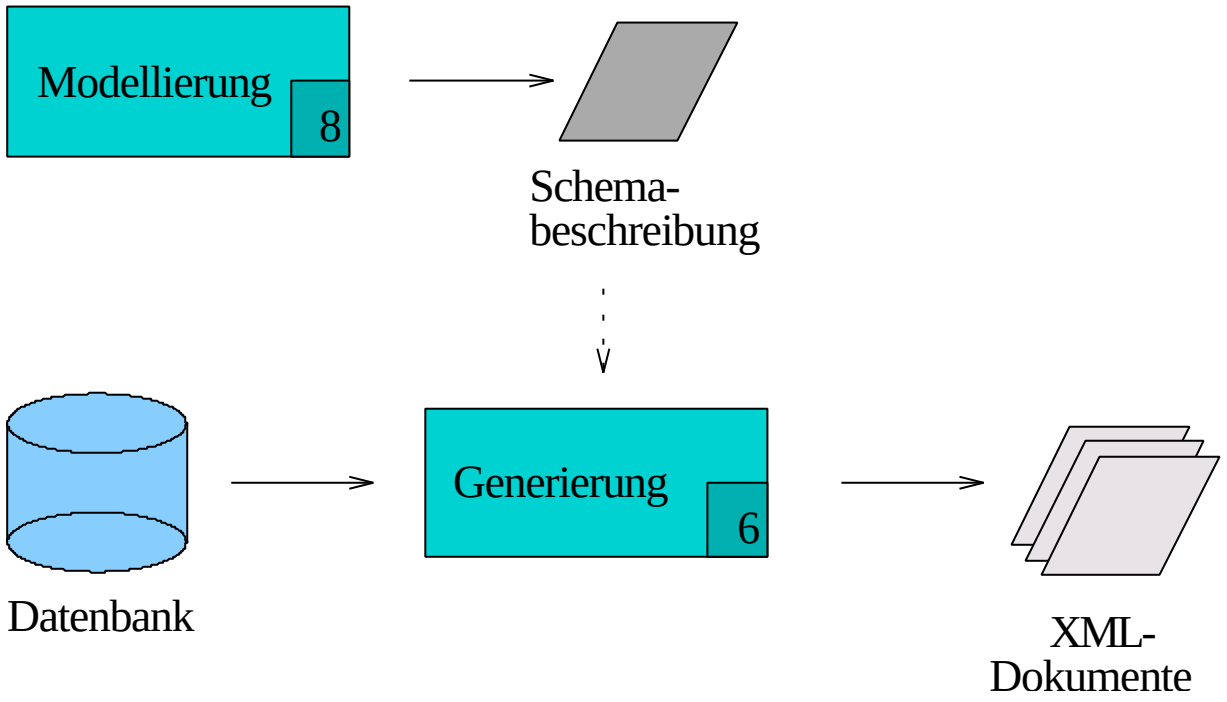
Generierung von XML-Dokumenten

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



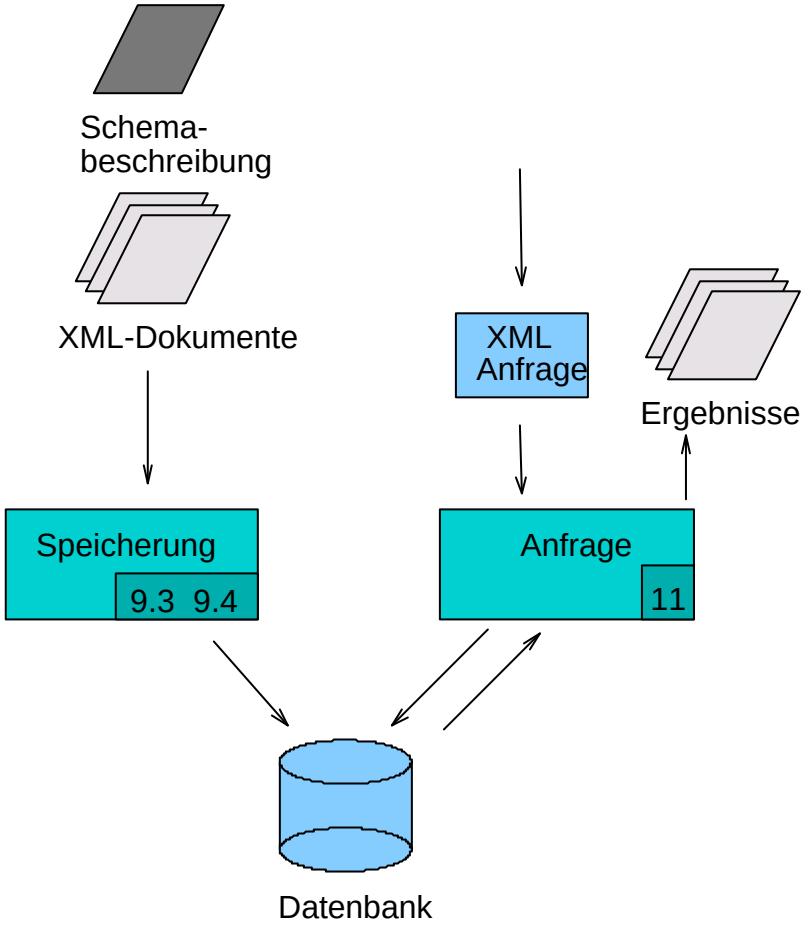
Speicherung datenzentrierter XML-Dokumente

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



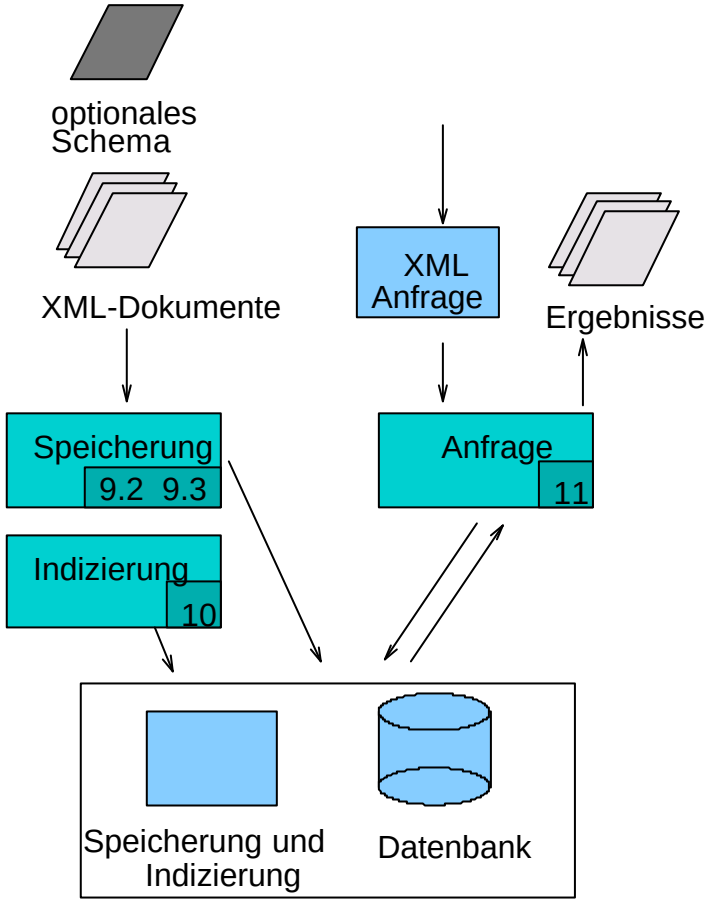
Speicherung dokumentzentrierter XML-Dokumente

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



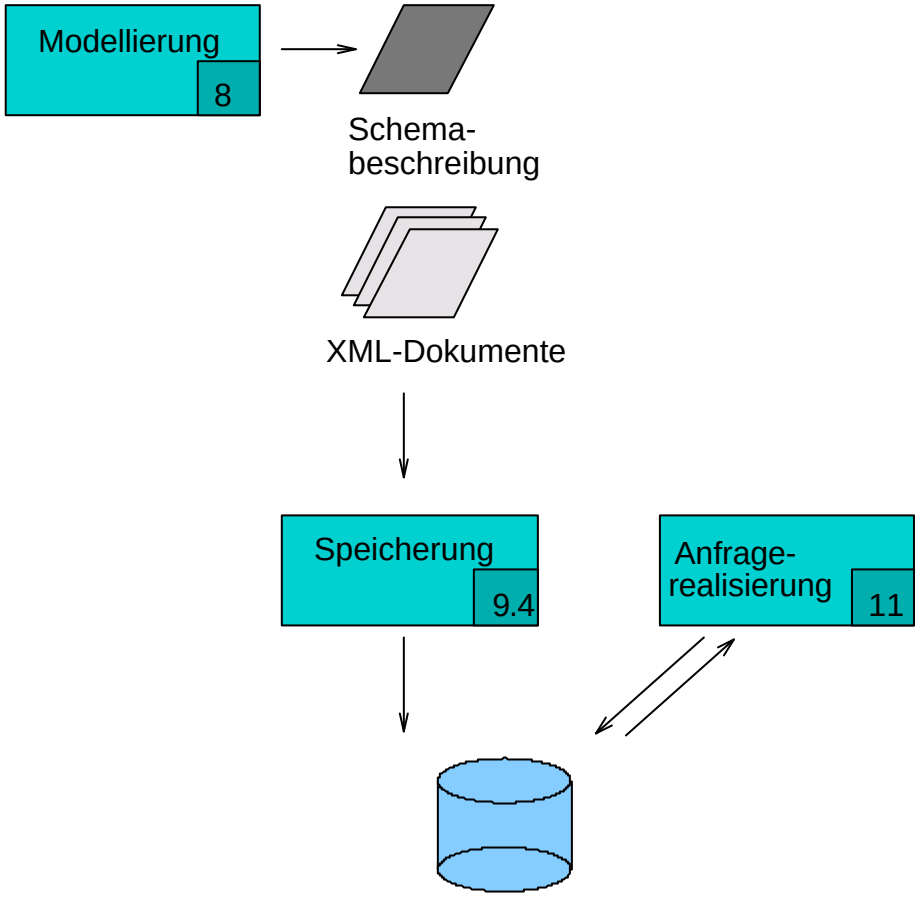
Speicherung von XML-Dokumenten in bestehende Datenbanken

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



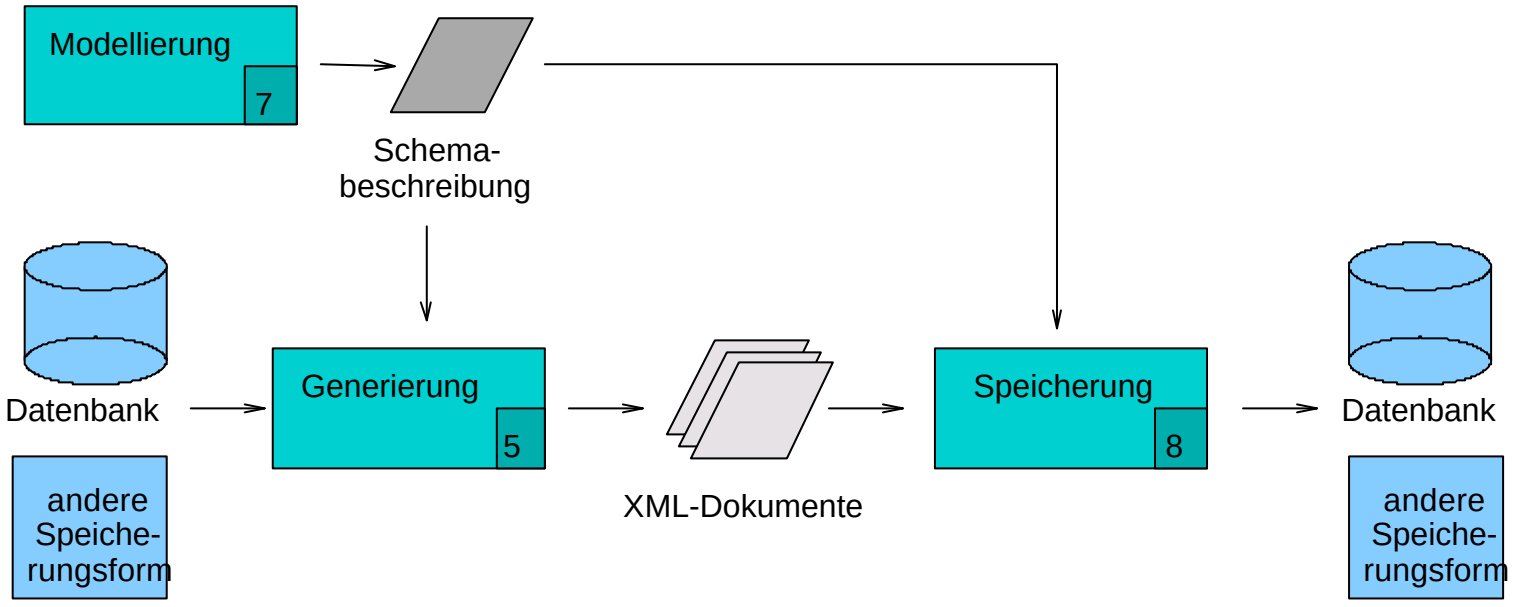
Electronic Data Interchange (EDI)

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



Inhalt des Tutorials

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Einführung – XML (XML, DTD, XML Schema, ...)
- Architektur von XML-Datenbanken

- XML-Anfragesprachen (XPath, XQuery)

- Speicherung von XML-Dokumenten (in Datenbanken)
- Generierung von XML-Dokumenten aus Datenbanken

- Indizierung
- Systeme

Weiterführende Literatur

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

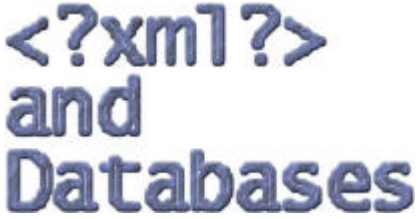
Indizierung
Systeme

- www.xml-und-datenbanken.de (Meike Klettke, Holger Meyer)
- www.rbourret.com (Ronald Bourret)
- www.oasis-open.com/cover (Robin Cover)

- Meike Klettke, Holger Meyer: *XML und Datenbanken*, dPunkt-Verlag, erscheint Oktober 2002

XML und Datenbanken
Einführung in XML

Meike Klettke
Universität Rostock
Fachbereich Informatik
meike@informatik.uni-rostock.de



www.xml-und-datenbanken.de

Inhalt

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- XML
  - Elemente, Attribute, Vergleich Elemente und Attribute, ID/IDREF
- DTDs
  - Element-, Attribut- und Entity-Deklarationen
- XML Schema
  - Einfache und komplexe Typen, Attribut- und Elementdeklaration
- Vergleich von DTD und XML Schema
- Weiterführende Literatur

Beispiele für XML-Dokumente

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

```
<order>
  <customer>Meyer</customer>
  <position>
    <isbn>1-234-56789-0</isbn>
    <number>2</number>
    <price currency=´Euro´>30.00</price>
  </position>
</order>
```

Elemente und Attribute
hierarchische Schachtelung
Sequenz

```
<content>
XML builds on the principles of two
existing languages, <emph>HTML
</emph> and <emph>SGML
</emph> to create a simple
mechanism ..  The generalized
markup concept .. </content>
```

Markup in Volltexten
Mixed Content Elemente

Syntax: Elemente und Attribute

Einleitung, Anwendungen
XML Architektur

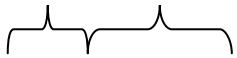
XML-Anfragen

XML-Speicherung
XML-Generierung

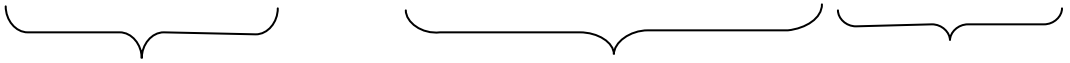
Indizierung
Systeme

XML-Dokument

Attributname Attributwert



<hotel id=´h0001´>Strand Hotel Hübner</hotel>



Start-Tag Element Ende-Tag

DTD

<!ELEMENT hotel (#PCDATA)>

<!ATTLIST hotel id ID #REQUIRED>

Syntax: Elemente und Attribute / 2

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

XML-Dokument

```
<hotel id=´h0001´>
  <name>Strand Hotel Hübner</name>
  <kategorie>4</kategorie>
</hotel>
```

Start-Tag

Element

Ende-Tag

DTD

```
<!ELEMENT hotel (name, kategorie)>
<!ATTLIST hotel id ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT kategorie (#PCDATA)>
```

Definition von Elementen in einer DTD

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- **Sequenz** (A , B)
  A und B müssen in der angegebene Reihenfolge im Dokument auftreten
- **Alternative** (A | B)
  entweder A oder B treten im Dokumente auf
- **Wiederholung**
  A? - 0..1 Mal
  A+ - 1..n Mal
  A* - 0..n Mal
- **Mixed Content** (#PCDATA | A | B)*
  A, B oder beliebiger Text treten im Dokument auf
- **ANY**

```
<!ELEMENT hotel
      (name, adresse)>
<!ELEMENT name
      (#PCDATA)>
<!ELEMENT adresse
      (plz, ort,
      (strasse, nummer?) |
      postfach)>
<!ELEMENT beschreibung
    (#PCDATA | ausstattung
     | gastronomie)*>
```

Definition von Attributen

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Attribute haben

- einen Namen
- einen Typ

  (CDATA, ID, IDREF/IDREFs, ENTITY/ENTITYS, NMTOKEN/NMTOKENS
  oder (wert1|wert2|...)
  Aufzählung möglicher Werte
- eine Angabe, ob das Attribut auftreten muss

  (#REQUIRED, #IMPLIED oder #FIXED)
- einen optionalen Defaultwert

  (bei #FIXED aber erforderlich)

```
<!ATTLIST preis waehrung
   CDATA #REQUIRED ´Euro´>
<!ATTLIST projekt id
   ID #REQUIRED>
<!ATTLIST person projekt
   IDREF #REQUIRED>
<!ATTLIST plz xml-sqltype
   CDATA #FIXED ´INTEGER´>
```

ID / IDREF

Attribute können als ID/ IDREF / IDREFS definiert sein

Werte innerhalb eines Dokumentes eindeutig

```
<projekt bearbeiter=´p0001´ />
<projekt bearbeiter=´a0001´ />
...
<person id=´p0001´>
   <name>Meyer</name>
</person>
...
<abteilung id=´a0001´>
```

```
<!ELEMENT projekt (EMPTY)>
<!ATTLIST projekt bearbeiter IDREF
                  #REQUIRED>
<!ELEMENT person (EMPTY)>
<!ATTLIST person id ID
                  #REQUIRED>
...
<!ELEMENT abteilung (EMPTY)>
<!ATTLIST abteilung id  ID
                   #REQUIRED>
```

Verwendung von Elementen oder Attributen

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Elemente	Attribute
Quantifizierer 1/?/+/*	#REQUIRED/#IMPLIED
Alternativen	keine Alternativen
keine Defaultwerte	Defaultwerte
keine Aufzählungstypen	Aufzählungstypen
geeignet für komplexere Informationen	geeignet für strukturierte und wenig detaillierte Informationen
Einsatz für Darstellung von Daten	Einsatz für Darstellung von Metadaten

XML - Weitere Bestandteile / 1

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Entities

- zur Definition von Dokumentbestandteilen

```
<!ENTITY hansesailtermin ´7.-10. August 2003´>
<!ENTITY hansesailteilnehmer ´200 Traditionssegler
und Museumsschiffe´>
```

```
<veranstaltung>
    Die Hansesail findet in diesem Jahr vom
     &hansesailtermin; statt.
    Es werden &hansesailteilnehmer; erwartet.
</veranstaltung>
```

XML – Weitere Bestandteile / 2

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Entities

- zur Verwendung spezieller Zeichen
  - < für <
  - > für >
  - & für &
  - ' für ´
  - " für ´´
- zur Einbindung anderer Dateiformate
  - Bilder und ähnliches
- Zur Verwendung in DTDs
  - <!ENTITY % adressdef ´(ort,plz,strasse,nummer)´>
  - <!ELEMENT adresse %adressdef;>
- Processing Instructions
- Kommentare

Eigenschaften von XML-Dokumenten

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Wohlgeformtheit
  - Dokument syntaktisch korrekt?
- Gültigkeit
  - Wohlgeformtheit und entspricht das XML-Dokument der zugeordneten DTD?

Ein weiteres Beispiel

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hotel SYSTEM "hotel_dt.dtd">
<hotel id="id001"
      url="http://www.hotel-
huebner.de">
    <name>Strand Hotel Huebner</name>
    <adresse>
        <plz>18119</plz>
        <ort>Rostock-Warnemuende</ort>
        ...
    </adresse>
    <hausbeschreibung> Direkt an der
Promenade von Warnemuende befindet sich
das Strand-Hotel Huebner mit Blick auf
Leuchtturm, Hafeneinfahrt und Strand.
</hausbeschreibung>
    <preise waehrung="DM">
        <einzelzimmer>ab 198,-</></>
...
</hotel>
```

```
<!-- Hotel DTD-->
<!ELEMENT hotel (name, kategorie?,
adresse, hausbeschreibung, preise*>
<!ATTLIST hotel id ID #REQUIRED
            url CDATA #REQUIRED>
<!ELEMENT  name (#PCDATA)>
<!ELEMENT  kategorie (#PCDATA)>
<!ELEMENT  adresse (plz, ort,
     strasse, hausnummer, telefon,
     fax?, e-mail?)>
<!ELEMENT  plz (#PCDATA)>
...
<!ELEMENT hausbeschreibung
(#PCDATA)>
<!ELEMENT preise (#PCDATA |
einzelzimmer | doppelzimmer |
appartment)*>
<!ATTLIST preise waehrung CDATA
#REQUIRED>
<!ELEMENT einzelzimmer (#PCDATA)>
...
```

Klassifikation von XML Dokumenten

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Datenzentrierte Dokumente

  strukturiert, regulär

  Beispiele: Produktkataloge, Bestellungen, Rechnungen

- Dokumentzentrierte Dokumente

  unstrukturiert, irregulär

  Beispiele: wissenschaftliche Artikel, Bücher, E-Mails, Webseiten

- Semistrukturierte Dokumente

  datenzentrierte und dokumentzentrierte Anteile

  Beispiele: Veröffentlichungen, Amazon

```
<order>
  <customer>Meyer</customer>
  <position>
    <isbn>1-234-56789-0</isbn>
    <number>2</number>
    <price currency=„Euro“>30.00</price>
  </position>
</order>
```

```
<content>
XML builds on the principles of two existing
 languages, <emph>HTML</emph> and
<emph>SGML</emph> to create a simple
mechanism ..
The generalized markup concept ..
</content>
```

```
<book>
  <author>Neil Bradley</author>
  <title>XML companion</title>
  <isbn>1-234-56789-0</isbn>
  <content>
    XML builds on the principles of two existing
    languages, <emph>HTML</emph> and ..
  </content>
</book>
```

Darstellung von Strukturinformationen

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Vorteile:

- Explizitmachen der Struktur
  (für Anwendungen und Applikationen)
- Erkennung und dadurch Vermeidung von Fehlern in den XML-Dokumenten

Schemata für XML-Dokumente

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- DTDs
  - (aus SGML, einfach, kompakt, gut lesbar, aber keine XML-Syntax)
- XML Schema
  - (wesentlich mehr Darstellungsmöglichkeiten, Datentypen, exakte Kardinalitäten, ...)

Document Type Definitions /1

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Deklaration von Elementen und Attributen
- Elementen können aus
  - einer Sequenz von Elementen (A,B),
  - einer Alternative von Elementen (A|B)
  - mit einer angegebenen Wiederholung (A?, A+, A*), möglich ist: Minimum 0 oder 1, Maximum 1 oder n sowie
  - beliebigen Kombinationen aus den oberen Möglichkeiten bestehen.
- Den Elementen können Attribute zugeordnet sein.

Document Type Definitions /2

Attribute haben

- einen Attributnamen,
- eine Typangabe (CDATA, NMTOKEN, NMTOKENS, ID, IDREF, IDREFS, ENTITY oder eine Aufzählung),
- eine Angabe, ob Attribute immer auftreten müssen oder optional sind (#REQUIRED, #IMPLIED, #FIXED) sowie
- einen Defaultwert.

XML Schema /1

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- seit Mai 2001 als Empfehlung des W3C verabschiedet
- enthält drei Teile:
  - Einführung,
  - Strukturen sowie
  - Datentypen
- XML Schema enthält wesentlich umfangreichere Darstellungsmöglichkeiten als DTDs

XML Schema /2

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- vielfältige vordefinierte Datentypen,
- Definition eigener Datentypen ist möglich,
- umfangreiche Darstellungsmöglichkeiten,
- Erweiterbarkeit,
- Darstellung von Integritätsbedingungen sowie
- XML-Syntax.

XML Schema /3

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- enthält Definitionen und Deklarationen
- mit den Definitionen werden neue Typen erzeugt
  - einfache Typdefinitionen,
  - komplexe Typdefinitionen,
- Deklarationen beschreiben das Inhaltsmodell von Elementen und Attributen
  - Attributdeklarationen
  - Elementdeklarationen

Einfache Typen

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Einfache Typdefinitionen

- vordefiniert oder benutzerdefiniert
- Vielzahl von Standardtypen (string, boolean, float, integer, decimal, time, date, u.v.m.) ist vordefiniert, vgl. Abbildung
- sind die Basis aller anderen Typen.

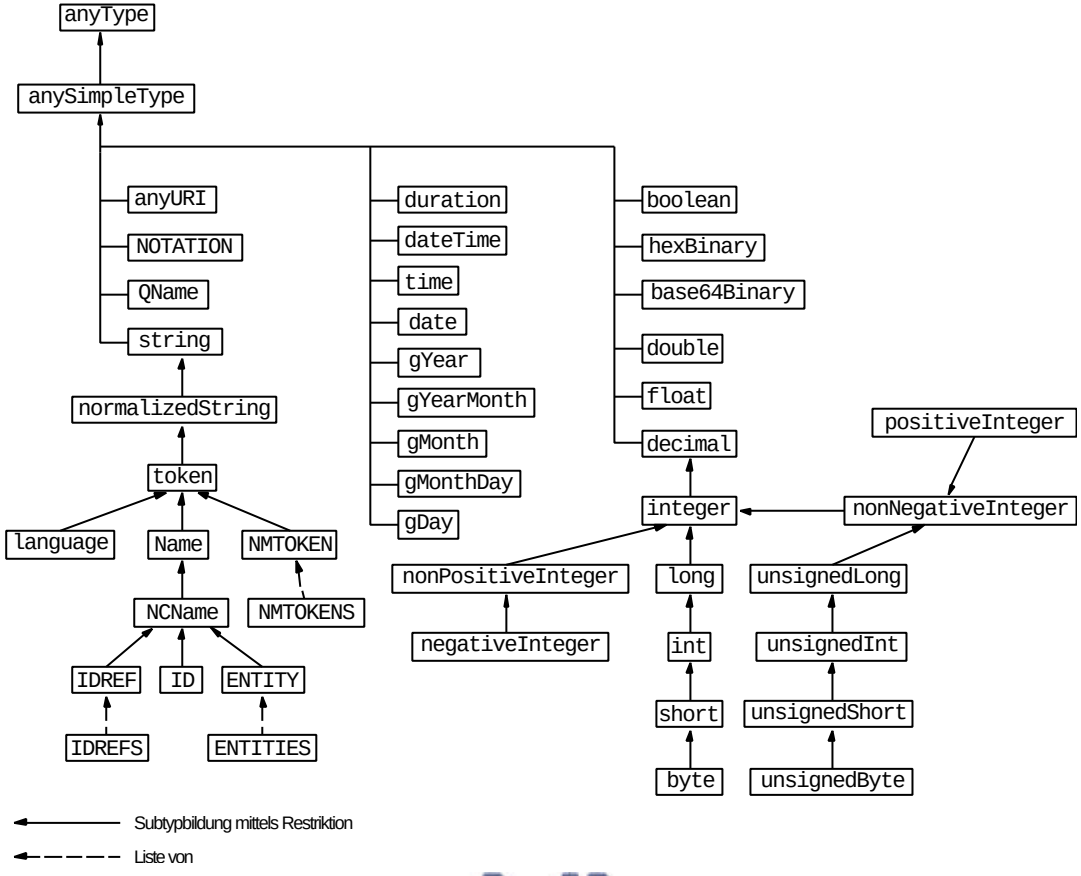
Typhierarchie von XML Schema

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



Beispiele für die Definition einfacher Typen /1

Einleitung, Anwendungen XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Definition einer Ortsangabe als String der Länge 20

```
<xs:simpleType  name="ort">
    <xs:restriction base="xs:string">
        <xs:length value="30"/>
    </xs:restriction>
</xs:simpleType>
```

- Definition einer Kategorie im Bereich zw. 1 und 5

```
<xs:simpleType  name="kategorie">
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="1"/>
        <xs:maxInclusive value="5"/>
    </xs:restriction>
</xs:simpleType>
```

Beispiele für die Definition einfacher Typen /2

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Definition einer Hausnummer als Zahl, der ein Buchstabe folgen kann

```
<xs:simpleType  name="hausnummer">
    <xs:restriction base="xs:string">
        <xs:pattern value="[1-9][0-9]*[a-z]?"/>
    </xs:restriction>
</xs:simpleType>
```

Benutzerdefinierte Typen

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- basieren auf den vordefinierten Datentypen
- durch sogenannte facets werden die Typen weiter eingeschränkt
- facets definieren Eigenschaften über einem Datentyp, die ihn von anderen Datentypen unterscheiden
  - lenght, minLenght, maxLength,
  - pattern, enumeration,
  - whitespace,
  - minExclusive, maxExclusive, minInclusive, maxInclusive,
  - precision, scale, encoding,duration und period.

Benutzerdefinierte Typen – Beispiel

Einleitung, Anwendungen XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Definition einer Adresse

```
<xs:complexType name="Adresse">
  <xs:complexContent>
     <xs:restriction base="xs:anyType">
       <xs:sequence>
         <xs:element name="Ort" minOccurs="1"
                     maxOccurs="1" type="ort" />
          <xs:element name="PLZ" minOccurs="1"
                      maxOccurs="1" type="xs:integer" />
          <xs:element name= "Strasse" minOccurs="1"
                      maxOccurs="1"
                      type= "xs:string"/>
          <xs:element name="Nr" minOccurs="1"
                      maxOccurs="1"
                      type="hausnummer"/>
        </xs:sequence>
      </xs:restriction>
   </xs:complexContent>
</xs:complexType>
```

Attributdeklaration

Einleitung, Anwendungen XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Attributdeklarationen beinhalten

- einen Namen,
- einen (einfachen) Typ,
  (vordefiniert oder benutzerdefiniert)
- eine Angabe, wie das Attribut verwendet wird (use), mögliche Werte: optional, required, fixed
- einen optionalen Defaultwert (value).
  Ist use=´fixed´, so gibt der Defaultwert den Wert des Attributes an.

Beispiel zur Attributdeklaration

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

```
<xs:attribute name="id" type="xs:ID" use="required"/>

<xs:attribute name="url" type="xs:string"
              use="required"/>

<xs:attribute name="creation-time" type="xs:string"
              use="optional"/>

<xs:attribute name="autor" type="xs:string"
              use="required" value="Hans Müller"/>
```

Deklaration von Attributgruppen

Einleitung, Anwendungen XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Attributgruppen sind bei mehrfacher Verwendung der gleichen Attribute sinnvoll.

Beispiel:

```
<xs:attributeGroup name="preis_attribute">
    <xs:attribute name="waehrung" type="xs:string"
                  use="required"/>
    <xs:attribute name="betrag" type=preis_typ"
                  use="required"/>
</xs:attributeGroup>
```

Elementdeklaration

Einleitung, Anwendungen XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Elementdeklarationen enthalten

- einen Namen (name) oder eine Referenz (ref) auf eine andere Elementdeklaration
- einen Typ (type)
- Default- und Fixed-Werte (default, fixed) möglich
- die Häufigkeit des Auftretens (minOccurs, maxOccurs)
- Weiterhin kann durch nilable definiert werden, dass das Element leer sein kann.

Beispiele zur Elementdeklaration /1

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

// Deklaration eines Hotelnamens

```
<xs:element name="Hotelname" type="xs:string"/>
```

// Deklaration einer (optional auftretenden) Kategorie

```
<xs:element name="Kategorie" minOccurs="0"
            maxOccurs="1" type="kategorie"/>
```

// Deklaration einer Adresse als Verweis auf eine andere Elementdeklaration

```
<xs:element ref="Adresse"/>
```

Beispiel zur Elementdeklaration /2

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

```
<xs:element name="Hotel">
  <xs:complexType>
    <xs:sequence>
        …
        <xs:element ref="Telefon" minOccurs="1" maxOccurs="3"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Telefon">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Vorwahl" minOccurs="0"
                  maxOccurs="1" type="xs:string"/>
      <xs:element name="Rufnummer" minOccurs="1"
                  maxOccurs="1" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Definition von „Integritätsbedingungen“

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

definiert Bedingungen, die in den XML-Dokumenten erfüllt sein müssen.

Möglich sind:

- Eindeutigkeitsbedingung (unique)
- Schlüssel (key)
- sowie Referenzen auf Schlüssel (keyref)

Die Definition der Elemente und Attribute, auf die eine Integritätsbedingung angewendet werden soll, erfolgt über einen XPath-Ausdruck.

Beispiele

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

```
<xs:key name="Hotel_key">
    <xs:selector xpath="/Hotelinformationen/Hotel"/>
    <xs:field xpath="@HotelID/"/
</xs:key>

<xs:keyref name="Hotelverzeichnis_keyref" refer="Hotel_key“>
    <xs:selector xpath="/Hotelverzeichnis/Ort/Hotels"/>
    <xs:field xpath="@HotelID/"/>
</xs:keyref>
```

- selector gibt eine Knotenmenge an, die eindeutig sein soll.
- Mit field wird relativ dazu angegeben, auf welchem Elementwert oder Attribut die Eindeutigkeit getestet werden soll.

Definition globaler Typen

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Typdefinitionen können global erfolgen.

- zugeordneter Bezeichner existiert
- Typdefinitionen können mehrfach verwendet werden.

Global definierte Typen können die Basis für andere Typdefinitionen sein.

Beispiel für globale Typdefinitionen

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

```
<xsd:element name="adresse">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="ort"/>
      <xsd:element name="plz" minOccurs="0"/>
      <xsd:element name="strasse"/>
      <xsd:element name="hausnummer"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
<xsd:element ref="hausnummer" type="xsd:integer"/>
<xsd:element ref="ort" type="xsd:string"/>
<xsd:element ref="plz" type="xsd:integer"/>
<xsd:element ref="strasse" type="xsd:string"/>
```

Definition lokaler Typen

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Typdefinitionen können lokal erfolgen.

- zugeordneter Bezeichner existiert nicht
- Definitionen sind dementsprechend nicht mehrfach verwendbar.

Beispiel zur Definition lokaler Typen

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

```
<xs:element name="adresse">
  <xs:complexType>
    <xs:all>
      <xs:element name="plz">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:length value="5"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="ort" type="xs:string"/>
      <xs:element name="strasse" type="xs:string"/>
      <xs:element name="hausnummer">
        <xs:simpleType name="hausnummer“>
          <xs:restriction base="xs:string">
            <xs:pattern value="[1-9][0-9]+[a-z]?"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:all>
  </xs:complexType>
</xs:element>
```

Kombination beider Möglichkeiten

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Globale oder lokale Typdefinitionen?
- Meist Kombination beider Varianten sinnvoll.
- Typen, die mehrfach verwendet werden sollen oder die Basis für weitere Typdefinitionen darstellen, müssen lokal definiert werden.

XML Schema - Zusammenfassung

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Vorteile von XML Schema:

- umfangreiche Darstellungsmöglichkeiten
- Zuordnung von Typinformationen
- Darstellung von Integritätsbedingungen
- Hat selbst XML-Syntax (für Verarbeitung wichtig, Verwendung von XML-Prozessoren zum Zugriff möglich, Einsatz von XSLT)

Nachteile von XML Schema:

- sehr umfangreich
- schwer zu lesen und manuell zu erstellen

XML Schema vs. DTD

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- XML Schema umfasst alle Modellierungsmöglichkeiten, die mit der DTD bereitstehen, mit Ausnahme von Parameter-Entities.
- Übersetzung: DTD in XML Schema → verlustfrei, wenn die Entities vor der Übersetzung expandiert werden

- DTDs enthalten wesentlich weniger Darstellungsmöglich-keiten als XML Schema und auch keine Typinformationen.
- Übersetzung: XML Schema in DTDs → Informationen gehen verloren

Weiterführende Literatur

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

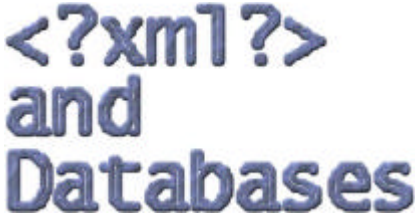
- www.w3.org (Informationen des W3C)
- Neil Bradley: *The XML companion*, Addison Wesley, 1998
- Henning Behme, Stefan Mintert: *XML in der Praxis,* Addison Wesley, 1998.
- Robert Eckstein: *XML pocket reference,* O´Reilly, 1999

- Erhard Rahm, Gottfried Vossen (Hrsg.) *Web und Datenbanken*, dPunkt-Verlag, 2002,

  (Kapitel 2: XML Schema von Schöning und Waterfeld)

XML und Datenbanken
Architektur

Meike Klettke
Universität Rostock
Fachbereich Informatik
meike@informatik.uni-rostock.de



www.xml-und-datenbanken.de

Motivation

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- XML-Dokumente können für sehr verschiedene Anwendungen eingesetzt werden
- Aussehen der Dokumente unterscheidet sich stark

- Vielzahl von Methoden zur Speicherung existiert
- Mehrere Varianten zur Modellierung von XML-Dokumenten und deren Struktur

- **Klassifikation** und
- Einordnung bekannter Methoden

Inhalt

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- 3-Ebenen von Datenbanken
- 3-Ebenen in der Dokumentverarbeitung ?
- 3-Ebenen für XML-Anwendungen ??

- Logische Ebene
- Physische Ebene
- Konzeptelle Ebene

} Reihenfolge entspricht der historischen Entwicklung

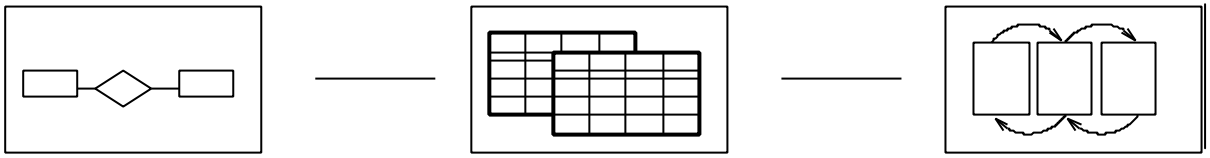
Drei-Ebenen Architektur von Datenbanken

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



- Konzeptueller Entwurf: ERM
- Logische Ebene: Relationenmodell, relationale Algebra
- Physische Ebene: interne Speicherung, Indizierung, Transaktionsverarbeitung, Anfrageoptimierung

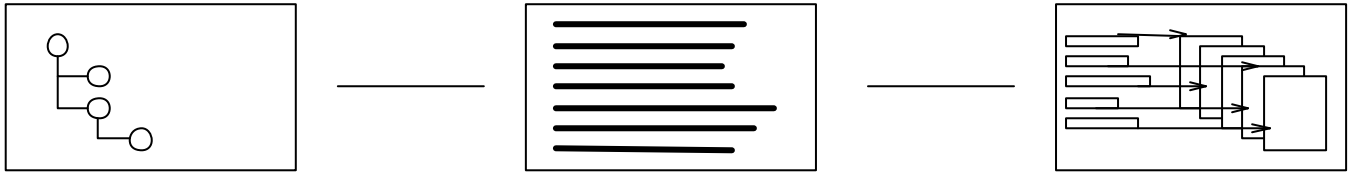
Drei-Ebenen Architektur in der Dokumentverarbeitung

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



konzeptuelle Ebene | logische Ebene | physische Ebene

- Konzeptueller Entwurf: Editoren
- Logische Ebene: Dokumentenmodell
- Physische Ebene: interne Speicherung, Indizierung, Anfragerealisierung

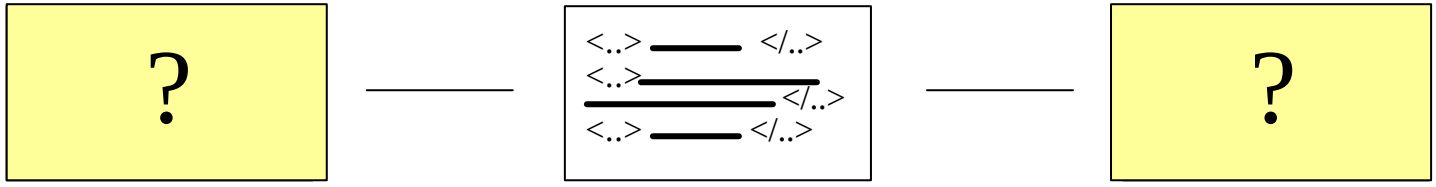
Drei-Ebenen Architektur für XML-Anwendungen

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



konzeptuelle Ebene | logische Ebene | physische Ebene

- XML-Dokumente
  - Anfragen, Updates ?
  - Transformation ?
  - Speicherung ?
  - Modellierung ?

Unterteilung von XML-Dokumenten

Datenzentrierte Dokumente

strukturiert, regulär

Beispiele: Produktkataloge, Bestellungen, Rechnungen

```
<order>
  <customer>Meyer</customer>
  <position>
    <isbn>1-234-56789-0</isbn>
    <number>2</number>
    <price currency=„Euro“>30.00</price>
  </position>
</order>
```

Dokumentzentrierte Dokumente

unstrukturiert, irregulär

Beispiele: wissenschaftliche Artikel, Bücher, E-Mails, Webseiten

```
<content>
XML builds on the principles of two existing
 languages, <emph>HTML</emph> and
<emph>SGML</emph> to create a simple
mechanism ..
The generalized markup concept ..
</content>
```

Semistrukturierte Dokumente

datenzentrierte und dokumentzentrierte Anteile

Beispiele: Veröffentlichungen, Amazon

```
<book>
  <author>Neil Bradley</author>
  <title>XML companion</title>
  <isbn>1-234-56789-0</isbn>
  <content>
    XML builds on the principles of two existing
    languages, <emph>HTML</emph> and ..
  </content>
</book>
```

Bedeutung des Dokumentcharakters

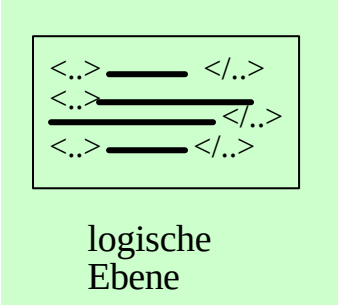
Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Sehr verschiedene Arten von XML-Dokumenten auf logischer Ebene



- Das heißt:
  - nicht nur eine Methode für Anfragen, zur internen Speicherung, zur konzeptuellen Modellierung existiert
  - sondern jeweils ganze Bandbreite

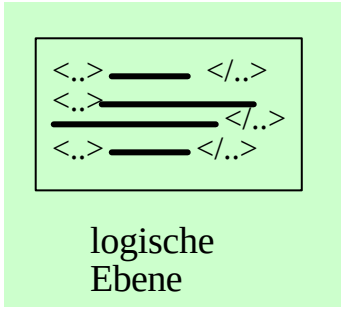
Logische Ebene

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



dokument-zentriert

Dokumentmodell
Anfragen/Updates an Struktur und Inhalt
XML, SGML
XQuery, XPath, DOM
XSLT

semi-strukturiert

Daten- und Dokumentmodell
Anfragen/Updates an Struktur und Inhalt
XML, OEM
XQuery, Lorel
XSLT

daten-zentriert

Datenmodell
Anfragen/Updates an Inhalt
XML, RDBM, OODBM
XQuery, SQL, OQL
XSLT, DB-Import, -Export

Realisierungen für die logische Ebene

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Dokumentzentrierte XML-Dokumente
  - Darstellung: **XML**, SGML
  - Anfragen: **XQuery**, XPath, IR-Anfragen
  - Transformationen: **XSLT**
- Semistrukturierte XML-Dokumente
  - Darstellung: **XML**, OEM
  - Anfragen: **XQuery**, Lorel
  - Transformationen: **XSLT**
- Datenzentrierte XML-Dokumente
  - Darstellung: **XML**, rel. DM, oo DM
  - Anfragen: **XQuery**, SQL, OQL
  - Transformationen: **XSLT**, Import- und Export von Datenbankinformationen

Anforderungen an die Speicherung von XML-Dokumenten

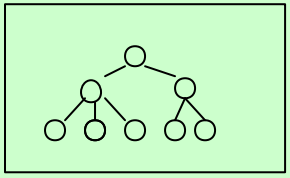
Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Nach (Fiebig, Kanne, Moerkotte)



physische Ebene

- effektive Speicherung
- effizienter Zugriff auf XML-Dokumente oder Teile davon
  - Transaktionsverwaltung
  - Unterstützung von XPath und XQuery
  - Unterstützung von SAX und DOM für Anwendungen
- Wiederherstellbarkeit der Dokumente

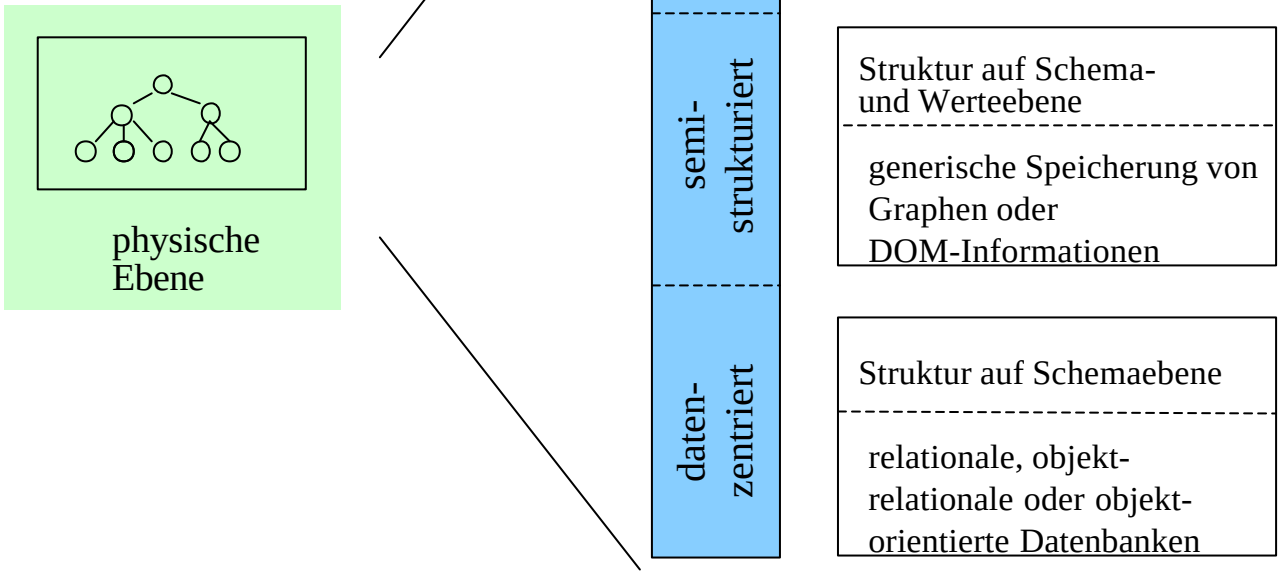
Realisierungen für die physische Ebene / 1

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



Realisierungen für die physische Ebene / 2

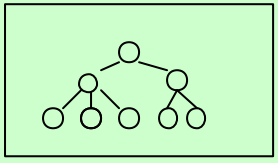
Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Speicherung der XML-Dokumente als Ganzes und Indizierung (textbasiert native)
  - Volltextindex
  - Volltext- und Strukturindex
- Speicherung der Graphenstruktur (modellbasiertes natives Verfahren)
  - generische Graphspeicherung
  - Speicherung der DOM-Informationen
- strukturierte Abbildung auf Datenbanken
  - relationale Datenbanken
  - objekt-orientierte und objekt-relationale Datenbanken
  - Einsatz von Mappingverfahren



physische Ebene

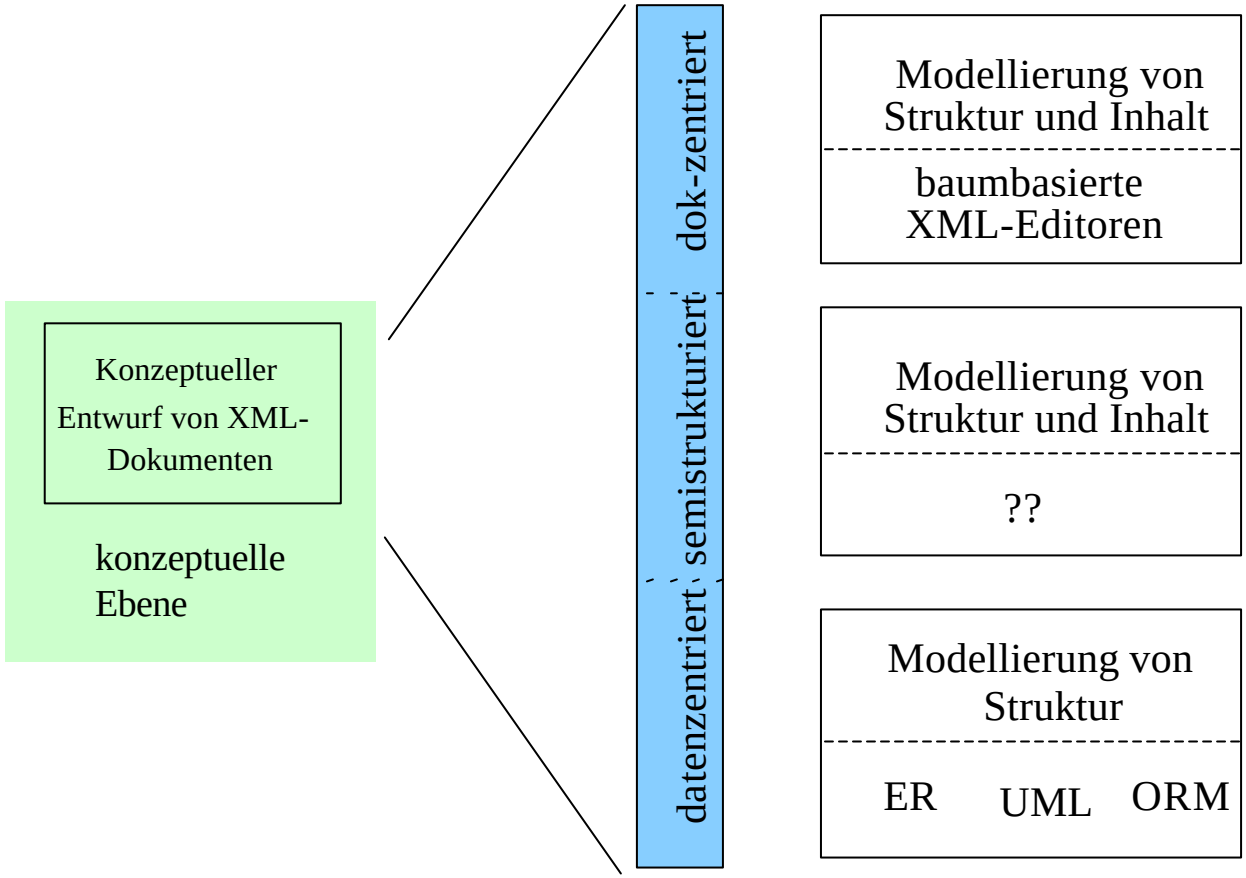
Konzeptuelle Ebene

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



Realisierungen für die konzeptuelle Modellierung

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Dokumentzentrierte XML-Dokumente
- Graphische Editoren
- Visualisierung von Baum- und Graphstruktur

Semistrukturierte XML-Dokumente
- ??

Datenzentrierte XML-Dokumente
- Erweiterung von ER
- Erweiterung und Einschränkung von UML
- Erweiterung von ORM

Konzeptueller Entwurf von XML-Dokumenten

konzeptuelle Ebene

Hier werden in Zukunft noch weitere Forschungsansätze und Tools dazukommen

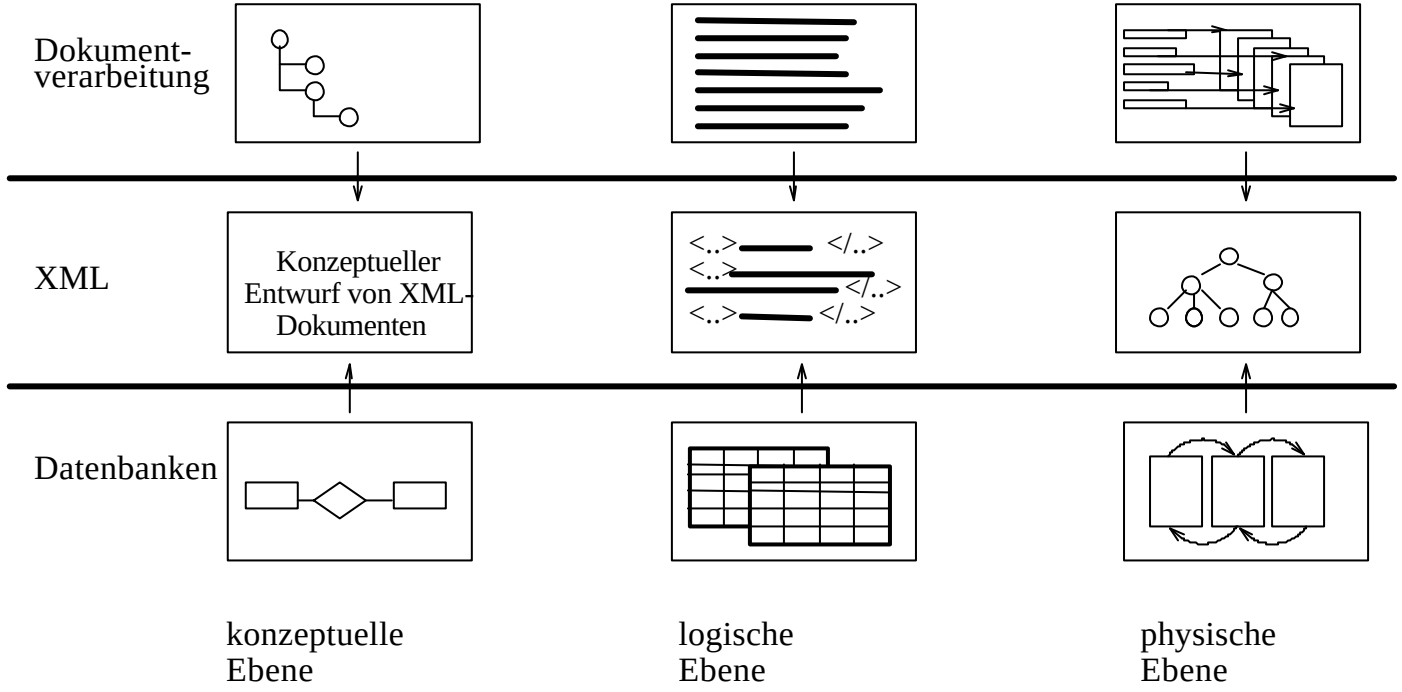
Zusammenfassung: XML-Architektur

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



Zusammenfassung

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- XML-Datenbanken
  - Vielzahl von Verfahren und Techniken existieren nebeneinander
  - oft aus den Bereichen Datenbanken und Dokumentverarbeitung
  - generell: es gibt keine „beste Lösung“
  - geeignetste Lösung ist abhängig von konkreten Erfordernissen



  - vieles ist noch in der Entwicklung

Weiterführende Informationen

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- XML-Anfragesprachen
- Überblick über Speicherungsverfahren
- Indizierungstechniken
- Umsetzung in Systemen

XML und Datenbanken — Anfragesprachen

Holger Meyer Meike Klettke

Database Research Group
University of Rostock
E-Mail: {hme|meike}@informatik.uni-rostock.de

7. Oktober 2002

Inhalt

1. Überblick zu XML-Anfragesprachen
2. Anforderungen
3. **XPath 1.0**
4. XQL
5. XML-QL
6. XML Lorel
7. **XQuery 1.0**
8. Zusammenfassung

1. XML Anfragesprachen
1.1. Anforderungen

nach Maier, et al., weitere siehe W3C/XML Query

Abgeschlossenheit *Closure property*, XML ist sowohl Ein- als auch Ausgabe

Einbettung von Anfragen in XML und umgekehrt *Embedding*, Anfragen lassen sich in XML einbetten, XML-Fragmente können in Anfrage enthalten sein

Beliebiger Ort der Verarbeitung *Server-side processing*, Abgeschlossenheit und Kontextfreiheit

Optionales Schema (DTD) Bearbeitung muß nicht an das Vorhandensein eines Schemas gebunden sein, kann aber zur Optimierung genutzt werden

Anforderungen (2)

Erhaltung von Reihenfolge und Enthaltensein ordnungserhaltend: Document ordering, Nesting

Maschinengenerierbarkeit von Anfragen leichte Generierung von Anfragen durch Programme, GUI, . . .

Traversierung von Hyperlinks Unterstützung von Hypertextstrukturen, d. h. Folgen von Hyperlinks, minimal: Unterstützung von XML-Links (ID/IDREF) und XLinks

erweiterbares Typkonzept Unterstützung eines erweiterbaren Typkonzeptes, notnwendig z.B. für Vergleichsoperatoren auf unterschiedlichen Typen

Algebra algebraische Grundlage, Vollständigkeit, Abgeschlossenheit, Orthogonalität, Optimierbarkeit, Implementierung

1.2. Grundoperationen

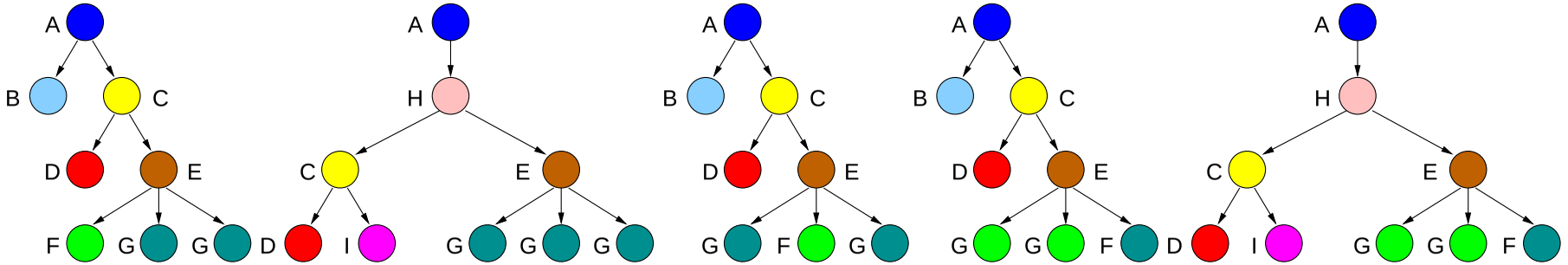
Mindestforderung, nach W3C/XML/QL98/David Maier

Selektion Selektion eines Dokumentes oder Dokumentelementes anhand Inhalt, Struktur oder Attribute

Extraktion und Reduktion gezielte Extraktion von Subelementen eines Dokumentes, Löschen ausgewählter Subelemente eines Elementes

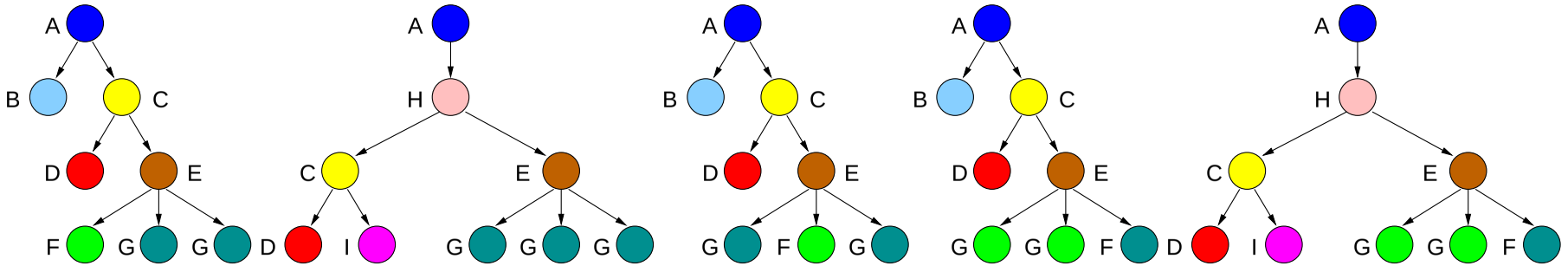
Kombination und Restrukturierung Zusammenfassen zweier oder mehrerer Elemente in einem neuen Element, Erzeugen einer neuen Elementmenge, die Anfrageergebnisse enthalten

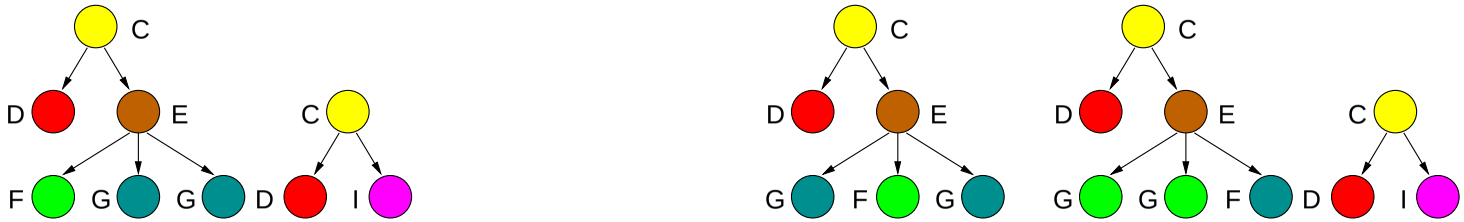
Selektion


A
B
C
D
E
F
G
G
D
A
H
C
E
I
G
G
G
A
B
C
D
E
G
F
G
A
B
C
D
E
G
G
F
D
A
H
C
E
I
G
G
F

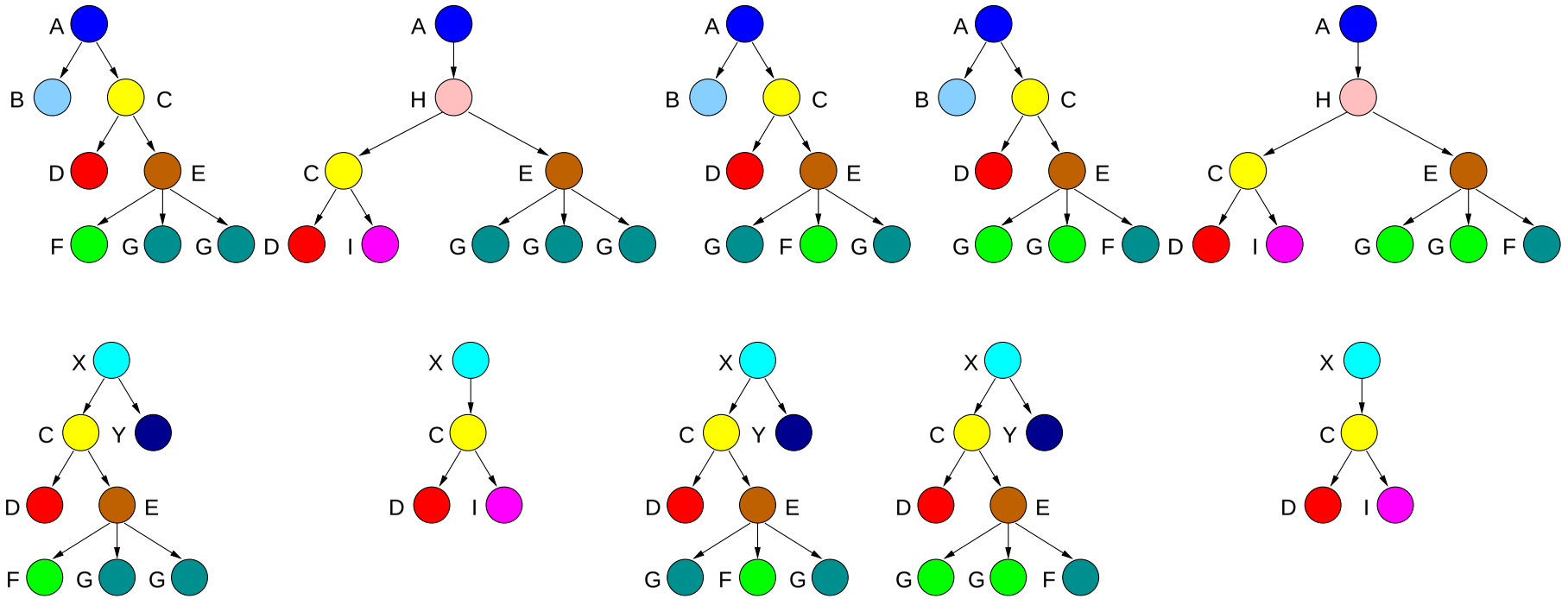

A
B
C
D
E
F
G
G
A
B
C
D
E
G
G
F
D
A
H
C
E
I
G
G
F

Extraktion und Reduktion





Kombination und Restrukturierung



1.3. Sprachvorschläge

- XPath 1.0, W3C
- XQuery 1.0 (und XPath 2.0), W3C (vormals Quilt)
- weitere:
  - XSL, XQL
  - XML-QL
  - XML-GL, graphische Anfragesprache
  - IRQL, SQL-Aufweichung für vage Anfragen, Volltextoperationen
  - SQL/MM, SQL-Erweiterungen für Content based retrieval
  - ...

2. XPath 1.0

- wird in einer Reihe von Standards benutzt: XSL, XLink/XPointer, XQuery
- ist bereits Recommendation,
- Adressierung von Teilen eines Dokumentes,
- Selektion von Knotenmengen,
- Formulierung von Bedingungen an diese Knotenmegen,
- grundlegendes Konstrukt sind XPath-Ausdrücke:
  - Pfadausdrücke (*Location path*),
  - logische und mathematische Verknüpfungen,
  - Funktionsaufrufe
- sind vom Typ her Knotemmengen oder Werte (`boolean`, `number`, `string`)

2.1. Pfadausdrücke

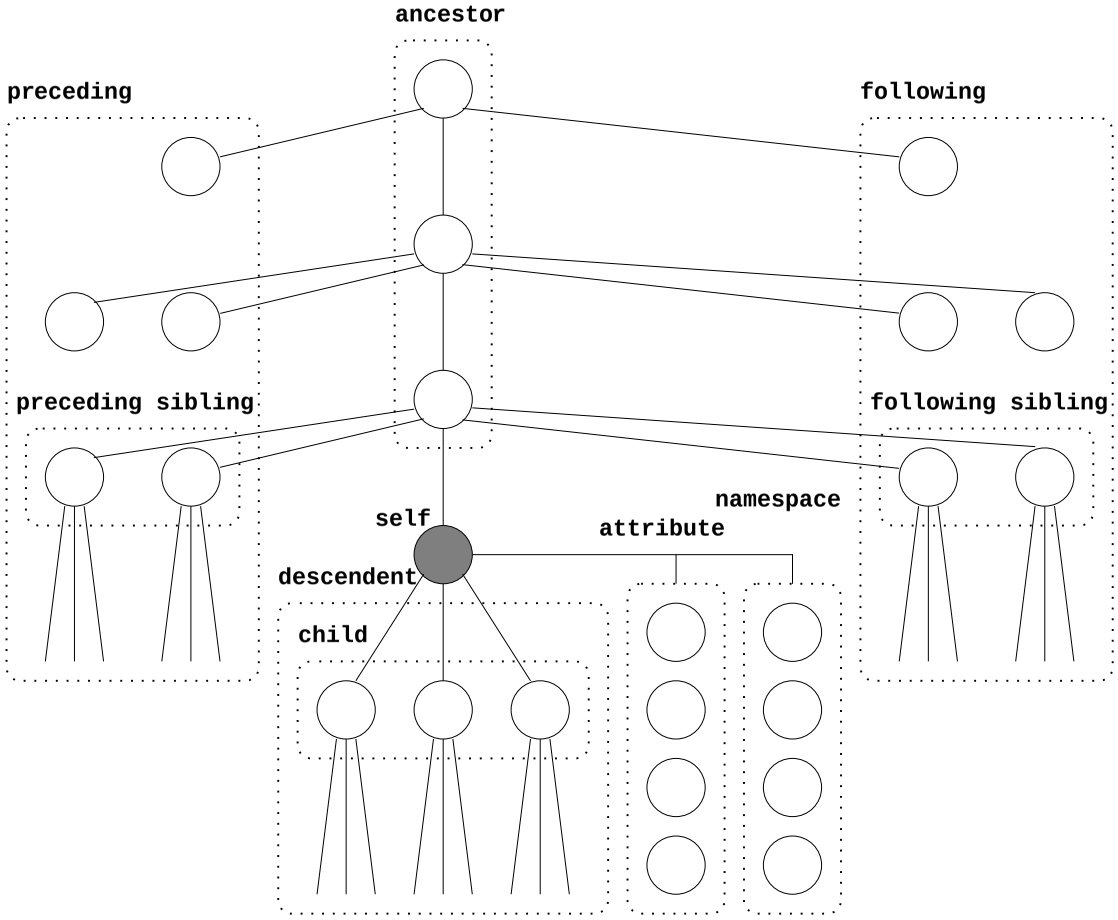
- kann aus mehreren Teilausdrücken (Bausteinen, Schritten) bestehen,
- Kopplung erfolgt über `'/'` (analog Dateisystemkomponenten),
- es gibt absolute und relative Pfadausdrücke,
- ein Schritt liefert eine Knotenmenge oder Werte,
- Abarbeitung zusammengesetzter Ausdrücke erfolgt “von links”,

2.2. Schritte

Achse::*Knotentest*[*Prädikat*]

- Achsenangabe, stellt Beziehung zwischen Kontextknoten und zu selektierenden Knoten her,
- Knotentest, gibt Knotentyp und den Namen der zu selektierenden Knoten an,
- null oder mehrer Prädikate schränken die Knotenmenge ein

2.3. Navigationsachsen in XPath



Knotentests

wenn wahr, dann wird jeweiliger Knoten in Ausgabemenge übernommen

`node()`

`*` wahr für alle Elementknoten

`text()` wahr für alle Textknoten

`comment()` wahr für alle Kommentarknoten

`processing-instruction()` wahr, wenn es sich um *Processing instruc-tion*s handelt

Prädikate

- wenn wahr, dann wird jeweiliger Knoten in Ausgabemenge übernommen,
- wenn Typ des Prädikates `number` ist, wird Knoten der ensprechenden Kontextposition übernommen,
- `and`, `or` als logische Operatoren bei Wahrheitsausdrücken,
- Vergleichsoperatoren: `<`, `<=`, `!=`, `...`,
- *Achtung*: in XML-Dokumenten Kodierung mit `>` und `<`,
- für Zahlen: `+`, `-`, `mod` und `div`,
- `'|'` zur Vereinigung von Knotenmengen,

XPath syntaktisch

Abk	Langform	Bedeutung
	`child::`	fehlt Achsenangabe, werden Kinderknoten selektiert
`.`	`self::node()`	bezeichnet den aktuellen Knoten
`..`	`parent::node()`	bezeichnet den Vaterknoten
`/`		bezeichnet den Wurzelknoten oder trennt Pfadkomponenten
`//`		bezeichnet alle Nachkommen des aktuellen Knotens
`@`	`attribute::`	bezeichnet Attribute des aktuellen Knotens
`*`		bezeichnet einen Knoten mit einem beliebigen Namen
`[`*expr*`]`		bezeichnet ein Prädikat für den aktuellen Teilpfad, *expr* ist ein boolscher Ausdruck
`[`*n*`]`		wenn das Prädikat aus einer natürlichen Zahl besteht, bezeichnet es das *n*-te Element aus der Liste von Knoten, für die der aktuelle Teilpfad gilt

2.4. Funktionsbibliothek

allgemeine und `string`, `boolean`, `number`-Funktionen, Auswahl:

`number last()` liefert die Position des letzten Elementes,

`number postion()` gibt Kontextposition an,

`node-set id(`*`object`*`)` liefert den Knoten mit der referenzierten ID, Auflösung von IDREF,

`boolean contains(`*`string, string`*`)` wahr wenn zweites Argument Teil des ersten ist,

`boolean not(`*`boolean`*`)` Negation des angebenen Wertes,

`number sum(`*`node-set`*`)` Summer der zu Zahlen gewandelten Argument-knoten,

`...`

2.5. XPath — Beispiele

`/band[album/title = 'Unplugged' and year 1993]`

alle Bands mit Alben, deren Titel „Unplugged“ ist, und die nach 1993 erschienen

`/band[contains(album, 'Platte des Jahres')]/name`

liefert die Namen aller Bands, deren Albenbeschreibung „Platte des Jahres“ enthält

`/band[count(album) > 2]/(name|image)`

die Namen und bilder aller Bands mit mehr als zwei Alben

XPath — Achtung

- Prädikate sind nicht kommutative, `a[b][2] != a[2][b]`
- es gibt Probleme mit vorwärts/rückwärts-Achsen und Prädikate
- Expansion von `//` ist aufwending
- numerische Prädikate
- Optimierung macht Push Down von Knotentests und Prädikaten notwendig

3. XQL und XSLT

- XSL Stylesheet-Sprache für XML-Dokumente
- XSL selbst als XML-Ausprägung definiert
- liefert Mittel zur Transformation von XML-Dokumenten
- bietet Anfrage- und Restrukturierungsmechanismen
- Muster aus Pfadangaben, Wildcards und Qualifikationen
- aufgrund Zeichenketten-basierten Syntax lassen sich Muster z.B. in URL's oder Attributwerten verwenden
- zwei Namensräume: **`xsl`** Umformungsmöglichkeiten von XSL
- **`fo`** repräsentiert die Formatierung, eigenständige XML-Grammatik, unabhängig von der Transformation

3.1. XSLT-Elemente

- Definition von Templates für Teile des Zieldokumentes
- Füllen von Templates mit Daten aus dem Quell-XML-Dokument
- Regeln, die Anfragemustern (XSL patterns) mit Templates verbinden
- Gruppierung von Regeln für rekursive oder geschachtelte Verarbeitung
- Nachverarbeitung von Anfrageergebnissen, z.B. Sortierung oder Filterung

3.2. XQL — Übersicht

XQL, Vorschlag von Robie (Texcel Inc), Lapp (webMethods Inc) und Schach (Microsoft)

XQL greift XSLT-Ansätze und XPath auf

XQL-Anfragen folgen einer einfachen Struktur:

```
<xsl:for-each select = "XSL pattern">
  <!-- building result -->
</xsl:for-each>
```

XQL-Beispiel

Titel aller Platten Eric Claptons als XML-Fragment erzeugen:

```
<xsl:for-each select = "band[name='Eric Clapton']/album/title">
   <result>
     <album>
       <band><xsl:value-of select="name" /></band>
       <xsl:for-each select="album/title">
         <title><xsl:value-of /></title>
       </xsl:for-each>
     </album>
   </result>
</xsl:for-each>
```

Nachteile von XPath, XSLT und Erweiterung XQL

- Anfragemöglichkeiten beziehen sich auf Einzeldokumente, nicht auf Kollektionen oder gar verteilte Dokumentenbestände.
- Es gibt keine Möglichkeit, Verbünde z.B. über gemeinsame Variablen in verschiedenen Transformationsregeln zu realisieren.
- Einzelne Elemente können nur durch Muster identifiziert werden, ein unabhängiger Mechanismus zur Objektidentifikation fehlt. Damit kann eine Instanz eines Elementes evtl. mehrfach im Ausgabedokument auftauchen.
- In XSL-Mustern/Pfadausdrücken können keine regulären Ausdrücke verwandt werden.
- Inhalte können nicht mit Mustern (regulären Ausdrücken) beschrieben werden

Vorteile von XSL und Erweiterung XQL

- kompakte Syntax, geeignet für einfache Web-Anwendungen
- Transformationsmöglichkeiten
- XSL, XSLT, XQL/XPath sind W3C „Standards“
- es existieren eine Reihe von Implementierungen

4. XML-QL

Projekt der AT&T Labs (Deutsch, Fernandez, Florescu, Levy, Suciu)

Ähnlich wie XQL ist XML-QL in der Lage

- Baum- oder Graphenstrukturen als Ergebnis zu liefern,
- neue Elemente zu erzeugen,
- Restrukturierung vorzunehmen und
- Anfragen an XML-Dokumente zu stellen.

XML-QL — Abgrenzung zu XQL

- QBE-ähnlicher Anfragemechanismus
- durch **IN**-Operator Zugriff auf verschiedene Dokumente
- XML-QL-Muster können Variablen enthalten.
  - Anfrageergebnis: Menge aller möglichen Bindungen der Variablen an Werte, die angegebener Bedingung entsprechen,
  - macht Gruppierungen und Verbünde (Joins) möglich
- Nachteil: Document ordering geht verloren, Ergebnis sind Tupelmengen
- Grundmuster:

```
WHERE XML-QL pattern
CONSTRUCT output
```

XML-QL-Beispiele

Beispiel: Liste der Albumtitel, bei denen Eric Clapton mitspielte

```
WHERE
  <album>
    <musician>Eric Clapton</musician>
    <title>$t</title>
  </album> IN "www.abb.com/music/discs.html"
CONSTRUCT
  <album><title>$t</title></album>
```

XML-QL-Beispiele II

Beispiel: Gruppierung der Albentitel pro Band

```
WHERE
  <album>$a</> IN "www.abb.com/music/discs.html",
  <band><name>$b</></> IN $a
CONSTRUCT
  <result>
    <name>$b</>
    WHERE <title>$t</> IN $a
    CONSTRUCT <title>$t</>
  </>
```

XML-QL-Beispiele III

Beispiel: Platten von Bands mit Musikern, die vor 1935 geboren wurden

```
WHERE
  <album>
    <band><name>$b</></>
  </> CONTENT_AS $a IN "www.cdnow.com/records.html"
  <musician>
    <band><name>$b</></> <born>$y</>
  </> IN "www.bands-online.com/musicians.html",
  $y < 1935
CONSTRUCT
  <album>$a</>
```

5. XML Lorel

Lore Projekt

- Stanford University, CA
- komplettes DBMS, speziell zur Verwaltung semi-strukturierter Daten
- Überdenken aller Aspekte eines DBMS: Speicherverwaltung, Indexierung, Anfrageoperationen, Optimierung, Benutzerinterface,
- Datenmodell: OEM, einfaches, selbst beschreibendes, geschachteltes Objektmodell (Labeled directed graph)
- atomare Objekte (`int, real, string, gif, ...`)
- komplexe Objekte ($setof < label, subobject >$)
- Anfragesprache Lorel: „erweitertes Subset“ von OQL

XML OEM-Anpassungen

XML-basiertes Datenmodell:

- XML-Tag
  $\rightarrow$ String-Wert tag
- Attribute
  $\rightarrow$ geordnete Liste $listof < att - name, atomic - value >$
- Crosslink-Subelemente, Attribut `idref, idrefs`
  $\rightarrow$ geordnete Liste $listof < label, eid >$
- Subelemente
  $\rightarrow$ geordnete Liste $listof < label, eid >$

XML Lorel-Anpassungen

- zwei Modi

  Semantic mode Datenbank als Graph, IDREF/IDREFS nicht sichtbar, kein Unterschied zwischen Crosslink und Subelemente

  Literal mode Unterscheidung zwischen Links und Subelementen, Dokumentstruktur bleibt erhalten

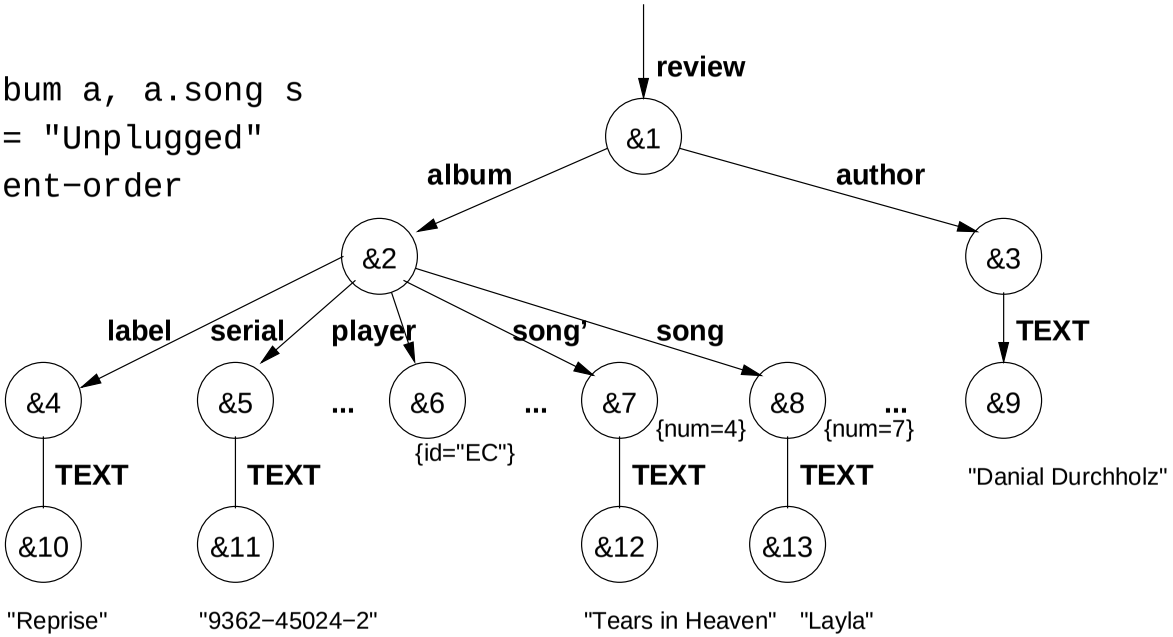
- Pfadausdrücke, Qualifiers für Unterscheidung Subelemente und Attribute
- Restrukturiering mit **`with`**-Klausel
- Funktionen zur Text-Transformation: **`flatten`**, **`concat`**, **`tag`**, **`xml`**, `...`
- Bereichsangaben, Beschränkung der Subelemente

XML Lorel-Beispiel

```
select s
from review.album a, a.song s
where a.musician = "Clapton"
order by document-order
```

XML-Beispiel und OEM-Graph

```
select s
from review.album a, a.song s
where a.title = "Unplugged"
order by document-order
```



Bewertung

- Änderungsoperationen
- einfaches Typkonzept
- Restrukturierung gegenüber Lorel hinzugefügt, `with`
- Dokumentordnung `order-by document-ordering`
- ID/IDREF-Mechanismus wird unterstützt
- XLink, XPointer-Unterstützung unklar, soll aber gehen
- DataGuide und DTD’s

6. XQuery 1.0

- aktueller Vorschlag W3C für XML-Anfragesprache
- basiert auf XPath
- Ähnlichkeit zu SQL
- Basiskonstrukt: FLWR-Ausdruck, steht für **FOR-LET-WHERE-RETURN**
- **FOR/LET**: geordnete Liste von Tupeln gebundener Variablen
- **WHERE**: eingeschränkte Liste von Tupeln gebundener Variablen
- **RETURN**: Ergebniskonstruktion, Instanzen des XML Query data model

Working Drafts (6/7/01)

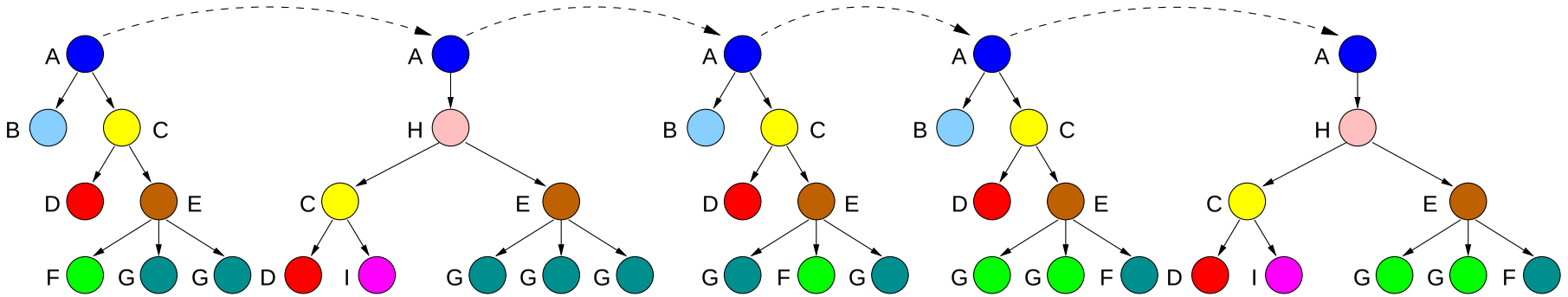
- *XML Query Requirements*
- *XML Query Use Cases*
- *XQuery 1.0: An XML Query Language*
- *XQuery 1.0 and XPath 2.0 Data Model*
- *XQuery 1.0 Formal Semantics* war *XML Query Algebra*
- *XML Syntax for XQuery (XQueryX)*
- *Functions and Operators*

5 Schritte zu W3C „Recommendation“

- Working Draft
- Last Call Working Draft
- Candidate Recommendation
- Proposed Recommendation
- W3C Recommendation

6.1. XQuery — Sprachüberblick

- funktionale Sprache
- Ausdrücke aus anderen Ausdrücken zusammengesetzt
- orthogonale Anwendung unterschiedlicher Ausdruckstypen
- Datenmodell ist geordneter Wald (ordered nodeset, Sequenz)



XQuery — Basiskonstrukte

- XML-Einbettung in oder von XQuery Ausdrücke
- Elementkonstruktoren zur Erstellung oder Ableitung neuer XML-Elemente
- Pfadausdrücke (XPath 1.0) zur Selektion von Dokumentbestandteilen
- Anwendung von datentypspezifischen Operatoren
- Funktionsaufrufe, neben Standardfunktionen nutzerdefinierte Funktionen

XQuery — Basiskonstrukte (cont'd)

- FLWR-Ausdrücke, Anfragen ähnlich zu SQL SFW-Blöcken
- bedingte Anweisungen, zur Steuerung der Auswertung von Ausdrücken
- quantifizierte Ausdrücke unter Verwendung der Quantoren ALL und ANY
- Test von Datentypen und Typumwandlung

XQuery — Generelle Vorgehensweise

- Dokumentzugriff mit `document(`*`url`*`)`
- Knotennavigation mit XPath
- Variablenbindung im Kontext existierender Bindungen
- Operationen auf den Knotenmengen
- Erzeugung neuer Knoten
- Einfache Anfragen:

```
//zimmerausstattung
document("pensionen.xml")//zimmertyp/preis
```

6.2. XQuery Ausdrücke

- Konstanten und Variablen, Identifikatoren case-sensitive

```
$x, "47.2", "Hello"
```

- Kommentare

```
# Kommentar beginnt mit '#', endet mit Zeile
```

- Sequenzen

```
(1, /a/b/c, "Hello")
```

Pfadausdrücke unter Nutzung von XPath

Erweiterungen gegenüber XPath:

- Bereichausdrücke in Prädikaten, *expr* **TO** *expr*
- Variable oder Funktionsaufrufe wie `document(`*`url`*`)` vor Pfadausdrücke stellen

```
/hotel[name="Hotel Hübner"]/zimmertyp[typ="DZ"]
$g/gäste[@nachname = "Heuer"]/@rechnung=>summe
document("db.xml")/hotel[@name="Hübner"]
```

Operatoren und Funktionsaufrufe

- `sum($a + 1, $b - 2), $zaus` **`UNION`** `$haus`
- nutzerdefinierte Funktionen

```
NAMESPACE
  xf = "http://www.w3.org/2001/08/xquery-operators"
DEFINE FUNCTION startChar($str)
  RETURNS xsd:string {
  # sourceStr, startPos (1-based), length
  xf:substring($str, 1, 1)
}
```

Hinweis: wenn `$str` Sequenz, dann auch Rückgabewert

Elementkonstrukturen

- reines XML, wird so in das Ergebnis übernommen, wie es ist

```
<zimmer typ="DZ">
  <ausstattung>TV</ausstattung>
</zimmer>
```

- XML mit geschachtelten Ausdrücken in '{' '}'

```
<zimmer typ={$z/@typ}> {
  $zaus UNION $haus
} </zimmer>
```

Elementkonstrukturen (cont'd)

- XML mit berechneten Element- und Attributnamen

```
<{$z} {$t}="EZ">
  <{$a}>Minibar</>
</>
```

- Vorschlag Don Chamberlin für berechnete Element- und Attributnamen

```
ELEMENT {$z} {
  ATTRIBUT {$t} {EZ}
  ELEMENT {$a} {"Minibar"}
}
```

XQuery Ausdrücke (cont'd)

- Sortierung mit **SORTBY**

```
/hotel[name = "Hübner"]
  SORTBY zimmer/preis DESCENDING
```

- Bedingte Ausdrücke

```
IF expr THEN expr ELSE expr
```

- Verwendung von Quantoren

```
SOME var IN expr SATISFIES expr
EVERY var IN expr SATISFIES expr
```

Typanpassungen

- Typprüfung mit **INSTANCEOF** und **TYPESWITCH**

```
expr INSTANCEOF type
TYPESWITCH ($ausstattung)
CASE zimmeraus RETURN "Zimmer"
CASE konferenzaus RETURN "Konferenz"
DEFAULT RETURN "Hotel"
```

- Typwandlung mit **CAST** und **TREAT**

```
CAST AS type (expr)
TREAT AS type (expr)
```

6.3. FLWR-Ausdrücke

allgemeine Struktur:

```
FLWR-expr ::=
    (FOR-expr | LET-expr)+
    WHERE-expr?
    RETURN-expr
FOR-expr ::=
    FOR $var IN expr (, $var IN expr)*
LET-expr ::=
    LET $var := expr (, $var := expr)*
WHERE-expr ::= WHERE expr
RETURN-expr ::= RETURN expr
```

Variablen

- nach der erfolgten Bindung sind die Werte nicht mehr änderbar
- Variablenbindung nur innerhalb des aktuellen und aller eingeschlossenen Anfrageausdrücke sichtbar
- wird Ausdruck verlassen, ist Variable ungebunden
- Zugriff auf ungebundenen Variablen $\rightarrow$ Ausnahme
- wird hingegen Variable mehrfach gebunden, ist immer unmittelbar zuvor erfolgte Bindung sichtbar
- Typ einer Variablen ergibt sich aus Bindung
- Variablenbindung erfolgt in **FOR**- und **LET**-Ausdrücken

LET-Ausdruck

- **LET**-Ausdruck bindet Menge von Werten
- die bei Auswertung eines Ausdruckes *expr* entstehen
- geschlossen an eine Variable `$var`

```
LET $var := expr [, $var := expr ...]
```

- im einfachsten Fall folgt nach der **LET**-Klausel ein **RETURN**-Ausdruck
- liefert den Variableninhalt als Ergebnis

LET-Ausdruck (cont'd)

```
LET $z := //zimmertyp
RETURN
    $z
```

Dieses einfache Beispiel wird in drei Schritten abgearbeitet:

1. Werte den XPath-Ausdruck `//zimmertyp` aus
2. Weise die resultierende Knotenmenge an die Variable `$z` zu (Variablenbindung)
3. Gebe die gesamte Sequenz von Knoten zurück

FOR-Klausel

- Für jedes Element der Ergebnismenge erfolgt eine Bindung an `$var`
- Wie bei **LET** können mehrere Variablen gebunden werden
- Variablen für alle untergeordneten Ausdrücke sichtbar

```
FOR $var IN expr [, $var IN expr ...]
```

```
FOR $z IN //zimmertyp
RETURN
    $z
```

FOR-Klausel (cont'd)

Abarbeitung jedoch anders:

- `$z` wird jeweils an Elemente der Sequenz (Auswertung von `//zimmer-typ`) gebunden
- Binden erfolgt mehrfach, für jeden Zimmertyp genau einmal
- Nachfolgende Klauseln werden für jede Iteration und Bindung einmal ausgewertet
- **`RETURN`** wird für jeden Schritt ausgewertet, Resultat zu einem Zwischenergebnis hinzugefügt
- Aufsummierte Gesamtergebnis von **`RETURN`** wird am Schleifenende zurückgegeben

Beispiel

```
<hotel name="Hotel Neptun">
   <zimmertyp typ="EZ" preis="180" währung="DM/>
   <foto href="neptun01.jpeg"/>
</hotel>
<hotel name="Hotel Hübner">
   <zimmertyp typ="EZ" preis="150" währung="DM"/>
   <zimmertyp typ="DZ" preis="180" währung="DM"/>
</hotel>
<hotel name="Pension Dräger">
   <foto href="bild-pd01.jpeg"/>
   <foto href="bild-pd02.jpeg"/>
</hotel>
```

Beispiel (cont'd)

XQuery-Anfrage:

```
FOR $hotel IN //hotel
RETURN $hotel/foto
```

Ergebnis:

```
<foto href="neptun01.jpeg"/>
<foto href="bild-pd01.jpeg"/>
<foto href="bild-pd02.jpeg"/>
```

Geschachtelte FOR-Klauseln

```
<billighotels> {
FOR $h IN //hotel
    FOR $z IN $h/zimmertyp
    WHERE $z/@preis <= 100
    RETURN <hotel>
        <name>{ $h/@name }</name>
        <preis>{ $z/@preis }</preis>
    </hotel> } </billighotels>
```

Ergebnis:

```
<billighotels>
  <hotel><name>...</name><preis>...</preis></hotel> ...
</billighotels>
```

Geschachteltes FOR (cont'd)

```
<billighotels> {
FOR $h IN //hotel
RETURN <hotel name={ $h/@name }> {
    FOR $z IN $h/zimmertyp
    WHERE $z/@preis <= 100
    RETURN <preis>{ $z/@preis }</preis>
} </hotel> } </billighotels>
```

Ergebnis:

```
<billighotels>
  <hotel name="Hotel Hübner">
    <preis>...</preis>...</hotel>...
</billighotels>
```

6.4. Was kann XQuery nicht?

- Insert, Delete, Update
- Exception Handling
- Sichtendefinition
- Transaktionen
- . . .

6.5. XQuery-Implementierungen

- Universität Rostock, Guido Rost, `www.xml-and-databases.com`
- „Kweelt“ von der University of Pennsylvania, Arnaud Sahuguet, `db.cis.upenn.edu/kweelt/`
- Fatdog Software, Howard Katz, `www.fatdog.com`
- „XML Query Demo“, Microsoft, Michael Rys, `msdn.microsoft.com/xml/`
- Software AG, Jonathan Robie, `www.softwareag.com/developer/downloads/default.htm`
- Lucent Bell Labs, Jerome Simeon, *XML DevCon 2001*
- GMD Darmstadt, P. Fankhauser, *XML DevCon 2001*

7. Vergleich

Kriterium	XQL	XML-QL	XML Lorel	XQuery
Restructering	$\surd$	$\surd$	$\surd$	$\surd$
n Documents	$-$	$\surd$	$\surd$	$\surd$
Ordering	$\surd$	$-$	$\surd$	$\surd$
Links	$-$	$\surd$	ID/IDREF	$\surd$
Join	$-$	$\surd$	$\surd$	$\surd$
Types	$-$	$-$	integer,real,string,id,idref	$\surd$

8. Zusammenfassung

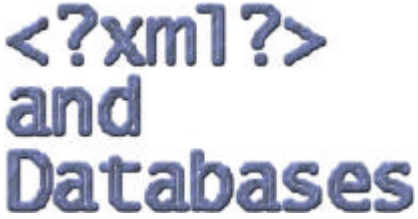
Offene Probleme:

- Typkonzept (XML Schema, RDF, Information Sets, . . .)
- Sichtenkonzept
- Updates (keine Transformationen!)
- XML-Algebren, Optimierung
- Volltext-Suchfunktionalität
- aktuelle Vorschläge des W3C

  `http://www.w3.org/XML/Query`

XML und Datenbanken
Speicherungsmethoden

Meike Klettke
Universität Rostock
Fachbereich Informatik
meike@informatik.uni-rostock.de



www.xml-und-datenbanken.de

Inhalt

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Anforderungen an die Speicherung
- Motivation der Existenz verschiedener Varianten

- Vorstellung verschiedener Speicherungsverfahren
  - Speicherung und Indizierung
  - Speicherung der Graphstruktur
  - strukturierte Speicherung in Datenbanken

- Weiterführende Literatur

Anforderungen an die Speicherung von XML-Dokumenten

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Nach (Fiebig, Kanne, Moerkotte)

- Effektive Speicherung
- Effizienter Zugriff auf XML-Dokumente oder Teile davon
  - Transaktionsverwaltung
  - Unterstützung von XPath und XQuery
  - Unterstützung von SAX und DOM für Anwendungen
- Wiederherstellbarkeit der Dokumente (oder der Informationen aus den Dokumenten)

Bedeutung des Dokumentcharakters

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- XML-Dokumente können die ganze Bandbreite von Daten bis zu Volltextdokumenten einnehmen

  (dokumentzentriert, semistrukturiert, datenzentriert)

- Entsprechend eignen sich auch Speicherungsverfahren von der Dokumentenverarbeitung bis zur Datenbanktechnologie
- Weiterhin: Neuentwicklung von Methoden
- Keine optimale Lösung für alle Anwendungen, Dokumentcharakter spielt entscheidende Rolle

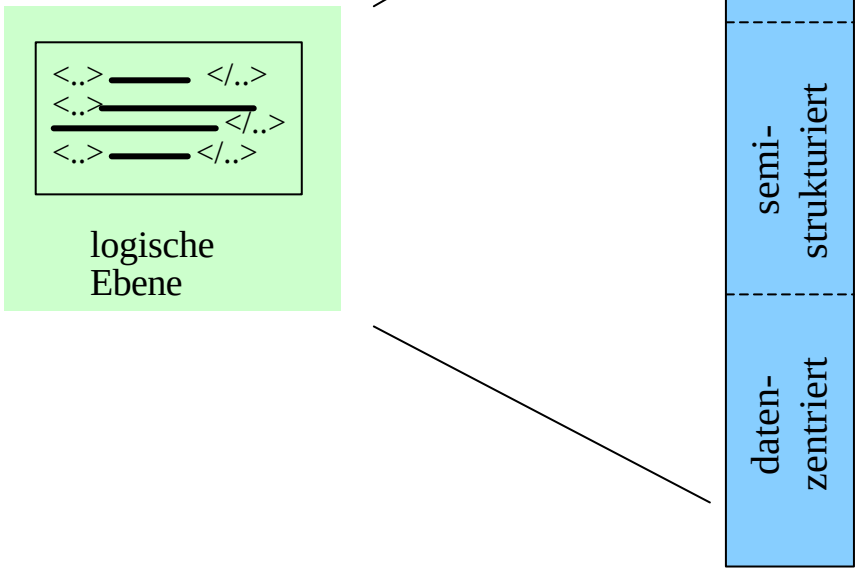
Logische Ebene

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Realisierungen für die logische Ebene

Dokumentzentrierte XML-Dokumente:

- Darstellung: **XML**, SGML
- Anfragen: **XQuery**, XPath, DOM, IR-Anfragen

Semistrukturierte XML-Dokumente:

- Darstellung: **XML**, OEM
- Anfragen: **XQuery**, Lorel

Datenzentrierte XML-Dokumente:

- Darstellung: **XML**, rel. DM, oo. DM
- Anfragen: **XQuery**, SQL, OQL

große Band-breite

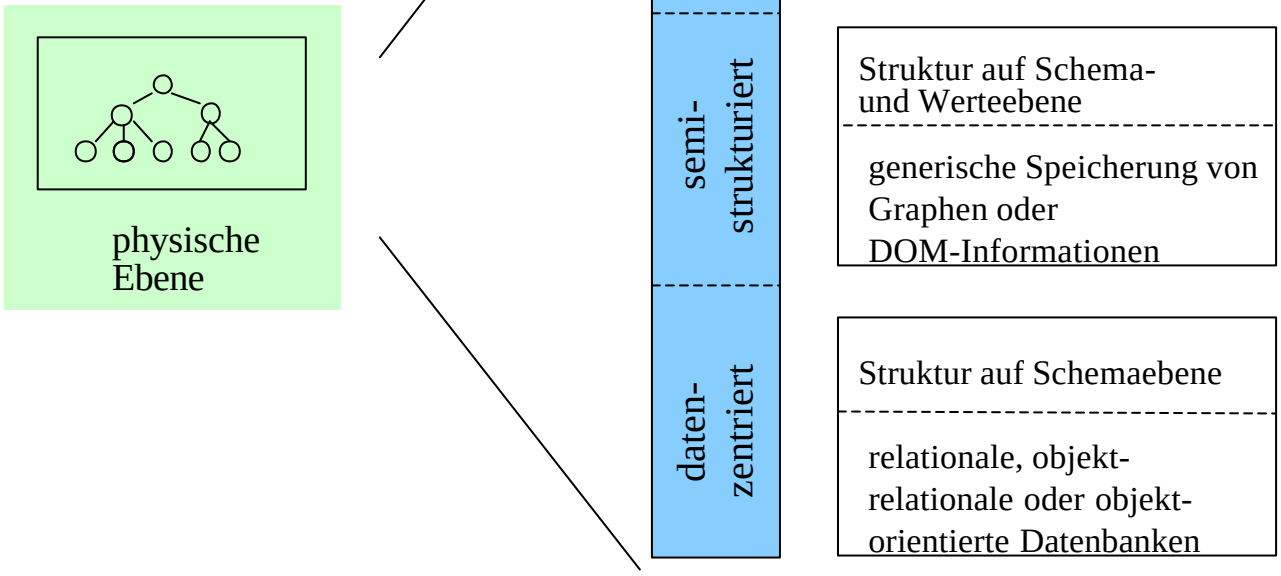
Architektur: physische Ebene

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



Realisierungen für die physische Ebene

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Speicherung der XML-Dokumente als Ganzes und Indizierung (textbasiert native Verfahren)

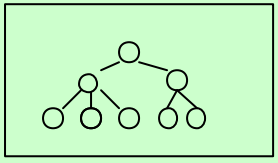
- Volltextindex
- Volltext- und Strukturindex

Speicherung der Graphenstruktur (modellbasierte native Verfahren)

- generische Graphspeicherung
- Speicherung der DOM-Informationen

strukturierte Abbildung auf Datenbanken

- relationale Datenbanken
- objektorientierte und objektrelationale Datenbanken
- Einsatz von benutzerdefinierten Mappingverfahren



physische Ebene

Speicherungsmethoden

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Vorstellung von verschiedenen Methoden
  - Grundprinzip der Speicherung
  - Eigenschaften
  - Anfragemöglichkeiten

- Dabei Klassifikation bekannter Methoden

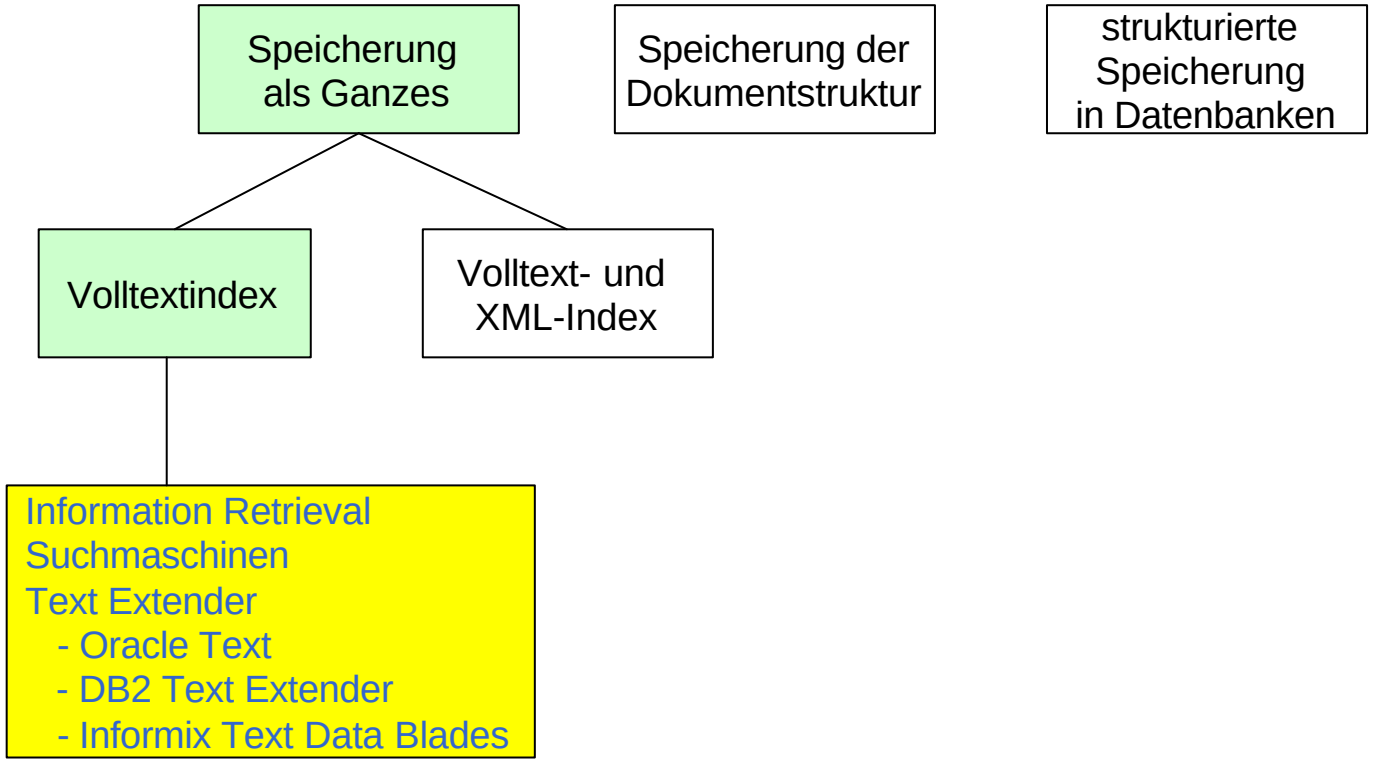
Speicherung von XML-Dokumenten

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



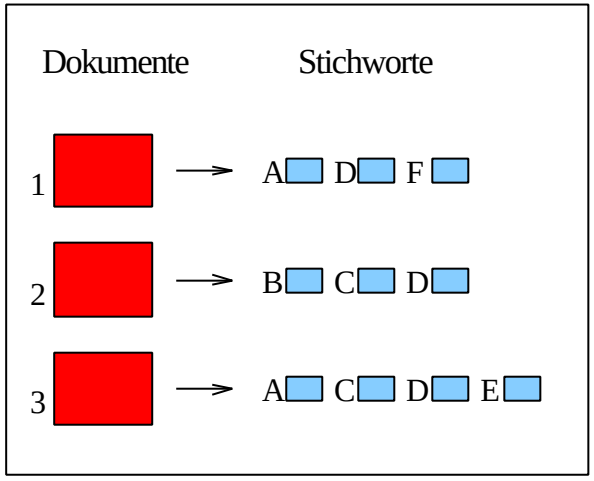
Grundprinzip der invertierten Liste

Einleitung,
Anwendungen
XML
Architektur

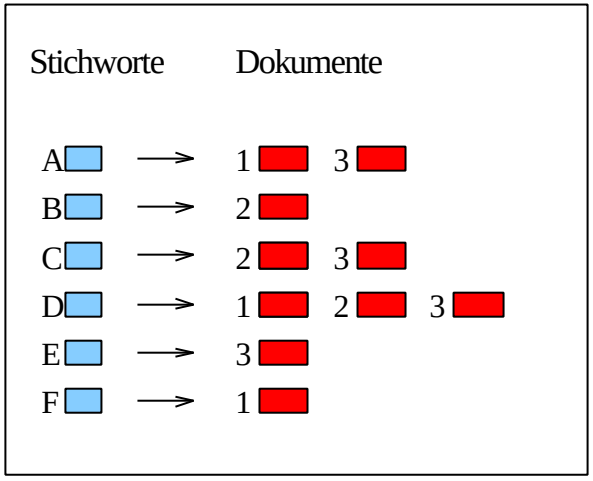
XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



Bestimmung der Stichworte der Dokumente



Invertierte Speicherung:

Speicherung der Stichworte und der zugehörigen Dokumente

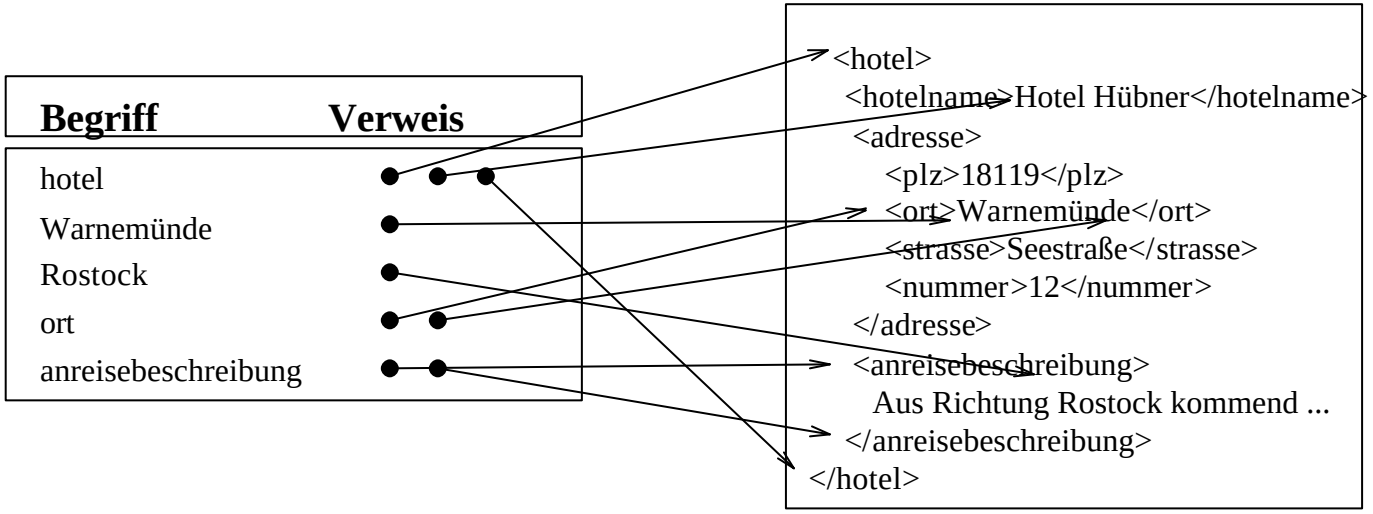
Volltext-Index

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



- bekannte Methode (älter als relationale Datenbanken)
- Verfahren aus dem Bereich der Dokumentverarbeitung

Volltext-Index

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Nicht nur Suche nach exaktem Auftreten der Begriffe möglich, Verwendung von

- Statistischen wortbasierten Verfahren
  - (Stoppwortlisten, Streichen von selten vorkommenden Begriffen)
- Linguistischen Verfahren
  - Wortnormalisierung
  - (Groß-, Kleinschreibung, Getrennt- und Zusammenschreibung, Umlaute),
  - Wortdekomposition durch Regeln (engl.) oder Wörterbücher (dt.)
  - Stammwortreduktion
- Wissensbasierten Verfahren
  - (Einsatz von Ontologien und Thesauren zur Suche nach Synonymen sowie Über- und Unterbegriffen)

Anfragen

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- boolesches Retrieval

  Kombination von Anfrageteilen mit AND, OR oder NOT
  Beispiele:
  - Hotel AND Warnemünde
  - (Hotel OR Pension) AND (Warnemünde OR Rostock)
- linguistische Anfragen
  - Stammwortsuche, phonetische Suche, Ähnlichkeitssuche usw.
- Position der Stichworte im Index erlaubt weitere Anfragen

  Beispiele:
  - Suchbegriff 'Datenbanken' tritt vor 'XML' im Dokument auf
  - der Begriff 'Information' folgt unmittelbar auf 'Retrieval'
  - 'XML' und 'Datenbanken' treten mit einem Abstand von höchstens 5 Wörtern auf

Eigenschaften des Volltext-Indexes

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Schemabeschreibung	nicht erforderlich
Dokumentrekonstruktion	Dokumente bleiben im Original erhalten
Anfragen	Anfragen des Information Retrieval
Weitere Besonderheiten	Volltextfunktionen (SQL-MM) keine Auswertung des XML-Markups
Einsatz	für dokumentzentrierte XML-Anwendungen

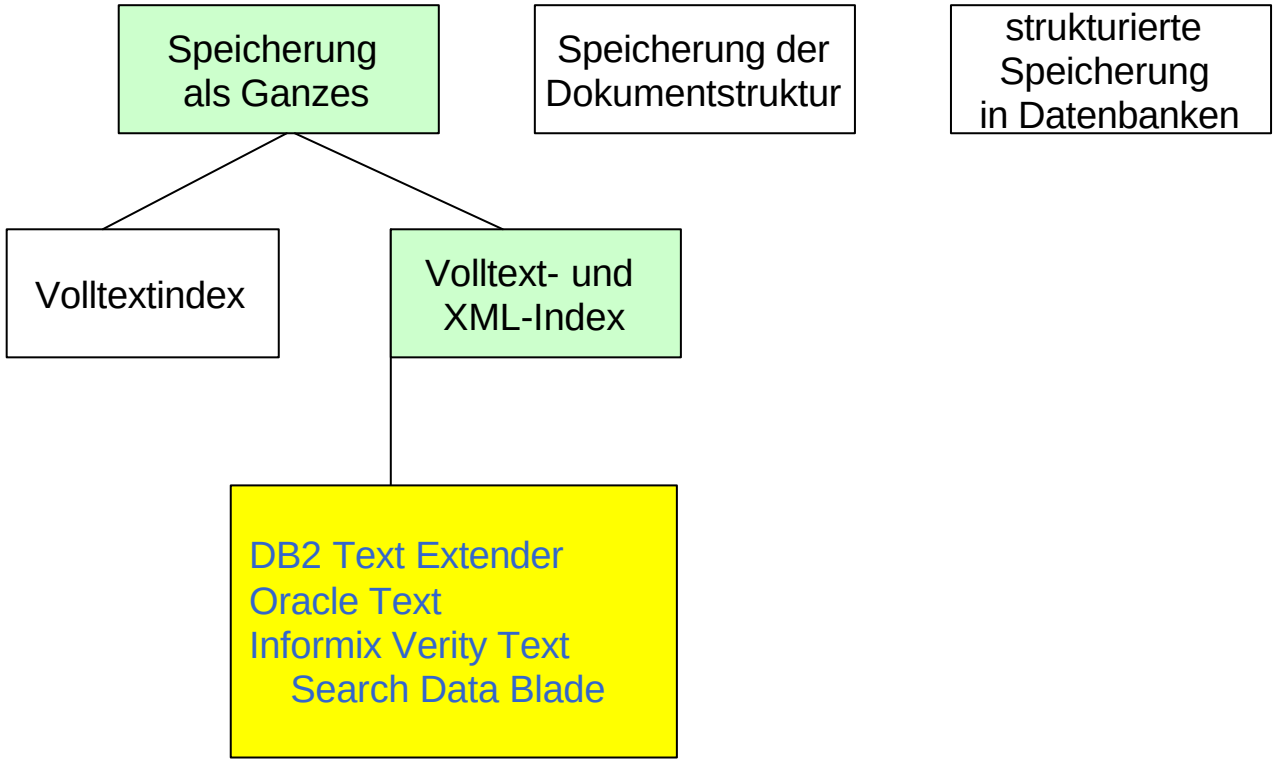
Speicherung von XML-Dokumenten

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



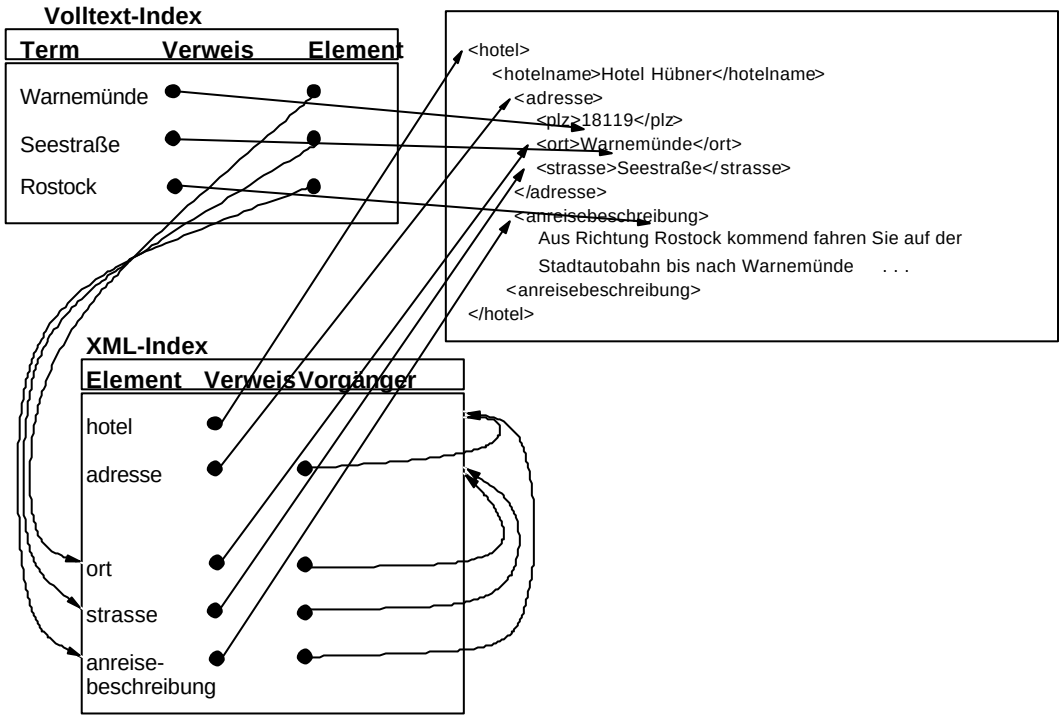
Volltext- und XML-Index

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



- XML - Struktur kann in Anfragen ausgewertet werden

Volltext- und XML-Index

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Strukturinformationen müssen aus dem Index erkennbar und wiederherstellbar sein,
  - Zuordnung des Markup zu den Inhalten sowie
  - Darstellung der hierarchischen Schachtelung und Anordnung der Elemente/Attribute

Detailliertere Darstellung der Indizierungsverfahren im letzten Teil des Tutorials

Eigenschaften des Volltext- und XML-Indexes

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Schemabeschreibung	nicht erforderlich
Dokumentrekonstruktion	Dokumente bleiben im Original erhalten
Anfragen	Anfragen des Information Retrieval Auswertung des Markup in den Anfragen XML-Anfragen möglich
Weitere Besonderheiten	Volltextfunktionen (SQL-MM)
Einsatz	für dokumentzentrierte und semistrukturierte Anwendungen

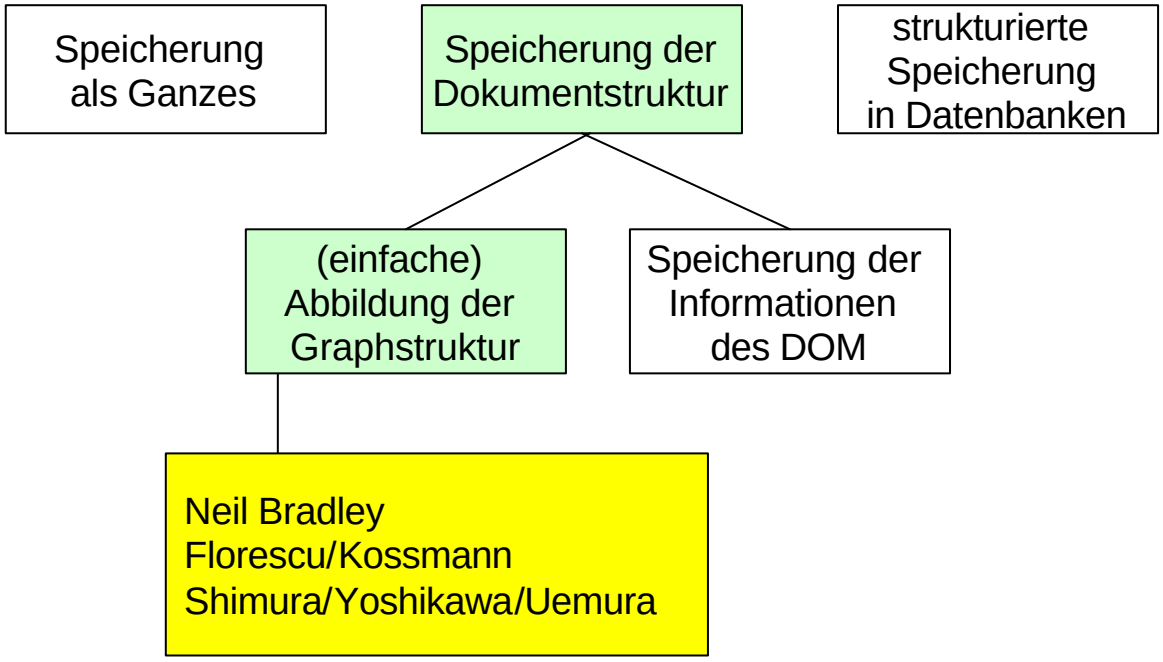
Speicherung von XML-Dokumenten

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



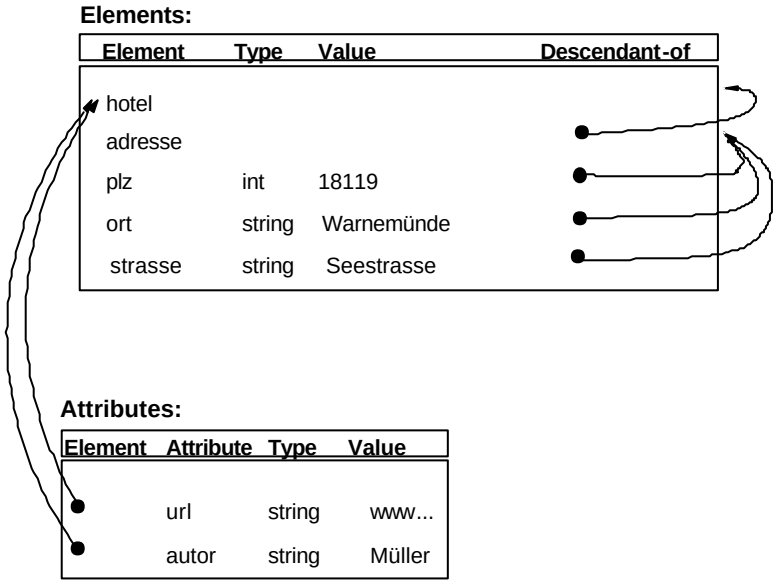
Speicherung der Graphstruktur / 1

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



- generische Speicherung der Struktur des XML-Dokumentes

Speicherung der Graphstruktur /2

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Verwendung von Relationen zur Speicherung von Elementen und Attributen

Elemente:

Elementname	Wert	Verweis auf Vorgänger	Ordnung

Attribute:

Attributname	Wert	Verweis auf Element

- vollständige Wiederherstellung der Struktur ist möglich
- Erweiterungen zur adäquaten Speicherung von Datentypen möglich

Anfragen /1

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

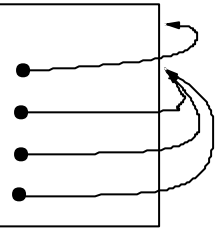
Angepasstes SQL, durch Datenbankschema bestimmt:

Beispiel:

```
(Hotels in Warnemünde)
select a.wert
from  Elemente a, Elemente b
where (a.elementname=
         'hotelname') and
      (b.elementname='ort') and
      (b.wert='Warnemünde') and
      (a.DocID=b.DocID)
```

Elements:

Element	Type	Value	Descendant-of
hotel			
adresse			
plz	int	18119	
ort	string	Warnemünde	
strasse	string	Seestrasse	



Anfragen /2

Einleitung,
Anwendungen
XML
Architektur

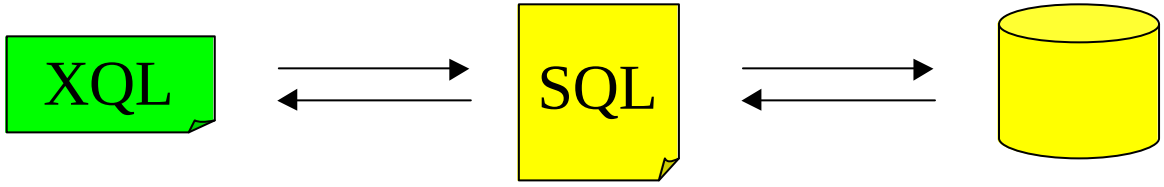
XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



XML-Anfragen sind möglich,



- werden intern auf SQL-Anfragen (unter Berücksichtigung der Speicherstruktur) umgesetzt,
- Ergebnis der XML-Anfrage wird aus Ergebnis der Datenbankanfrage generiert

Eigenschaften der Speicherung der Graphenstruktur

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Schemabeschreibung	zur Speicherung nicht erforderlich
Dokumentrekonstruktion	möglich, aber sehr auswändig
Anfragen	- XML-Anfragen möglich - angepasste DB-Anfrage
Weitere Besonderheiten	Anfragen über vielen Elementen/Attributen sind aufwändig
Einsatz	daten-,dokumentzentrierte und semistrukturierte XML-Dokumente

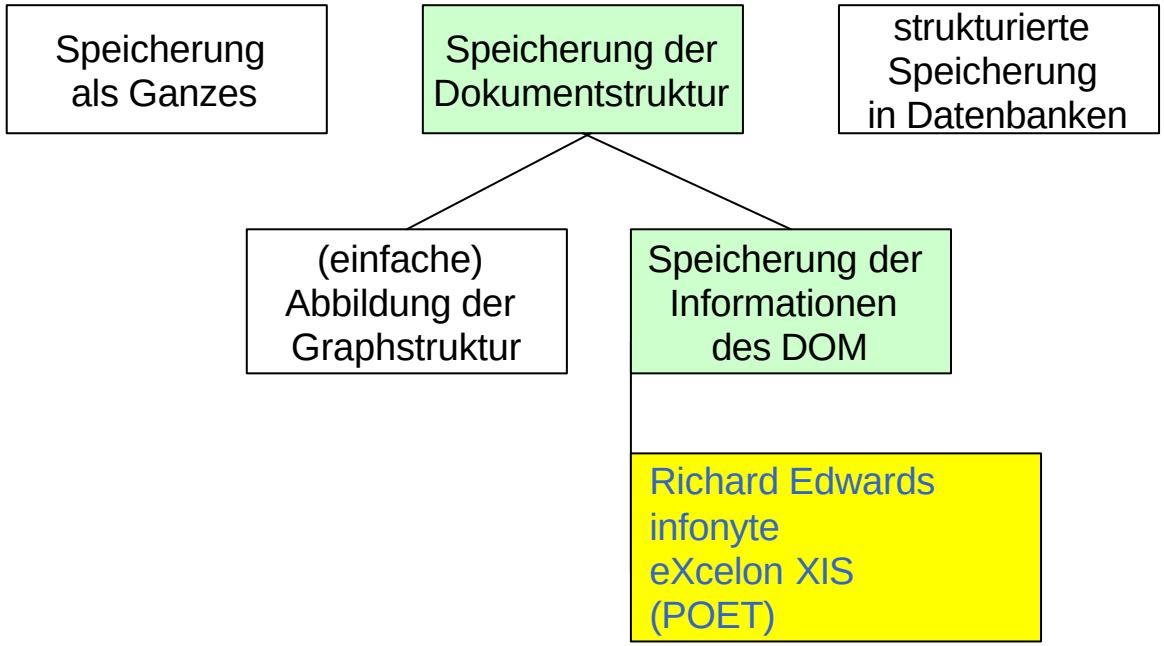
Speicherung von XML-Dokumenten

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



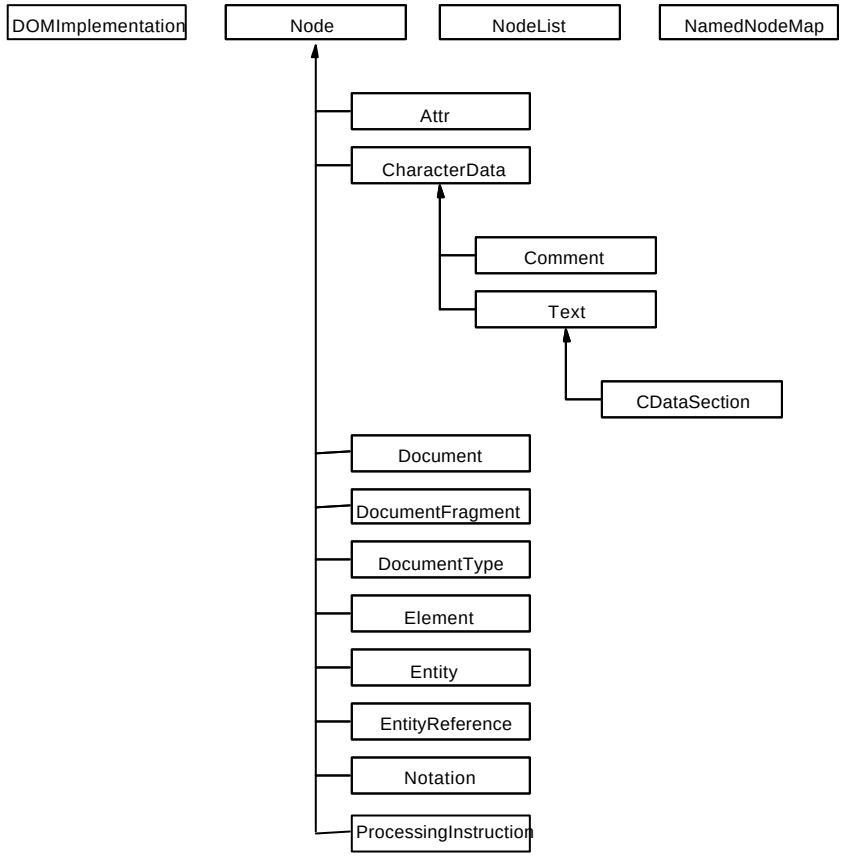
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Einschub: DOM
Document Object Model



Empfehlung des W3C

beschreibt Schnittstellen zum Zugriff auf XML-Dokumente und zur Veränderung von Struktur und Inhalten

die zugrundeliegende Implementierung und Speicherung der XML-Dokumente wird nicht definiert !

Methoden der Klasse: Node

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Fast alle Dokumentbestandteile basieren auf dieser Klasse

- Die Klasse Node enthält
  - Methoden zur Identifikation des Knotentyps
  - Methoden zur *Navigation* durch die Dokumentstruktur
  - Methoden zur *Manipulation* der Dokumentstruktur

DOM (Document Object Model) – Navigation durch das Dokument

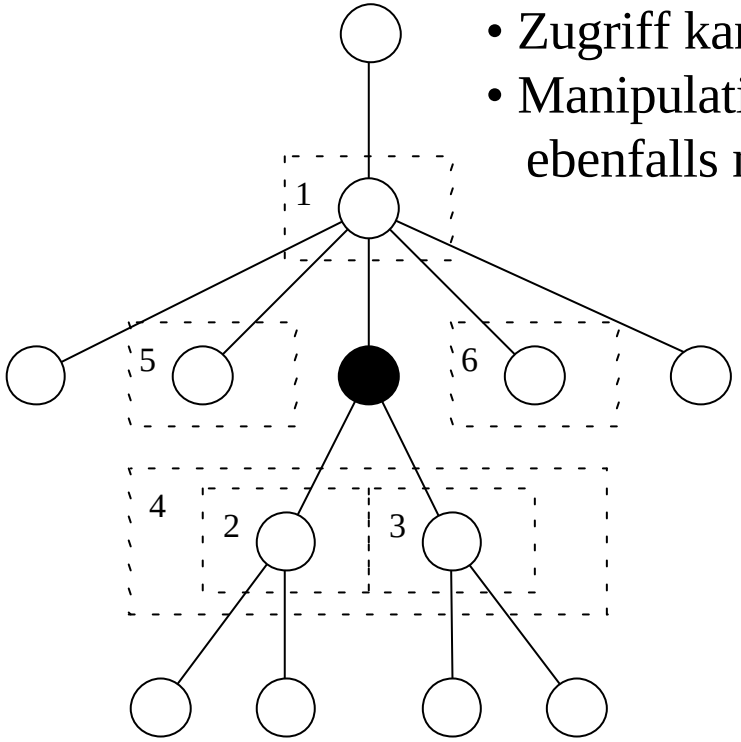
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- XML-Dokument als Baum dargestellt,
- Zugriff kann durch Navigation erfolgen
- Manipulation des Dokumentstruktur ebenfalls möglich



Ausgehend vom Knoten ●
liefern folgende Methoden der Klasse Node die Knoten bzw. Knotenlisten als Ergebnis:

1 - getParentNode()
2 - getFirstChild()
3 - getLastChild()
4 - getChildren()
5 - getPreviousSibling()
6 - getNextSibling()

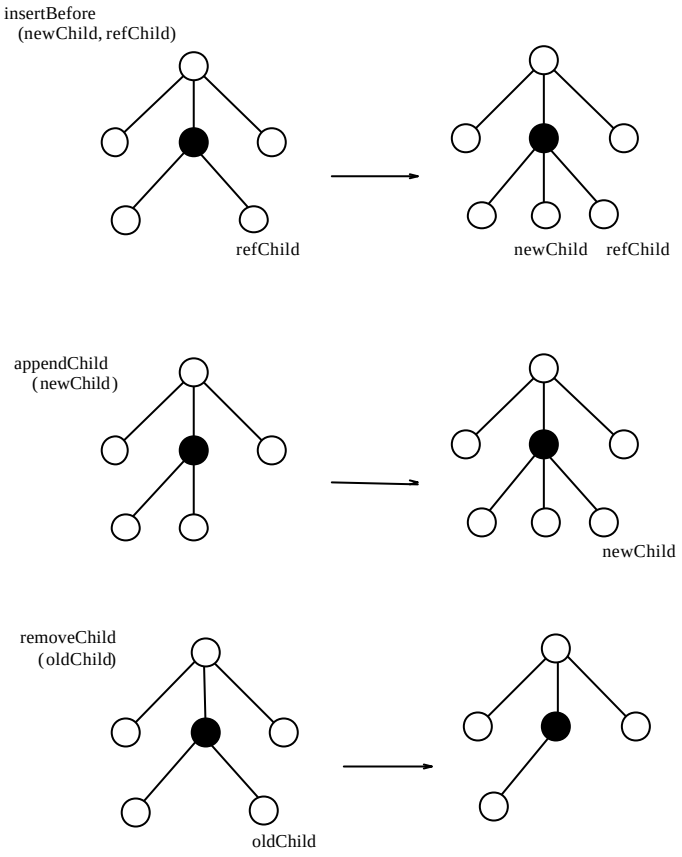
DOM – Manipulation der Struktur

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



Methoden zum
Löschen, Ergänzen,
Verschieben sowie Verändern
von Dokumentanteilen

Bestandteile der Schnittstelle
Node

Die Klasse: Element

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Methoden der Klasse Elemente
  - getTagName() liefert den Tagnamen eines Elementes
  - getAttribute(name) liefert das angegebene Attribut
  - setAttribute(name, value) ermöglicht das Verändern und Ergänzen von Attributen
  - removeAttribute(name) löscht das spezifizierte Attribut
  - getElementsByTagName(tagname) liefert alle Elemente des Dokumentes, die den angegebenen Elementnamen haben, bei Angabe von
  - getElementsByTagName(*) werden alle Elemente zurückgegeben.

Methoden der Klasse Attribute

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Attributinformationen können über die folgenden Methoden erfragt werden:
  - getName() gibt den Attributnamen zurück
  - getValue() liefert den Attributwert und
  - setValue(value) setzt den Attributwert.

Methoden der Klasse Character Data

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Zum Auslesen, Verändern und Setzen von Textanteilen
- Methoden der Klasse Character Data:
  - getLength()
  - getData() liefert den vollständigen Text
  - substringData(start,count)
  - appendData(text)
  - replaceData(offset,count,text)
  - insertData(offset,text)
  - deleteData(offset,count)

Speicherung basierend auf DOM /1

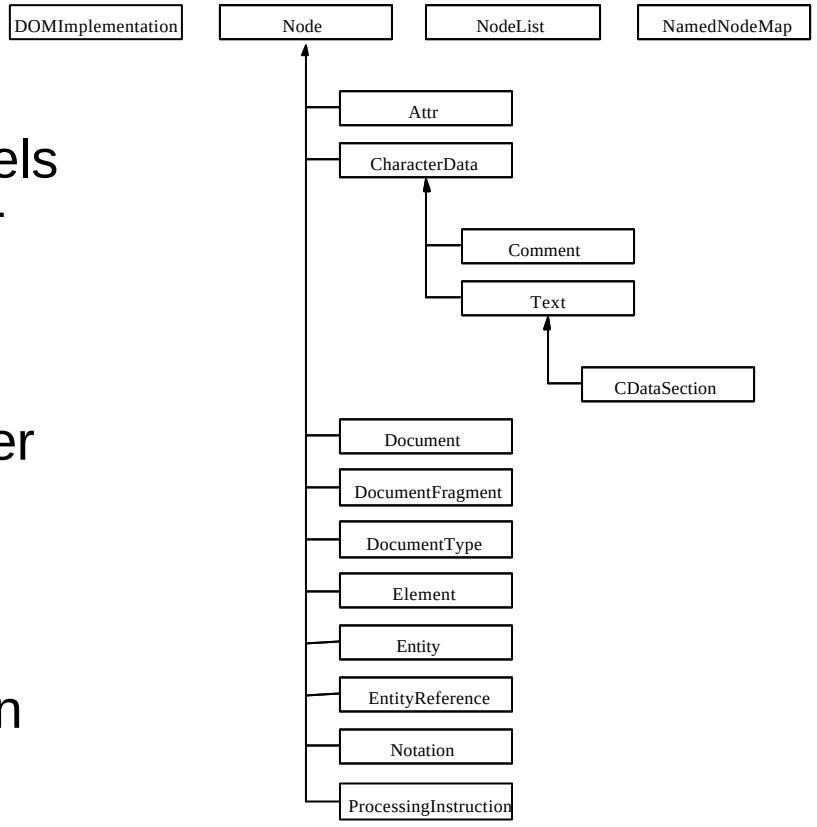
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Informationen des Document Object Models bilden das Schema der Datenbank
- Verwendung relationaler oder objektorientierter Datenbanken oder
- Entwicklung eigener Speicherungsstrukturen



Speicherung basierend auf DOM /2

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Methoden der Klasse Node:
- getChildren()
- getFirstChild()
- getNextSibling()
- getNodeType()
- getParentNode()
- getPreviousSibling()
- hasChildren()

NodeID	NodeType	DocID	ParentNode

PreviousSibling	NextSibling	FirstChild

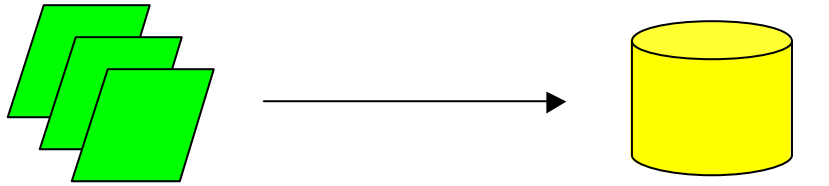
Methoden der Klasse Element:
- getAttributes()
- getElementsByTagName(String)
- getTagName()

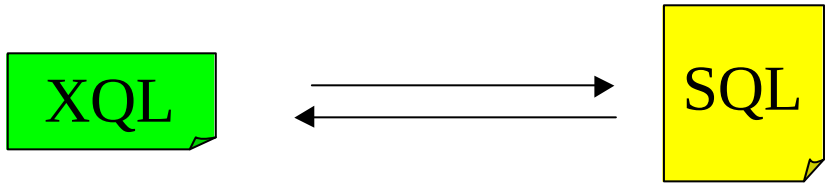
NodeID	TagName

Methoden der Klasse Attribut:
- getName()
- getValue()

NodeID	ElementID	AttributName	AttributValue

Anfragen







- DOM-Methoden
- XML-Anfragen

- Angepasste Datenbankanfragen (unter Kenntnis der Speicherungsstruktur)

Eigenschaften der auf DOM basierenden Speicherung

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Schemabeschreibung	Zur Speicherung nicht erforderlich
Dokumentrekonstruktion	möglich, aber auswändig
Anfragen	XML-Anfragen möglich angepasste Datenbankanfragen
Weitere Besonderheiten	standardisierte und allgemein akzeptierte Schnittstelle
Einsatz	für daten- und dokumentzen-trierte, sowie semistrukturierte XML-Anwendungen

Speicherung von XML-Dokumenten

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

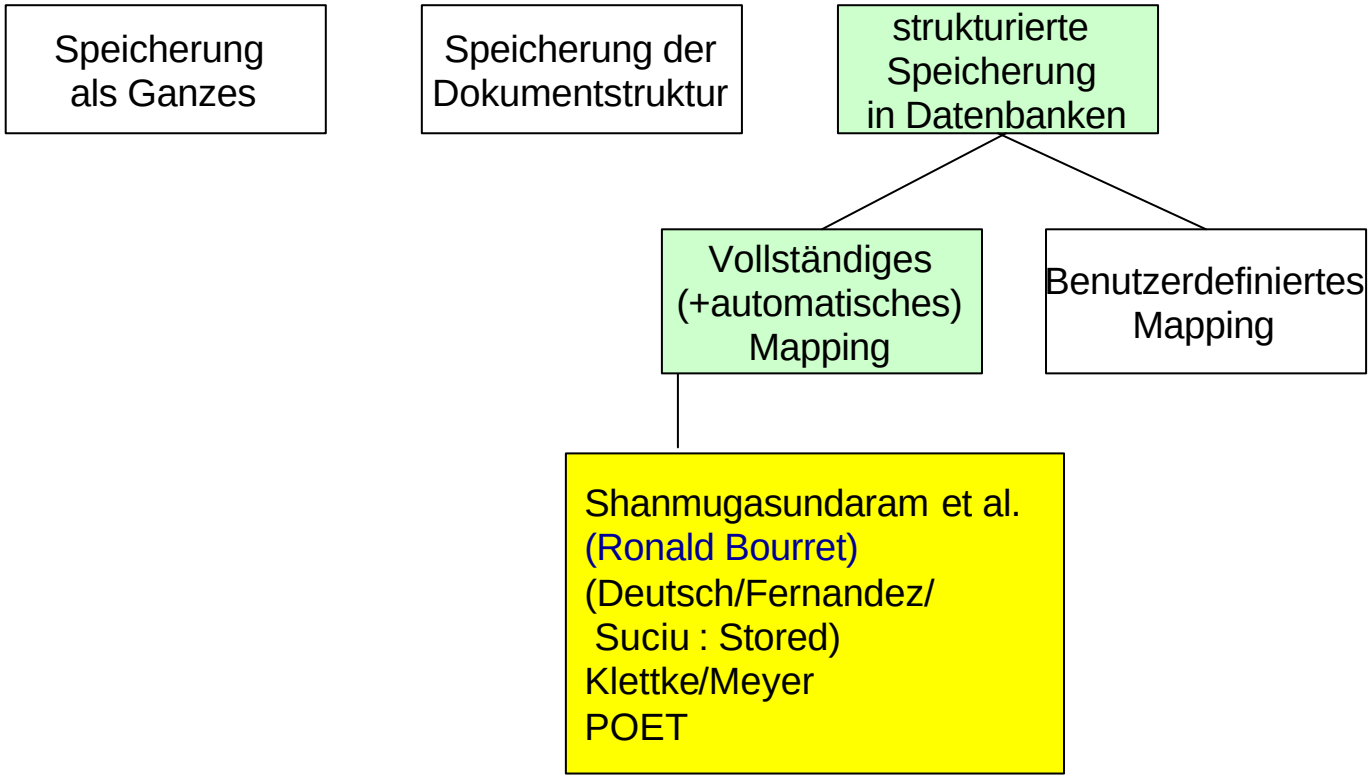


Abbildung der XML-Struktur auf Struktur relationaler Datenbanken

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

XML-Dokument

```
<hotel url="www.hotel-huebner.de">
  <hotelname>Hotel Hübner</hotelname>
  <adresse>
    <ort>Warnemünde</ort>
    <strasse>Seestraße</strasse>
    ...
  </adresse>
  <preise>
    <einzelzimmer>198</einzelzimmer>
    ...
  </preise>
</hotel>
```

Hotel:

HotelID	Hotelname	Adresse	Preise
H0001	Hotel Hübner	A0001	P0001

Adresse:

AdresseID	Ort	Strasse	...
A0001	Warnemünde	Seestraße	

Preise:

PreiseID	Einzelzimmer	...
P0001	198	

- DTD ist erforderlich
- typgerechte Speicherung

Abbildung der XML-Struktur auf objektorientierte Datenbanken

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

XML-Dokument

```
<hotel url="www.hotel-huebner.de">
  <hotelname>Hotel Hübner</hotelname>
  <adresse>
    <ort>Warnemünde</ort>
    <strasse>Seestraße</strasse>
    ...
  </adresse>
  <preise>
    <einzelzimmer>198</einzelzimmer>
    ...
  </preise>
</hotel>
```

Hotel:

HotelID	Hotelname	Adresse			Preise	
		Ort	Strasse	...	einzelzimmer	...
H0001	Hotel Hübner	Warnemünde	Seestraße		198	

- DTD ist erforderlich
- typgerechte Speicherung
- Datenbanken mit vielen Nullwerten

Erzeugen des Datenbankschemas für eine DTD (straight-forward) /1

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Elemente:

XML-Element	→	Attribut einer Relation
Sequenz von Elementen	→	Attribute einer Relation
Alternative von Elementen	→	Attribute einer Relation
Element mit Quantifizierer ?	→	Attribut mit Nullwerten
Elemente mit Quantifizierer +,*	→	Menge oder Liste von Attributen (SET OF, LIST OF)
Geschachtelte Elemente	→	TUPLE OF

Erzeugen des Datenbankschemas für eine DTD (straight-forward) /2

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Attribute:

XML-Attribut	⇒	Attribut einer Relation
IMPLIED	⇒	Nullwert erlaubt
REQUIRED	⇒	Nullwert nicht erlaubt
Defaultwert	⇒	Defaultwert

Weiteres Beispiel /1

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

```
<!ELEMENT book (front, body, references)>
<!ATTLIST   book isbn CDATA #REQUIRED>
<!ELEMENT front  (title, author+, edition?, publisher)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (first, second, e-mail?)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT second (#PCDATA)>
<!ELEMENT e-mail (#PCDATA)>
```

book:

isbn	front						body	references
	title	author			edition	publisher		
		first	second	e-mail				

Weiteres Beispiel / 2

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

XML-Document:

```
<!ELEMENT book (front, body, references)>
<!ATTLIST   book isbn CDATA #REQUIRED>
<!ELEMENT front  (title, author+,
                          edition?, publisher)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (first, second, e-mail?)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT second (#PCDATA)>
<!ELEMENT e-mail (#PCDATA)>
```

Informix-Syntax:

```
CREATE TABLE book
( isbn VARCHAR(20) NOT NULL,
  front ROW
    ( title VARCHAR(100) NOT NULL,
      author LIST
        (ROW
          ( first VARCHAR(30) NOT NULL,
            second VARCHAR(50) NOT NULL,
            email VARCHAR(30)))  NOT NULL,
      edition VARCHAR(20),
      publisher VARCHAR(40) NOT NULL),
    body ...
    references ...
);
```

Problemfall: Alternativen /1

Beispiel: <!ELEMENT unterkunft (hotel | pension | campingplatz)*>

drei Speicherungsvarianten sind möglich

1. alle Alternativen in einer Relation

Hotel			Pension				Campingplatz		
Kategorie	Hotel-name	...	Pensions-name	Zimmer	Aus--stattung	Preise	Klassifi-kation	..	..

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Problemfall: Alternativen /2

Beispiel: <!ELEMENT unterkunft (hotel | pension | campingplatz)*>

2. Aufspaltung in separate Relationen

Hotel

Kategorie	Hotel-name	...

Pension

Pensions-name	Zimmer	Aus--stattung	Preise

Campingplatz

Klassifi-kation	..	..

Problemfall: Alternativen /3

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Beispiel: <!ELEMENT unterkunft (hotel | pension | campingplatz)*>

3. Verwendung eines Attributes vom Typ XML

unterkunft
<pension> <pensionsname>Zum Kater</pensionsname> <zimmer>...</zimmer> ... </pension> <hotel> <kategorie>4 </kategorie> <hotelname>Strand Hotel Hübner </hotelname> </hotel>

Problemfall: Rekursionen

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

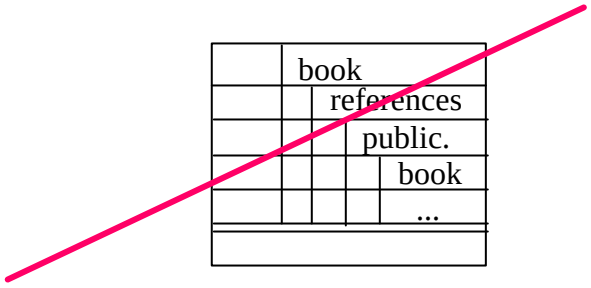
XML-Speicherung
XML-Generierung

Indizierung
Systeme

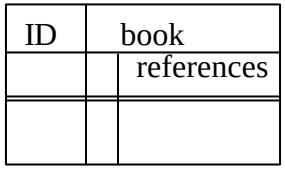
- Rekursionen (in DTDs)
- Vorgehensweise:
  - Markieren der Knoten
  - Aufspaltung in separate Relationen
  - Verwendung von Referenzen (primary/foreign key)
- Beispiel:

```
<!ELEMENT publications  (book | article | conference)*>
<!ELEMENT book  (front, body, references)>
<!ELEMENT references (publications+)>
```

publications:



publications:



Problemfall: Mixed Content

```
<anreisebeschreibung>
    Sie können unser Haus auf verschiedenen Wegen
    erreichen:
    <bahn>per Bahn: 1 km ab Bahnhof Warnemünde</bahn>
    <auto>per Auto: 19 km ab Autobahn A19
          Rostock--Berlin</auto>
Sie finden uns direkt an der Uferpromenade.
</anreisebeschreibung>
```

Ordnung	#PCDATA	Bahn	Auto
1	Sie können unser Haus auf verschiedenen Wegen erreichen:		
2		per Bahn: 1 km ab Bahnhof Warnemünde	
3			per Auto: 19 km ab Autobahn A19 Rostock—Berlin
4	Sie finden uns direkt an der Uferpromenade.		

Problemfälle: Mixed Content, ANY

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Abbildung in Relationen nur sehr schlecht geeignet
- geeignet ist in diesem Fall: XML-Datentyp

...	anreisebeschreibung

...	`Sie können unser Haus auf verschiedenen Wegen erreichen:`
	`  <bahn>per Bahn: 1 km ab Bahnhof  Warnemünde</bahn>`
	`  <auto>per Auto: 19 km ab Autobahn A19 Rostock--Berlin</auto>`
	`Sie finden uns direkt an der Uferpromenade.`
	`</anreisebeschreibung>`

- Gar nicht darstellbar: Inhaltsmodell ANY

Anfragen

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Datenbankanfragen (Struktur der Datenbanken muss bekannt sein)

- XML-Anfragen

  Prozess der Abbildung der XML-Dokumente auf Datenbanken muss protokolliert sein,

  damit Datenbankanfragen für XML-Anfragen generiert und Ergebnisse für die XML-Anfragen generiert werden können.

Vor- und Nachteile

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Vorteile: bei der Speicherung strukturierter Daten

- Anfragen, Datentypen, Aggregatfunktionen, Sichten
- Integration in andere Datenbanken

Nachteile: bei der Speicherung semi- und unstrukturierter Daten

- großes Schema, schwach gefüllte Datenbanken, viele Nullwerte
- Keine flexiblen Datentypen, Speicherung von Alternativen problematisch
- Fehlende Information Retrieval Anfragen, keine Volltextoperationen möglich

Strukturierte Speicherung in Datenbanken

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Schemabeschreibung	Zur Speicherung erforderlich
Dokumentrekonstruktion	Nur eingeschränkt möglich (Protokollierung des Abbildungsprozesses)
Anfragen	Datenbankanfragen XML-Anfragen möglich
Weitere Besonderheiten	Föderationen mit bestehenden Datenbanken möglich
Einsatz	für datenzentrierte XML-Anwendungen

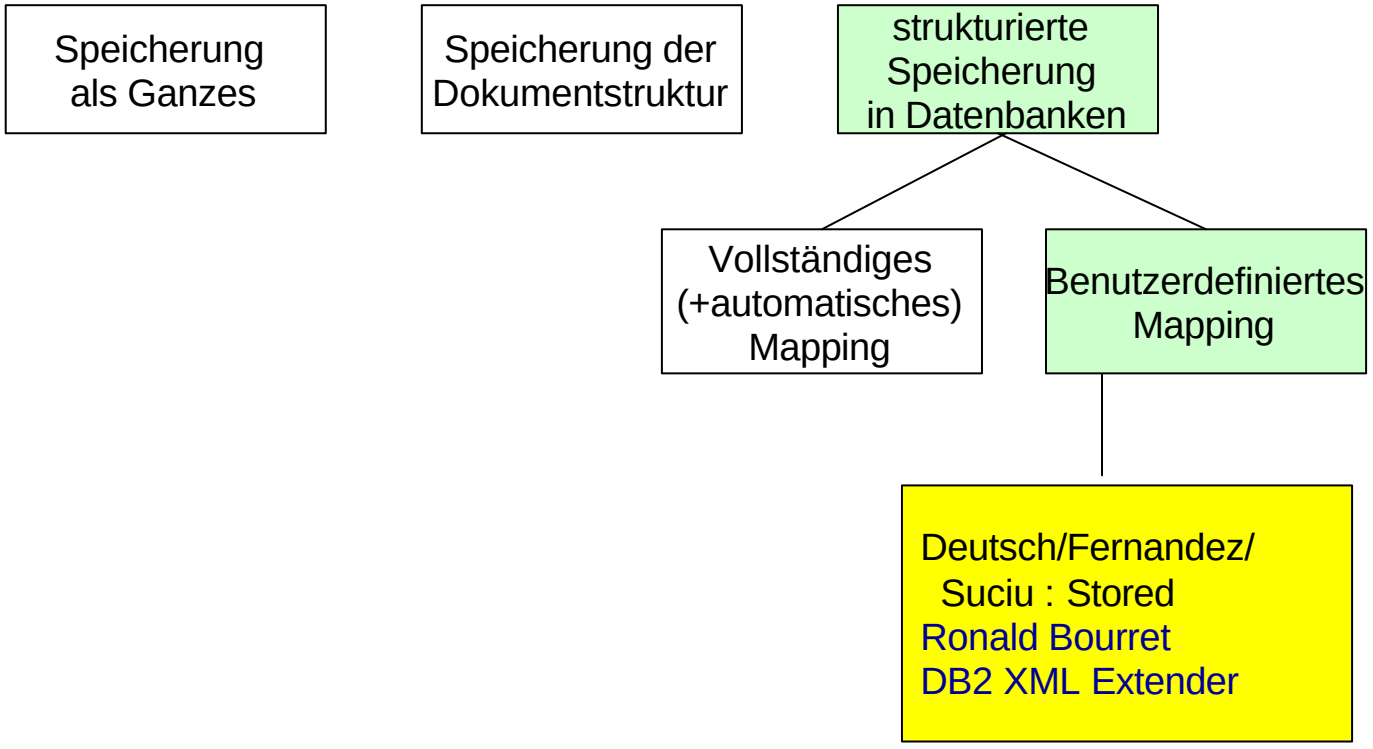
Speicherung von XML-Dokumenten

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



Benutzerdefiniertes Mapping /1

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Bei allen bisherigen Methoden konnte die Art der Speicherung nicht beeinflusst werden

- Jetzt folgt eine flexible Methode: anwenderdefiniertes Mapping
- Struktur der XML-Dokumente und Datenbankschema können unabhängig voneinander entworfen werden (und relativ autonom sein)
- Auch möglich: Speicherung von XML-Dokumenten in existierende Datenbanken

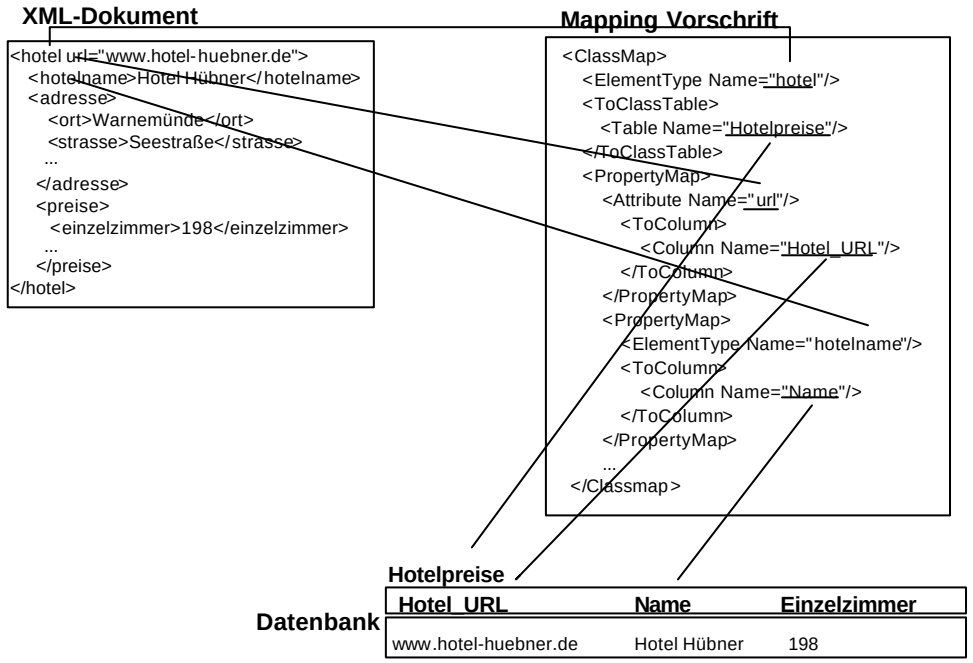
Benutzerdefiniertes Mapping /2

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



Mappingvorschrift stellt die Verbindung zwischen XML-Schema und Datenbanken dar

Mappingvorschrift

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Beispiel nach Ronald Bourret

```
<ClassMap>
     <ElementType Name="hotel"/>
     <ToClassTable>
         <Table Name="Hotelpreise"/>
     </ToClassTable>
     <PropertyMap>
         <Attribute Name="url"/>
         <ToColumn>
             <Column Name="Hotel_ULR"/>
         </ToColumn>
     </PropertyMap>
...
 </ClassMap>
```

Verbindung zwischen Elementen und Relationen

Verbindung zwischen Elementen/Attributen und Datenbank-attributen

Anfragen

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Datenbankanfragen (Struktur der Datenbanken muss bekannt sein)

- XML-Anfragen nur sehr eingeschränkt

  Prozess der Abbildung der XML-Dokumente auf Datenbanken muss protokolliert sein, und alle für die Anfrage relevanten Daten müssen gespeichert sein

Eigenschaften

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Schemabeschreibung	Zur Speicherung erforderlich
Dokumentrekonstruktion	Meist nicht möglich (Voraussetzung: Protokollierung des Abbildungsprozesses, vollständige Abb.)
Anfragen	Datenbankanfragen
Weitere Besonderheiten	Integration in bestehende Datenbanken möglich, XML-Dokumente und DB voneinander unabhängig
Einsatz	für datenzentrierte XML-Anwendungen

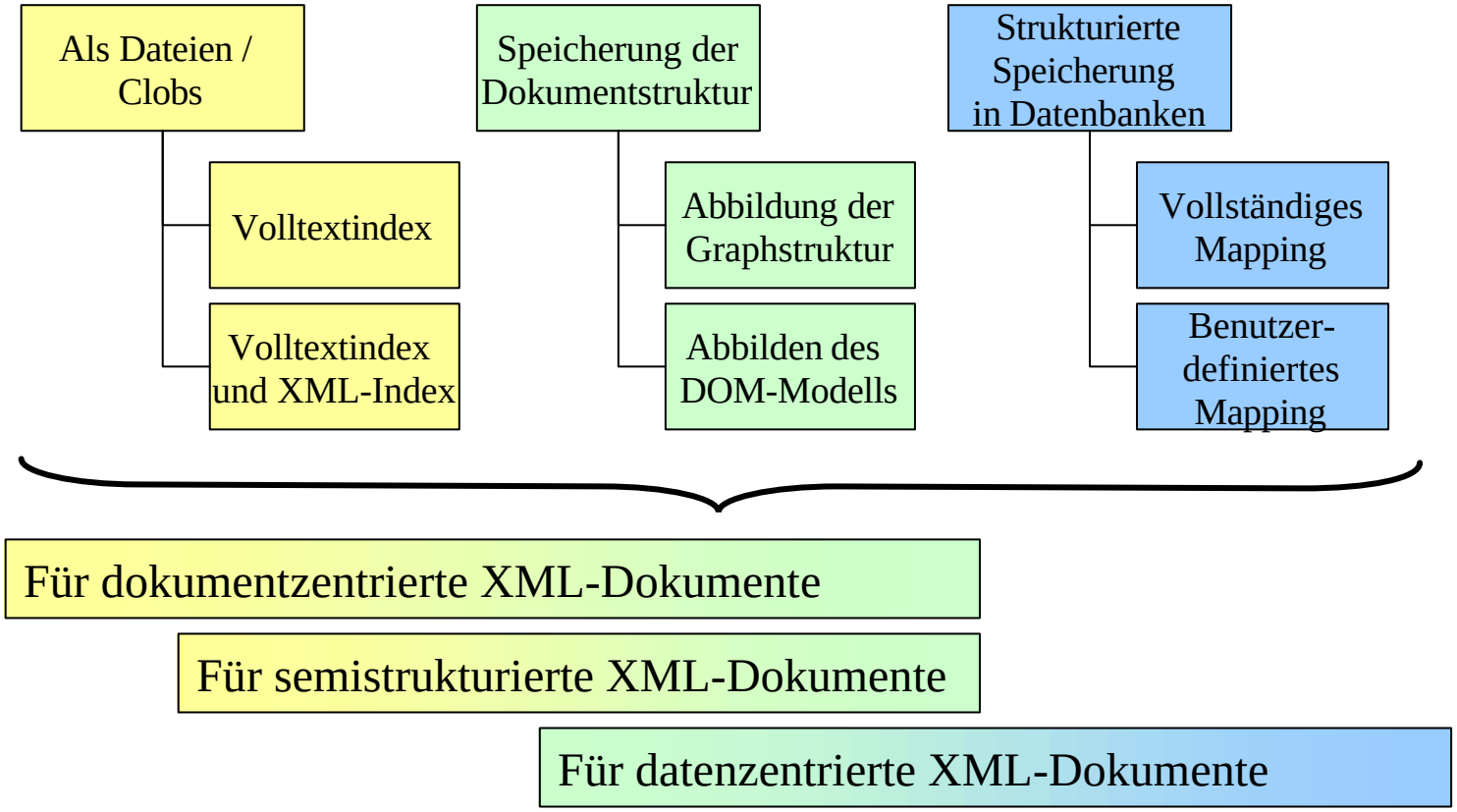
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Speicherung von XML-Dokumenten



Hybride Speicherung – Motivation

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Es gibt eine Reihe von Methoden zur Speicherung von XML-Dokumenten.

Welche eignet sich für XML-Dokumente, die sowohl daten- als auch dokumentzentriert sind?

- Algorithmus zur Bestimmung von daten- und dokumentzentrierten Anteilen eines XML-Dokument
- Entsprechende Speicherung

Hybride Ansätze

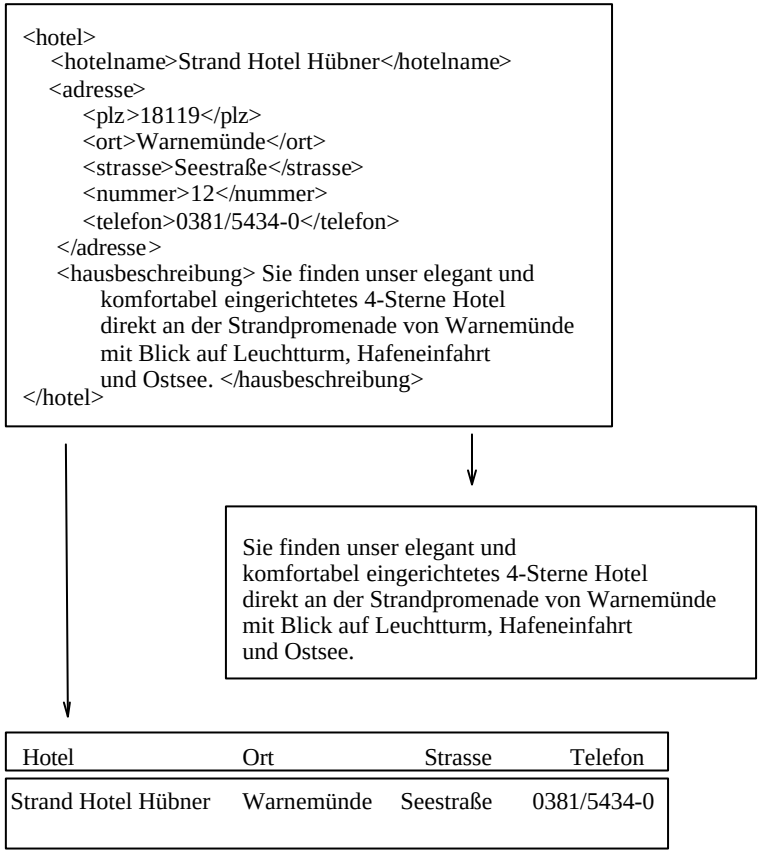
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Auswahl unterschiedlicher Speicherungsmethoden für verschiedene Dokumentanteile



Hybride Datenbanken

= objektrelationale Datenbanken mit Attributen vom Typ XML

Beispiel:

front			content
title	**author**	**publisher**	
XML companion	Neil Bradley	Addison Wesley	XML builds on the principles of two existing languages, <emph>HTML</emph> and <emph>SGML</emph> to create a simple mechanism .. XML builds on the principles of two existing languages, <emph>HTML</emph> and <emph>SGML</emph> to create a simple mechanism ..

- „strukturierte“ Anteile des XML-Dokumentes als objektrelationale Strukturen
- „unstrukturierte“ Anteile als Attribute vom Typ XML

Algorithmus zum Erstellen des Datenbankschemas

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Erkennen der regulären und irregulären Anteile,

Separieren dieser Teile,

Erstellen eines Datenbankentwurfes:

- Aufbau eines DTD-Graphen
- Bestimmen eines Maßes für die Signifikanz w jedes Knotens
- Festlegen eines Limits, das die Granularität des entstehenden Datenbankschemas angibt
- Ableitung der Attribute vom Typ XML
- Abbildung der strukturierten Anteile auf objektrelationale Datenbanken (wie zuvor angegeben)

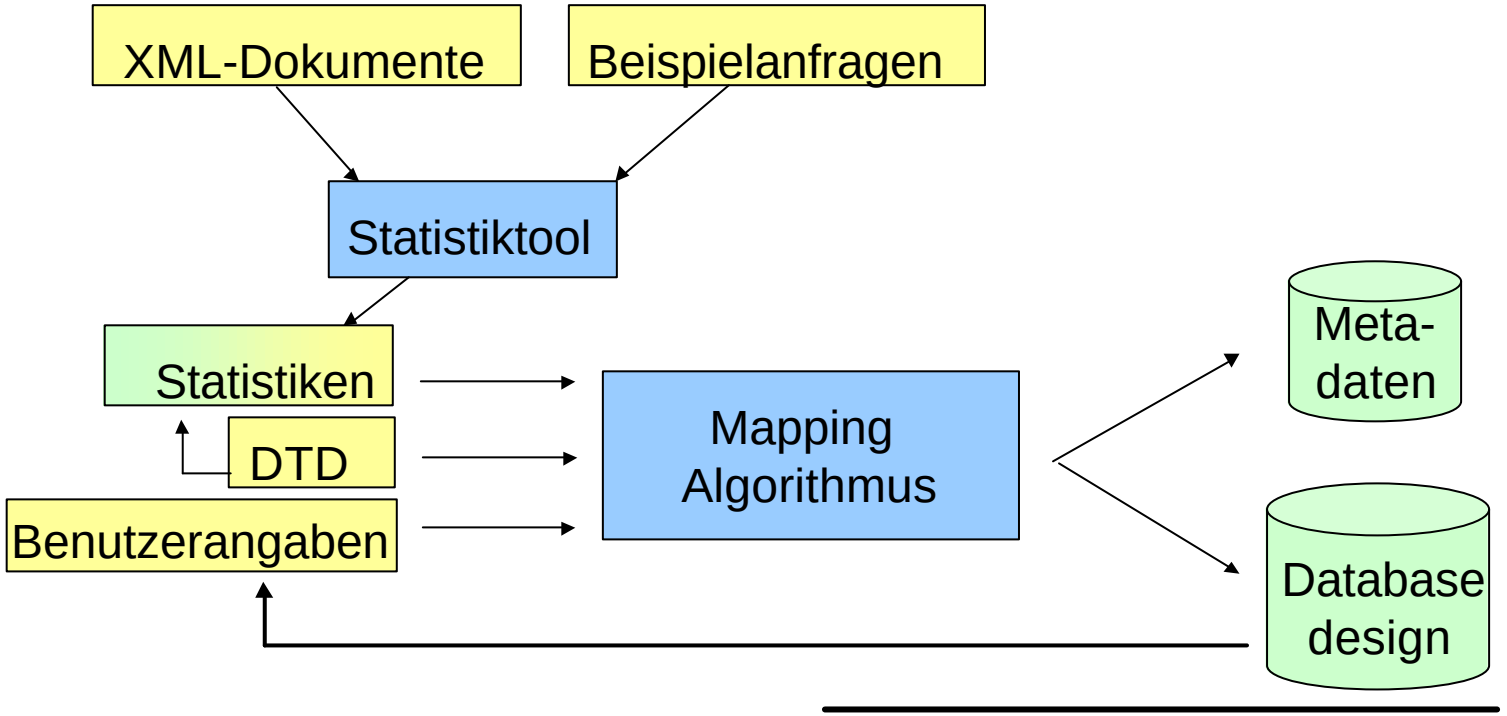
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

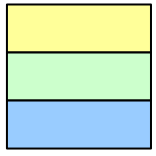
XML-Speicherung
XML-Generierung

Indizierung
Systeme

Architektur



Automatischer oder
semiautomatischer Algorithmus



Eingaben
Ausgaben
Programmkomponenten

Bestimmen der Signifikanz /1

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Beeinflusst durch:

- DTD
  - Auswertung Inhaltsmodell und Quantifizierer
- Mengen von XML-Dokumenten
  - Vorkommenshäufigkeit von Elementen und Attributen
- Beispielanfragen
  - Welche Arten von Anfragen, wie häufig werden welche Elemente angefragt
  - (Im Gegensatz zur Information, wie häufig welche Elemente und Attribute in den Dokumenten auftreten)

Bestimmen der Signifikanz /2

Einleitung,
Anwendungen
XML
Architektur

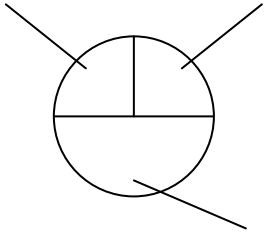
XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

$$w = \frac{1}{4} * w_D + \frac{1}{4} * w_Q + \frac{1}{2} * w_S$$

w_D – aus den XML-**D**okumenten abgeleitet



w_Q – aus den **Q**ueries abgeleitet

w_S – aus der DTD **S**truktur abgeleitet

Visualisierung der Gewichte durch Verwendung einer Grau-Skala

DTD Graph

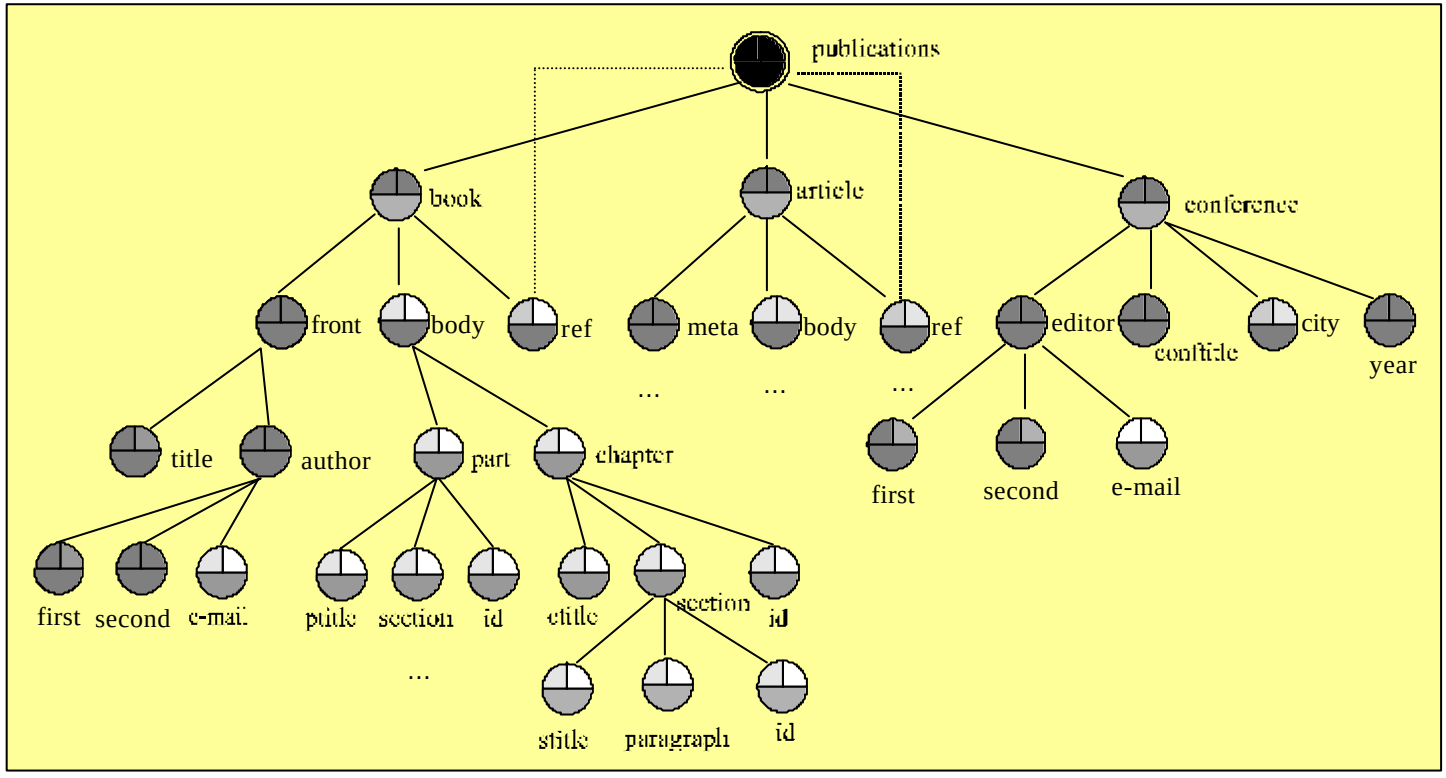
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Beispiel DBLP (Michael Ley):



Festlegen des Limits

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

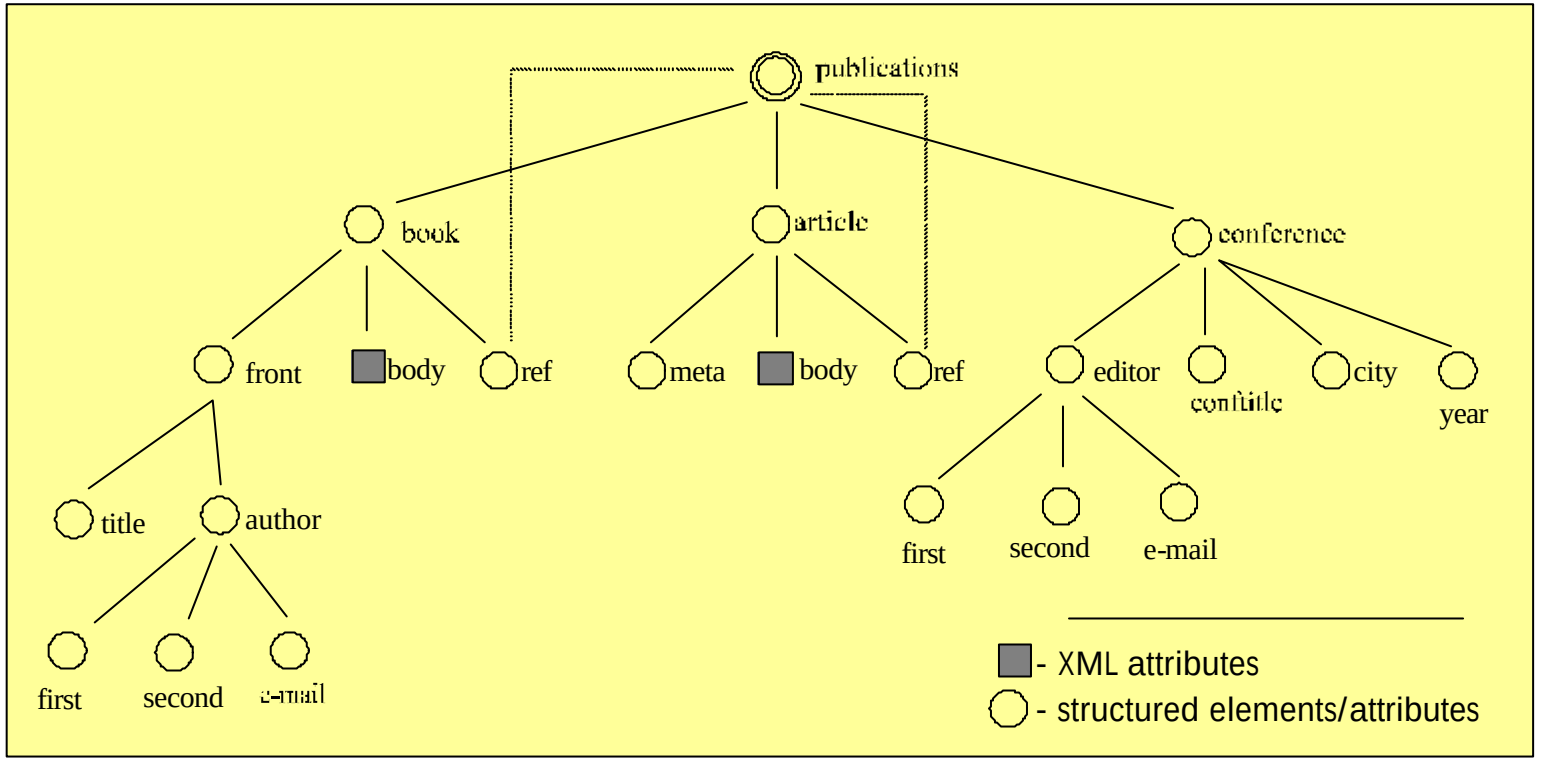
XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Das Limit bestimmt,
  - welche Elemente/Attribute strukturiert gespeichert werden und
  - welche als XML-Attribute gespeichert werden.
- Wahl des Limits ist abhängig von
  - der Anwendung
  - der Unterstützung, die bei der strukturierten Speicherung erfolgt, sowie
  - der Unterstützung, die der XML Datentyp liefert (Anfragerealisierung auf dem Datentyp)
- niedriges Limit - viele Anteile strukturiert gespeichert
- hohes Limit - große Anteile als XML Typ gespeichert

Abgeleiteter DTD Graph

für das Beispiel DBLP von Michael Ley:



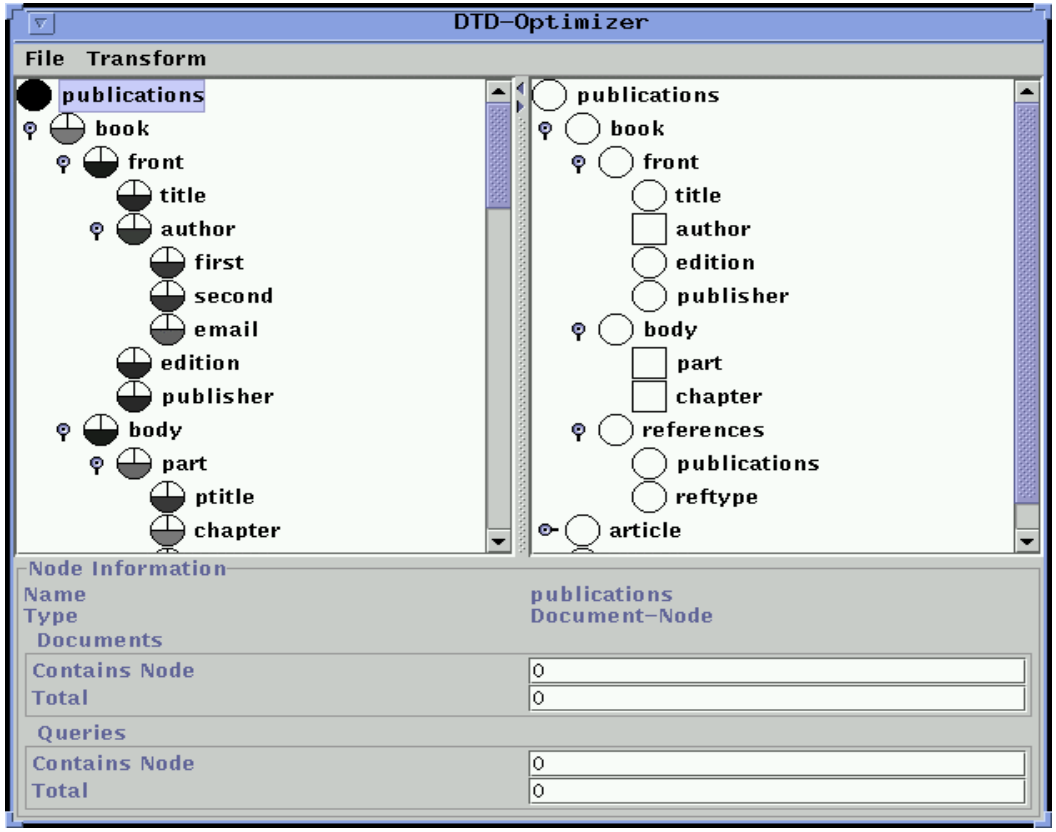
Prototyp

Einleitung,
Anwendungen
XML
Architektur

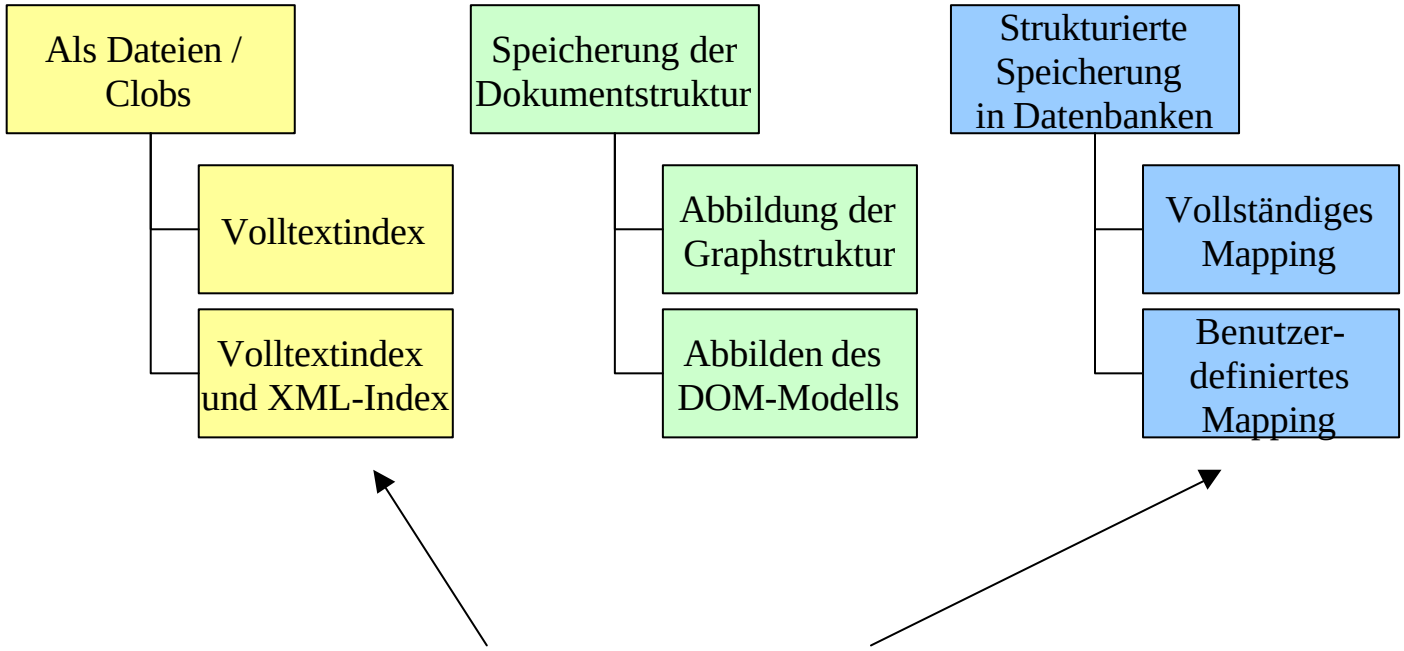
XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



Integration der hybriden Speicherung in die Klassifikation



- Erkennen von daten- und dokumentzentrierten Anteilen
- Speicherung mit verschiedenen Verfahren

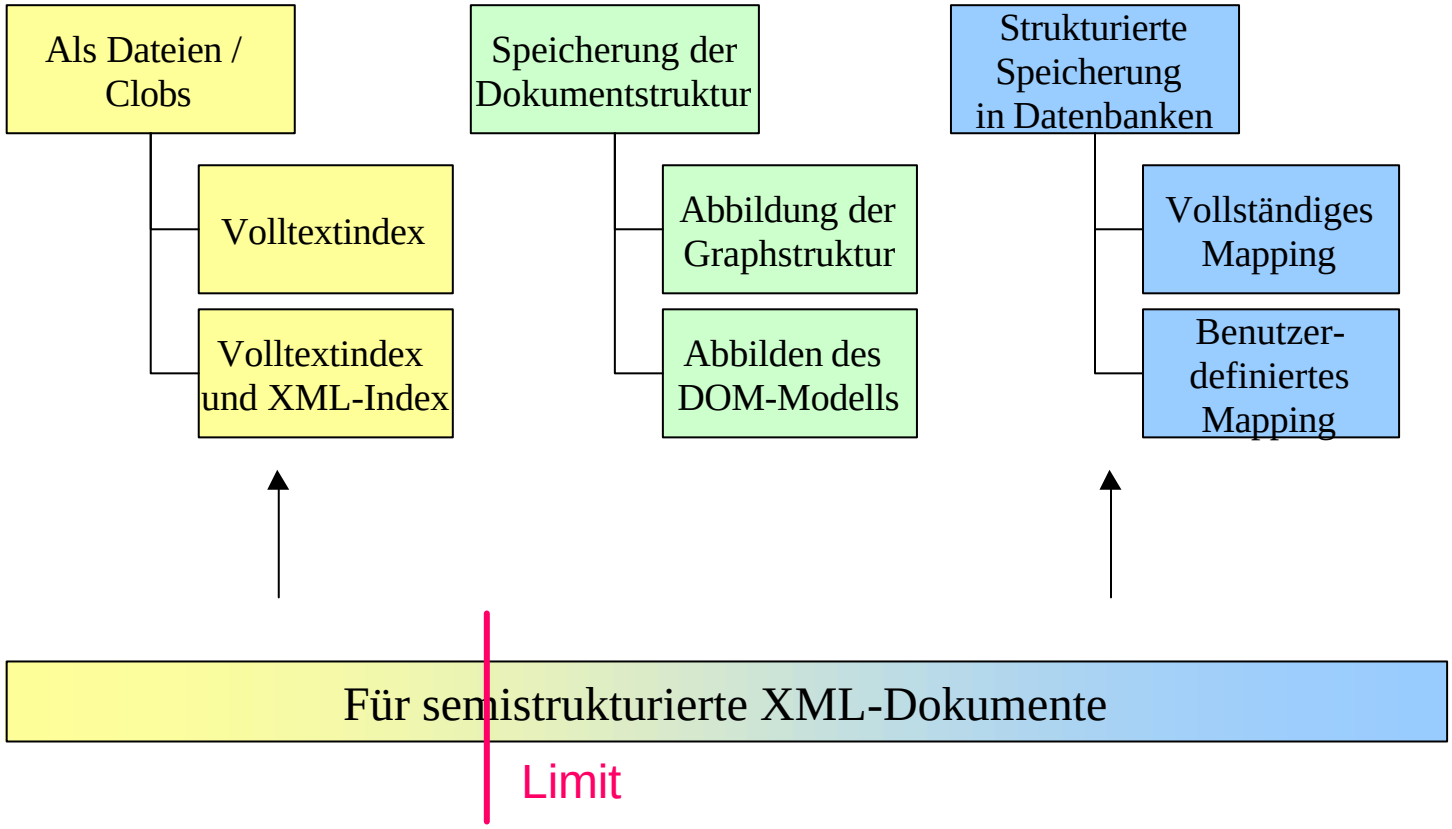
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Integration der hybriden Speicherung in die Klassifikation



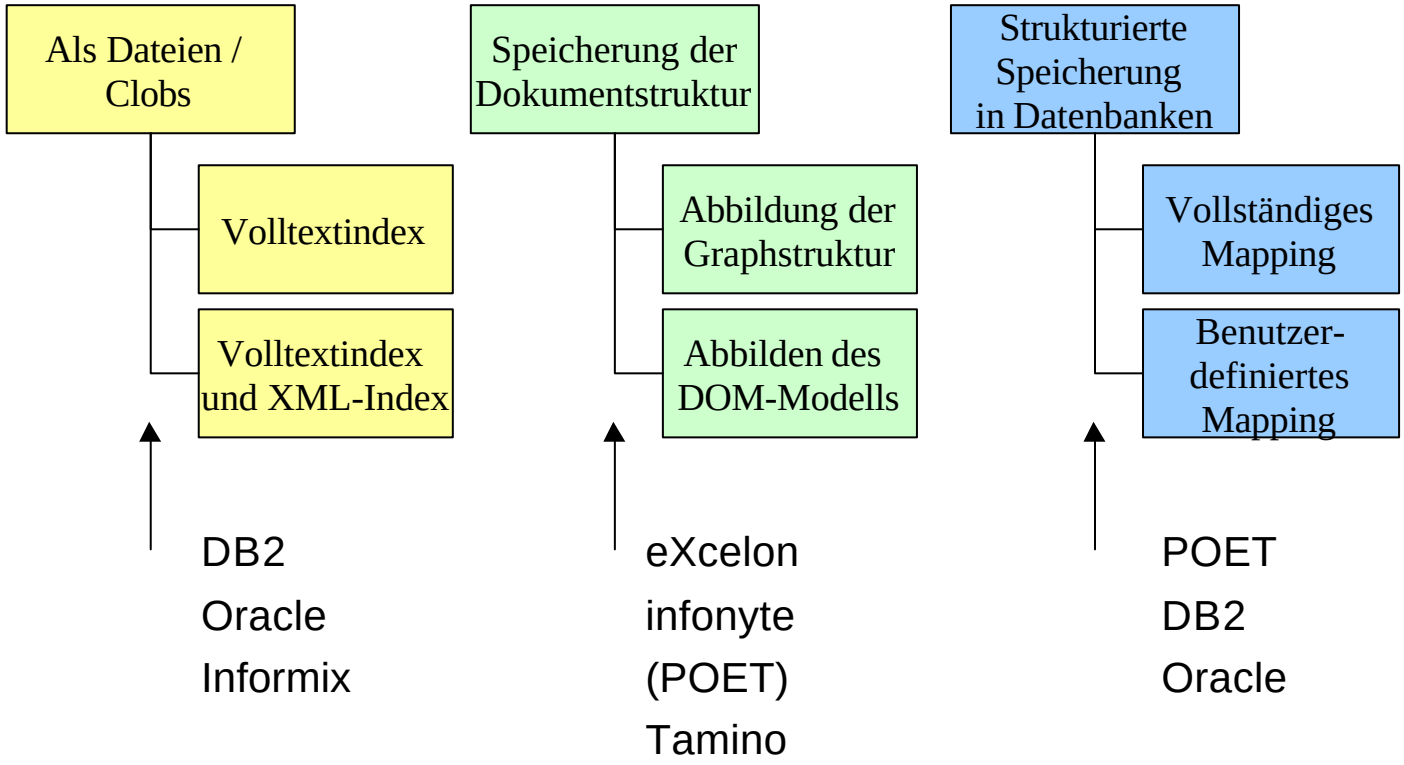
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Einsatz der Methoden



Zusammenfassung /1

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Verschiedene Verfahren zur Speicherung von XML-Dokumenten existieren:

- Speicherung der XML-Dokumente als Ganzes und Indizierung (textbasiert native Verfahren)
- Speicherung der Graphenstruktur (modellbasierte native Verfahren)
- strukturierte Abbildung auf Datenbanken
- Hybride Verfahren

Zusammenfassung /2

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Speicherung von XML-Dokumenten

- Verfahren oft aus den Bereichen Datenbanken und Dokumentverarbeitung
- generell: es gibt keine „beste Lösung“ für alle XML-Dokumente
- geeignetste Lösung ist abhängig von konkreten Erfordernissen

- vieles ist noch in der Entwicklung

Literatur /1

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Neil Bradley, *The XML companion*, Addison Wesley, 1998
- Yoshikawa, M., Amagasa, T., Shimura, T., Uemura, S., XRel: A Path-Based Approach for Storage and Retrieval of XML Documents Using Relational Databases: ACM Transactions on Internet Technology, Volume 1, Nummer 1, S. 110-141, 2001
- Daniela Florescu, Donald Kossmann, *Storing and Querying XML Data Using an RDBMS*, Bulletin of the Technical Committee on Data Engineering, volume 22, number 3, September 1999, pp. 27-34
- Jayaval Shanmugasundaram, Kristin Tufte, Gang He, Chun Zhang, David DeWitt, Jeffrey Naughton, *Relational Databases for Querying XML Documents - Limitations and Opportunities*, Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999, pp. 302-314
- Meike Klettke, Holger Meyer, *XML and Object-Relational Databases - Enhancing Structural Mappings Based on Statistics*, WebDB 2000, Mai 2000

Literatur /2

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

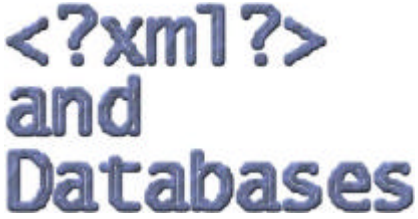
Indizierung
Systeme

– Ronald Bourret, C. Bornhövd, A.P. Buchmann, *A Generic Load/ Extract Utility for Data Transfer between XML Documents and Relational Databases*, Second International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems, WECWIS'00
– Ronald Bourret, *XML-DBMS - Middleware for Transferring Data between XML Documents and Relational Databases*, http://www.rpbourret.com/xmldbms/index.htm
– Alin Deutsch, Mary F. Fernandez, Dan Suciu, *Storing Semistructured Data with STORED*, SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1999, Philadephia, Pennsylvania, USA, ACM Press, 1999, pp. 431-442

– Meike Klettke, Holger Meyer, *XML and databases*, http://www.xml-und-datenbanken.de

XML und Datenbanken
Generieren von XML-Dokumenten

Meike Klettke
Universität Rostock
Fachbereich Informatik
meike@informatik.uni-rostock.de



www.xml-und-datenbanken.de

Inhalt

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Generierung von XML-Dokumenten

- aus relationalen Datenbanken
  - Anforderungen
  - für vollständige Datenbanken/Relationen
  - für Ergebnisse von SQL-Anfragen
  - anwenderdefinierte Verfahren
- aus objektrelationalen Datenbanken
- aus beliebigen Informationen
- Implementierungen
- Weiterführende Literatur

Generierung von XML-Dokumenten aus relationalen Datenbanken

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- am häufigsten eingesetzte Datenbanken
- große Datenmengen sind so bereits elektronisch gespeichert

- Verwendung dieser bietet sich an, wenn man XML-Dokumente erzeugen will
- generierte XML-Dokumente sind meist sehr regulär, gleichmäßig strukturiert



datenzentriert

Anforderungen

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Vollständige Abbildung von Datenbankinhalten

  Alle in der Datenbank vorhandenen Informationen müssen sich im XML-Dokument wiederfinden.

- Abbildung von Anfrageergebnissen oder Views

  Ausschnitt der gespeicherten Informationen, der durch eine Anfrage oder eine Sicht spezifiziert wird, wird als XML-Dokument dargestellt.

- Einsatz individueller Transformationsregeln

  Ein Ausschnitt der Daten sowie deren Struktur wird festgelegt.

Allgemeine Problemstellungen

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Abbildung von:

Datenbank	XML-Dokument
Relationen- und Attributnamen	Element- und Attributnamen
Datentypen	Datentypen von XML Schema
Daten aus Datenbanken	Inhalte in XML-Dokumenten

SQL/XML - Standardisierungsinitiative verschiedener Hersteller von Datenbanksystemen

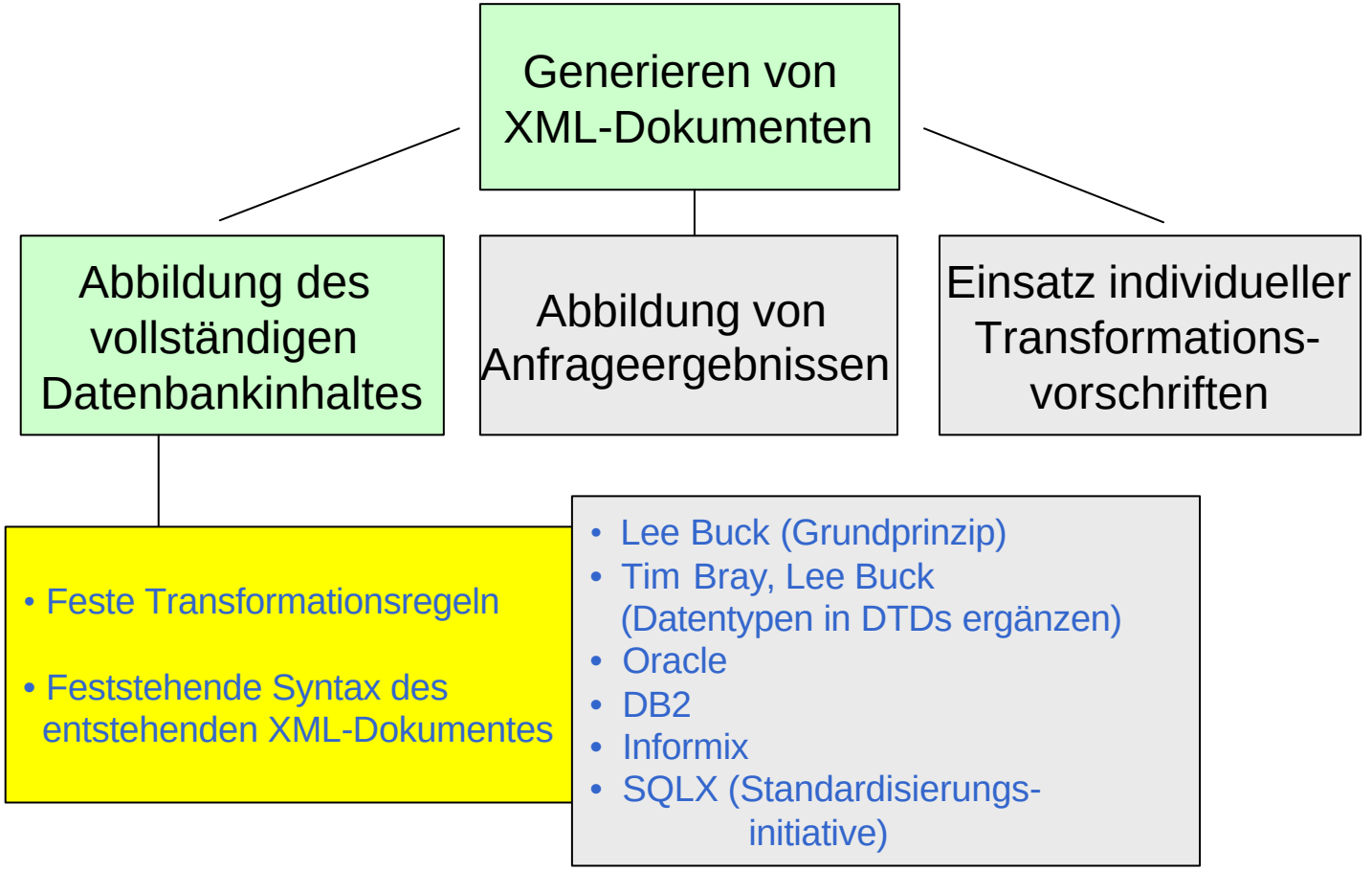
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Generieren von XML-Dokumenten aus Datenbanken



Vollständige Abbildung der Datenbankinhalte / 1

- Datenbank: betrachtet als dreistufige Hierarchie von
  Datenbank - Relation - Attributen
- Darstellung im XML-Dokument

Hotel:

HotelID	Name	Kategorie	Adresse
H0001	Hotel Huebner	4	A0001
H0002	Warnemuender Hof	3	A0002

Adresse:

AdresseID	Ort	PLZ	Strasse	Nr
A0001	Warne-muende	18119	Seestrasse	12
A0002	Warne-muende	18119	Stolteraer Weg	8

durch Elemente

```
<Hotelinformationen>
    <Hotel>
        <HotelID>H0001</HotelID>
        <Name>Hotel Huebner</Name>
        <Kategorie>4</Kategorie>
        <Hoteladr>A0001</Hoteladr>
    </Hotel>
    <Hotel>
            ...
    </Hotel>
    <Adresse>
            ...
    </Adresse>
</Hotelinformationen>
```

Vollständige Abbildung der Datenbankinhalte / 2

- Datenbank: betrachtet als dreistufige Hierarchie von
  Datenbank - Relation - Attributen
- Darstellung im XML-Dokument

Hotel:

HotelID	Name	Kategorie	Adresse
H0001	Hotel Huebner	4	A0001
H0002	Warnemuender Hof	3	A0002

Adresse:

AdresseID	Ort	PLZ	Strasse	Nr
A0001	Warne-muende	18119	Seestrasse	12
A0002	Warne-muende	18119	Stolteraer Weg	8

durch Elemente und Attribute

```
<Hotelinformationen>
    <Hotel
        HotelID='H0001'
        Name='Hotel Huebner'
        Kategorie='4'
        Hoteladr='A0001' />
    <Hotel  ... />
    <Adresse  ... />
</Hotelinformationen>
```

Sonderfälle: Schlüssel und Fremdschlüssel

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Verwendung eines XML-Schemas
  - Einsatz von key / keyref

- Verwendung einer DTD
  - Darstellung von Schlüsseln und Fremdschlüsseln
    - Abbildung auf Attribute (ID/IDREF)
    - Eindeutigmachen der Schlüssel- und Fremdschlüsselwerte

    (globale Eindeutigkeit von IDs in XML-Dokumenten vs. lokale Eindeutigkeit von Schlüsseln einer Relation)

Hierarchien aus Schlüssel- und Fremdschlüsselbeziehungen

- Schlüssel- und Fremdschlüsselbeziehungen werden verfolgt
- Hierarchien werden abgeleitet

Hotel:

HotelID	Name	Kategorie	Adresse
H0001	Hotel Huebner	4	A0001
H0002	Warnemuender Hof	3	A0002

Adresse:

AdresseID	Ort	PLZ	Strasse	Nr
A0001	Warne-muende	18119	Seestrasse	12
A0002	Warne-muende	18119	Stolteraer Weg	8

```
<!ELEMENT Hotel (HotelID, Name,
                 Kategorie?, Hoteladr)>
<!ELEMENT Hoteladr (AdresseID, Ort,
                    PLZ, Strasse, Nr)>
<!ELEMENT HotelID (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Kategorie (#PCDATA)>
<!ELEMENT AdresseID (#PCDATA)>
<!ELEMENT Nr (#PCDATA)>
 <!ELEMENT Ort (#PCDATA)>
<!ELEMENT PLZ (#PCDATA)>
<!ELEMENT Strasse (#PCDATA)>
```

Sonderfälle: Datentypen

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Verwendung von XML Schema: Vielzahl vordefinierter Datentypen

```
<xsd:element name=´ort´ type=´xsd:string´>
<xsd:element name=´plz´ type=´xsd:integer´>
```

- Erweiterung von DTDs um reservierte Attribute

```
<ort xml-sqltype=´varchar´>Rostock</ort>
<plz xml-sqltype=´integer´>18055</plz>
```

XML-Prozessoren oder Applikationen müssen diese Informationen kennen und auswerten, sonst nur protokollierende Funktion

Eigenschaften der vollständigen Abbildung des Datenbankinhaltes

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Ausgabe der Datenbank	vollständig
Erforderliche Informationen	keine
Variables Ausgabeformat	nein
Erhalt von Datentypen	durch erweiterte DTDs oder ein XML-Schema
Speicherung von Schlüsseln und Fremdschlüsseln	als ID/IDREF oder Abbilden auf Hierarchien im XML-Dokument
Weitere Besonderheiten	Parser müssen DTD-Erweiterungen auswerten XML Schema ist wesentlich besser geeignet

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Generieren von XML-Dokumenten aus Datenbanken

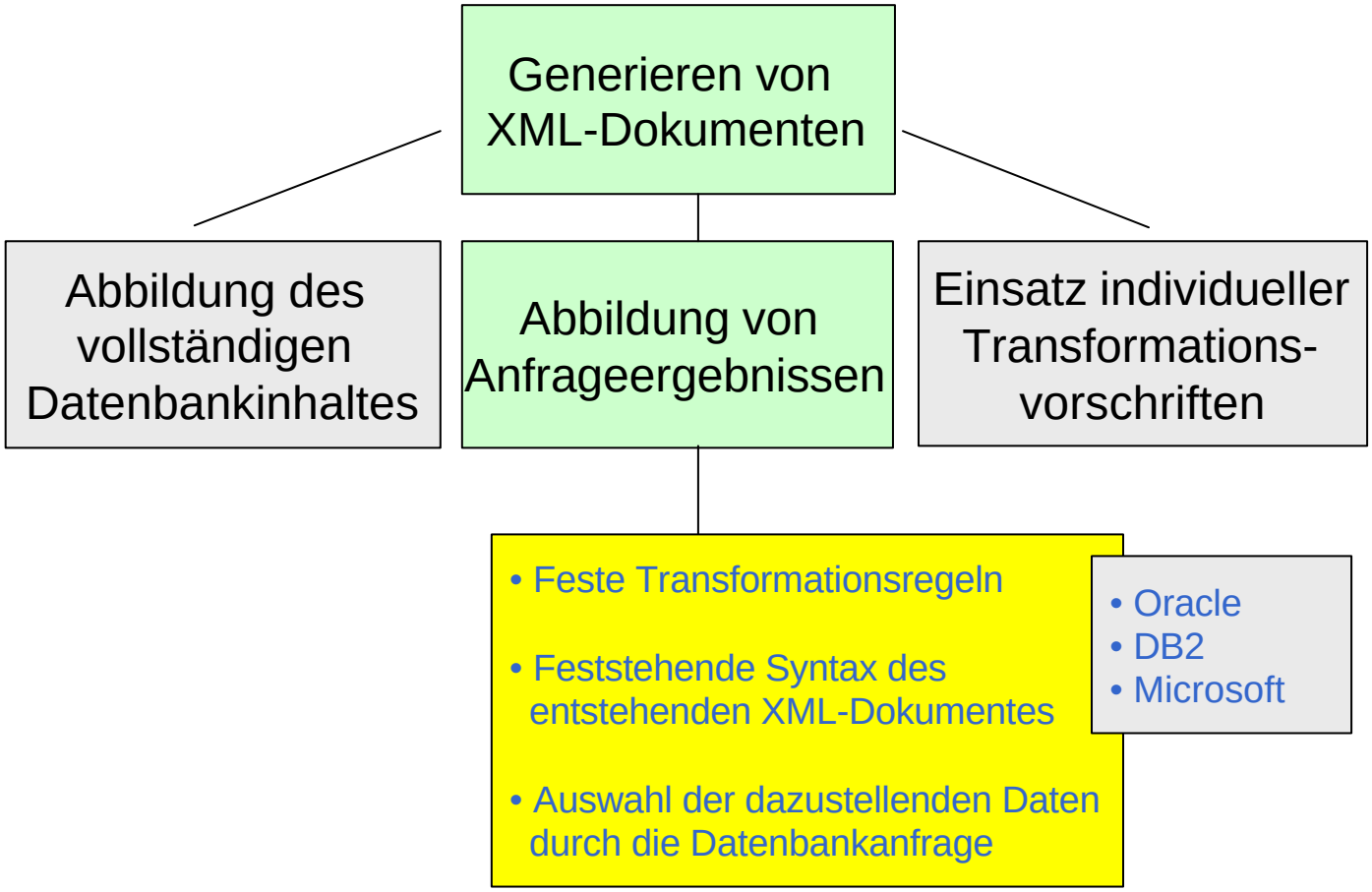


Abbildung von Anfrageergebnissen

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Ergebnisse von SQL-Anfragen mit XML-Syntax ausgeben

Beispiel:

```
SELECT Name, Kategorie, Ort
FROM Hotel, Adresse
WHERE (Ort='Warnemuende') AND
  (Hotel.Hoteladr=Adresse.AdresseID)
```

Ergebnis:

```
<result>
      <Name>Hotel Huebner</Name>
      <Kategorie>4</Kategorie>
      <Ort>Warnemuende</Ort>
</result>
```

Eigenschaften des Einsatzes von Datenbankanfragen

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Ausgabe der Datenbank	Ausschnitte
Erforderliche Informationen	Datenbankenanfrage/ View
Variables Ausgabeformat	nein
Erhalt von Datentypen	beim Generieren von XML Schema
Speicherung von Schlüsseln und Fremdschlüsseln	Bei Verwendung von XML Schema
Weitere Besonderheiten	keine

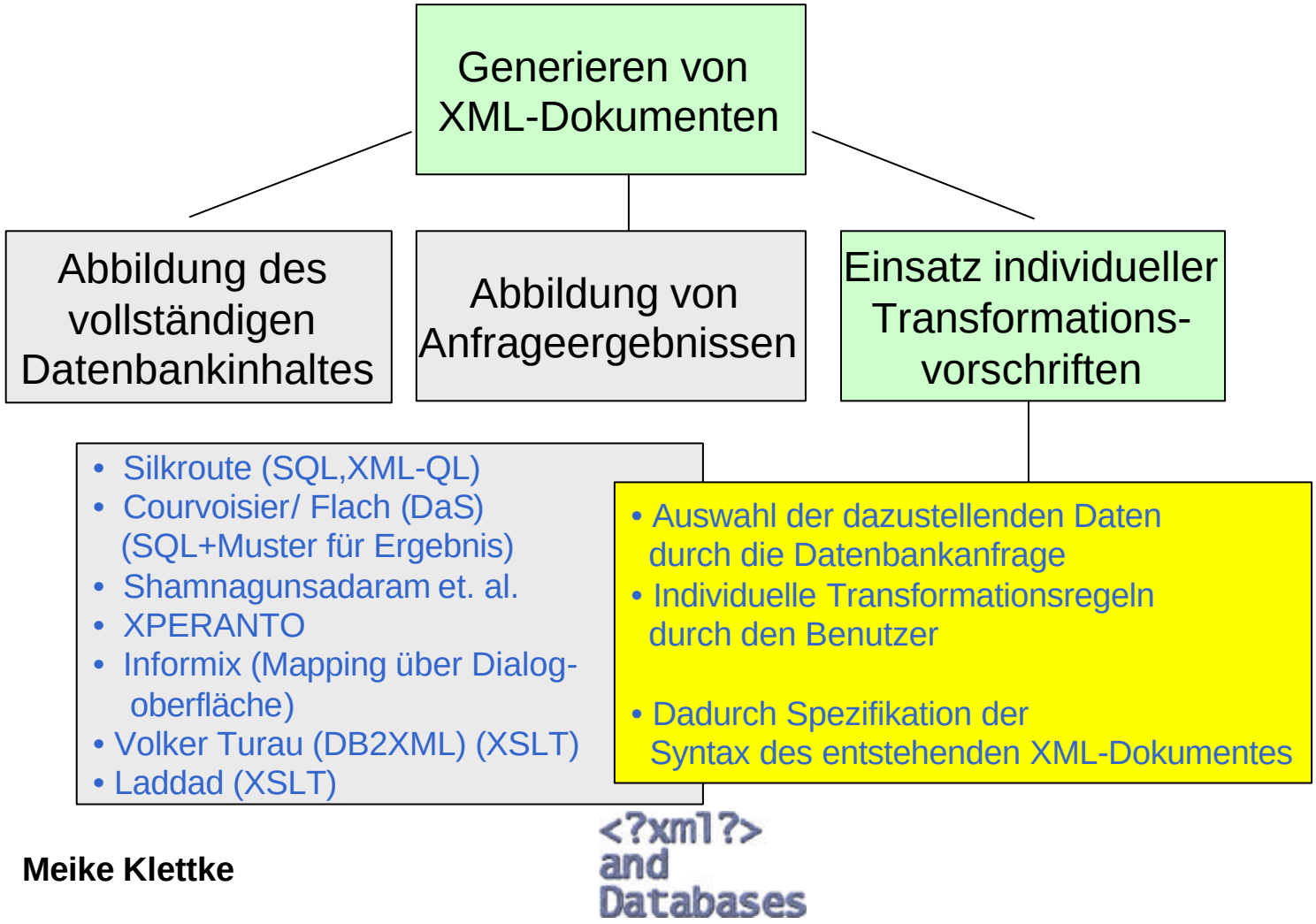
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Generieren von XML-Dokumenten aus Datenbanken



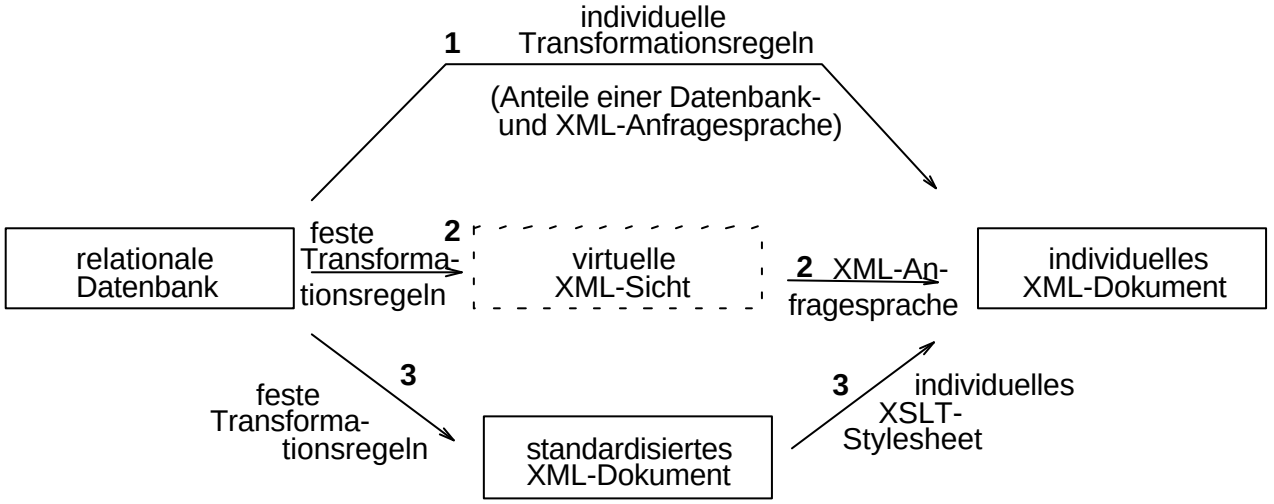
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Varianten zum Einsatz individueller Transformationsregeln



1
- XML-Constructor
- Database to XML Servlet

2
- Silkroute
- XPERANTO
- MS-SQL-Server

3
- XML-APIs for DB (Laddad)
- DB2XML

Einsatz von erweiterten Datenbank-Anfragesprachen

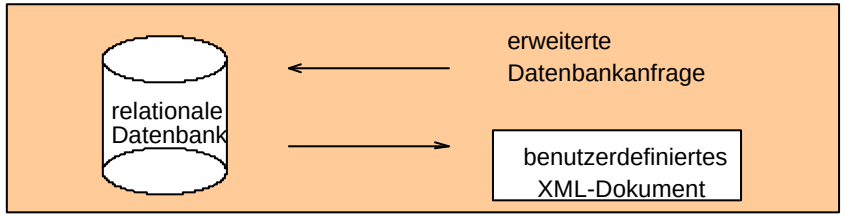
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Anteile von Datenbank- und XML-Anfragesprachen



- Datenbankanfrage zur Auswahl der dazustellenden Daten (= relevanten Anteile)
- XML-Anfrage zur Bestimmung der Syntax des Zielformates (CONSTRUCT)

Einsatz von erweiterten Datenbankanfragen

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Anfragebeispiel:

```
select *
from hotel, adresse
construct
<hotel>
  <name>{name}</name>
  <adresse>
    <plz>{plz}<plz>
    <wohnort>{ort}</wohnort>
    <strasse>{strasse}</strasse>
    <hausnummer>{nr}</hausnummer>
  </adresse>
</hotel>
```

Ergebnis:

```
<hotel>
  <name>Strand Hotel
   Huebner</name>
  <adresse>
    <plz>18119<plz>
    <wohnort>Warnemuende</wohnort>
    <strasse>Seestrasse</strasse>
    <hausnummer>12</hausnummer>
  </adresse>
</hotel>
```

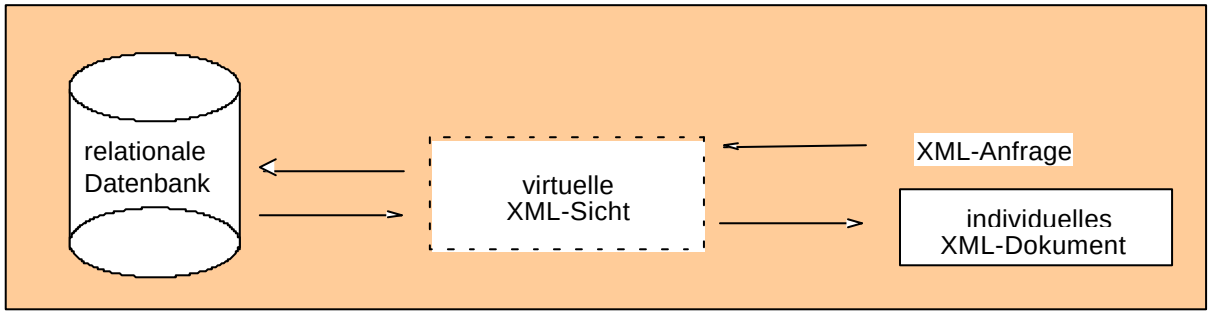
Verwendung von Transformations-regeln und XML-Anfragen

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme



- virtuelle „Sichtweise“ auf die Daten einer Datenbank in Form eines XML-Dokuments
- An dieses virtuelle XML-Dokument werden XML-Anfragen gestellt.

Beispiel Silkroute

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Erzeugen der virtuellen „XML-Sicht“

```
construct <view> {
   from adresse a, hotel h
     construct <hotel>
                  <name>$h.name</name>
                  <adresse>
                    <ort>$a.ort</ort>
                    <plz>$a.plz</plz>
                    <strasse>$a.strasse</strasse>
                     <nummer>$a.nr</nummer>
                  </adresse>
               </hotel>
</view>
```

- XML-Anfragen über dieser XML-Sicht (XML-QL, XQuery)

Einsatz von Transformationsregeln und XSLT

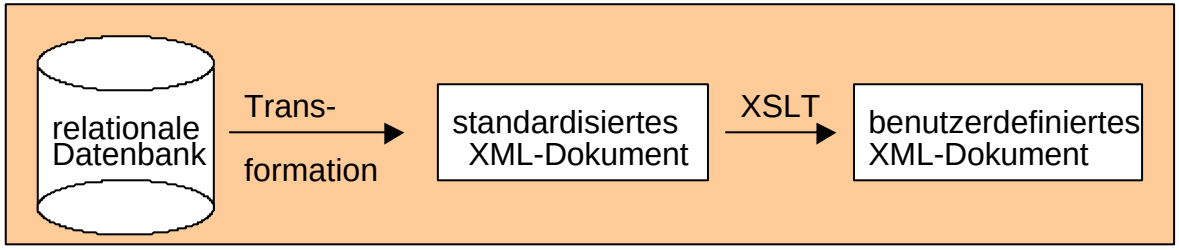
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Zweistufiger Prozess (Turau, Laddad)



1. Schritt: Ausgabe der vollständigen Datenbank in XML-Syntax im festen Format
2. Schritt: Einsatz von XSLT zur Auswahl der relevanten Teilinformationen und Bestimmung der Syntax des Zielformates

Eigenschaften des Einsatzes individueller Transformationen

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Ausgabe der Datenbank	vollständig oder Ausschnitte
Erforderliche Informationen	Datenbanken sowie XML-Anfrage oder XSLT
Variables Ausgabeformat	ja
Erhalt von Datentypen	bei Verwendung von XML Schema
Speicherung von Schlüsseln und Fremdschlüsseln	Bei Verwendung von XML Schema
Weitere Besonderheiten	keine

Verwendung von Mappingvorschriften

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Mappingvorschrift (oder Abbildungsvorschrift)

- legt die Abbildung von Datenbank-Inhalten auf XML-Dokumente fest und
- definiert die Verbindung zwischen der Datenbank und dem XML-Dokument.

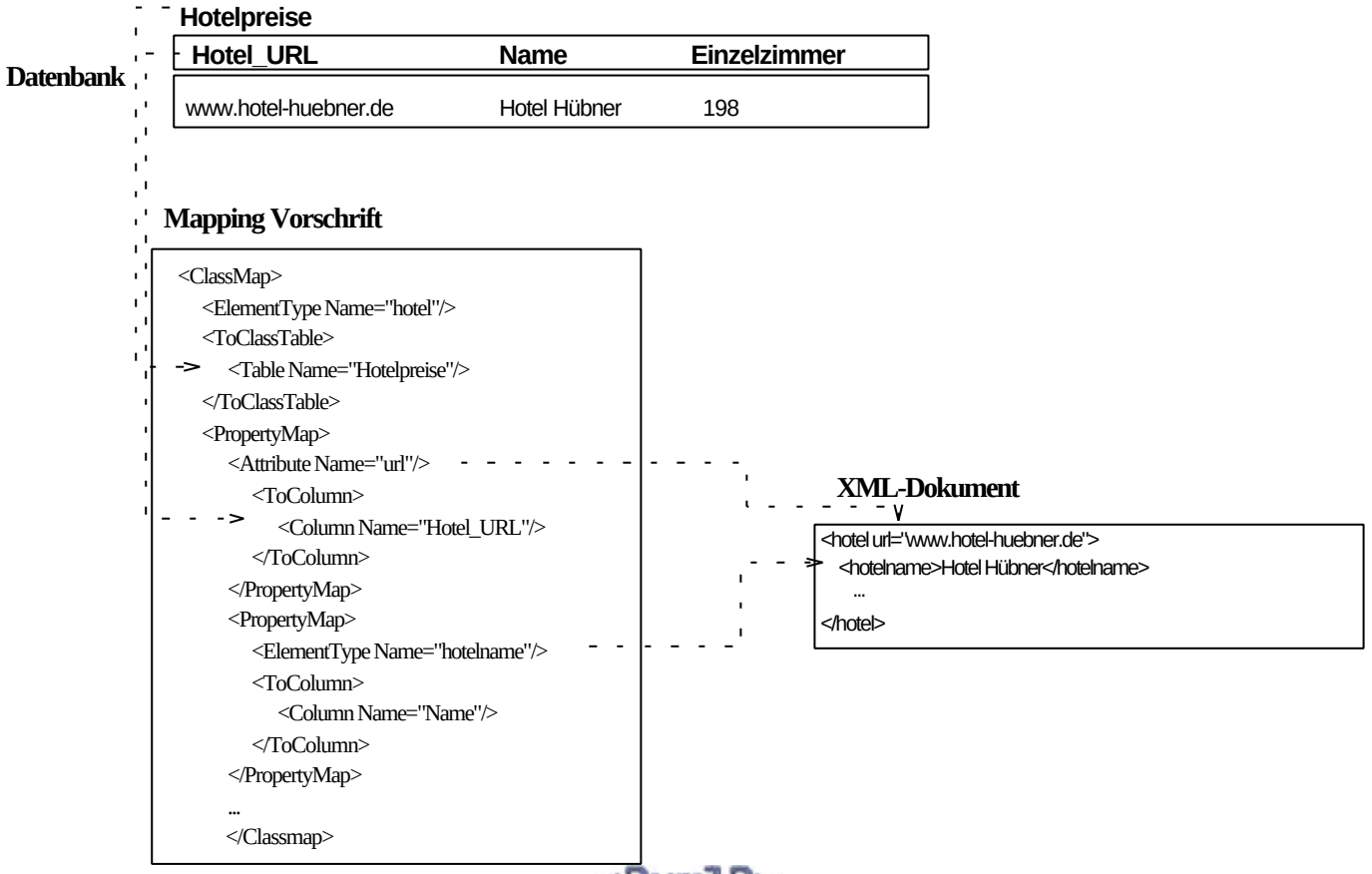
Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Verwendung von Mappingvorschriften



Eigenschaften des Einsatzes von Mappingvorschriften

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Ausgabe der Datenbank	vollständig oder Ausschnitte
Erforderliche Informationen	Mappingvorschrift
Variables Ausgabeformat	ja
Erhalt von Datentypen	bei Verwendung von XML Schema
Speicherung von Schlüsseln und Fremdschlüsseln	Bei Verwendung von XML Schema
Weitere Besonderheiten	keine

Verwendung von objekt-relationalen Datenbanken /1

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- adäquate Abbildung von komplexen Attributen (als Tupel, Mengen oder Listen definierten)
- Beispiel:

```
create row type Adresse_t
(
  PLZ      INTEGER,
  Ort      VARCHAR(25),
  Strasse  VARCHAR(20),
  Nr       INTEGER
);
create table Hotel
(
  HotelID  INTEGER  NOT NULL
           PRIMARY KEY,
  Name     VARCHAR(20) NOT NULL,
  Adresse  Adresse_t,
  Telefon  SET(INTEGER  NOT NULL)
);
```

```
<Hotelinformationen>
    <Hotel>
        <HotelID>H0001</HotelID>
        <Name>Hotel Hübner</Name>
        <Adresse>
            <PLZ>18119</PLZ>
            <Ort>Warnemünde</Ort>
            <Strasse>Seestrasse</Strasse>
            <Nr>12</Nr>
        </Adresse>
        <Telefon>0381/5434-0</Telefon>
        <Telefon>0381/5434-44</Telefon>
    </Hotel>
</Hotelinformationen>
```

Verwendung von objekt-relationalen Datenbanken /2

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- adäquate Abbildung von komplexen Attributen (als Tupel, Mengen oder Listen definierten)
- Beispiel:

```
create row type Adresse_t
(
  PLZ      INTEGER,
  Ort      VARCHAR(25),
  Strasse  VARCHAR(20),
  Nr       INTEGER
);
create table Hotel
(
  HotelID  INTEGER  NOT NULL
           PRIMARY KEY,
  Name     VARCHAR(20) NOT NULL,
  Adresse  Adresse_t,
  Telefon  SET(INTEGER  NOT NULL)
);
```

```
<!ELEMENT Hotelinformationen (Hotel*)>
<!ELEMENT Hotel (HotelID, Name,
                 Adresse, Telefon+)>
<!ELEMENT HotelID (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Adresse (PLZ, Ort,
                   Strasse, Nr)>
<!ELEMENT PLZ (#PCDATA)>
<!ELEMENT Ort (#PCDATA)>
<!ELEMENT Strasse (#PCDATA)>
<!ELEMENT Nr (#PCDATA)>
<!ELEMENT Telefon (#PCDATA)>
```

Eigenschaften der Ausgabe objektrelationaler Datenbanken

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Ausgabe der Datenbank	*v*ollständig
Erforderliche Informationen	keine
Variables Ausgabeformat	nein
Erhalt von Datentypen	bei Verwendung von XML Schema
Speicherung von Schlüsseln und Fremdschlüsseln	als ID/IDREF oder Verwendung von XML Schema
Weitere Besonderheiten	adäquate Darstellung von Tupeln, Mengen und Listen der Datenbank im XML-Dokument

Generierung von XML-Dokumenten aus weiteren Datenformaten /1

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

allgemeine Prinzipien:

- aussagekräftige Bezeichnungen für Element- und Attributnamen
- sinnvolle Gruppierungen und hierarchische Darstellungen
- Darstellung aller benötigten Informationen
- keine Darstellung von überflüssigen Informationen
- Lesbarkeit (für Anwendungen sowie für Menschen)
- Strukturierung der Informationen durch feingranulares Markup

Generierung von XML-Dokumenten aus weiteren Datenformaten /2

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

Alle Merkmale:

- nicht formal definierbar
- nicht formal überprüfbar

Qualität der erzeugten Dokumente überprüfen:

- Arbeit mit Beispieldokumenten

  (alle benötigten Informationen dargestellt, Zusammenhänge schlüssig)

- Betrachten von typischen Anfragen

  (lassen sich Informationen erfragen und vollständig ableiten)

Metriken bewerten Benutzerbarkeit und Änderbarkeit entstandener XML-Dokumente.

Zusammenfassung

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- verschiedene Methoden existieren, um XML-Dokumente aus anderweitig elektronisch gespeicherten Informationen zu generieren
- Schwerpunkt dabei relationale und objektrelationale Datenbanken

- viele Methoden werden in kommerziellen Datenbanksystemen umgesetzt

Literatur / 1

Einleitung, Anwendungen
XML Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Lee Buck, *XML representation of a relational database,* http://www.w3.org/XML/RDB.html
- Tim Bray, *Adding Strong Data Typing to SGML and XML,* http://www.textuality.com/xml/typing.html

- Jim Melton, *WG3 Tutorial Presentation on 2000-10-09, ISO/IEC JTC/SC 32 - Data Management and Interchange*, United States of America (ANSI), 2000
- Jim Melton, *ISO-ANSI Working Draft XML-Related Specifications (SQL/XML),* http://www.sqlx.org/, 2001

- Jayavel Shanmugasundaram, Eugene J. Shekita, Rimon Barr, Michael J. Carey, Bruce G. Lindsay, Hamid Pirahesh, Berthold Reinwald, *Efficiently Publishing Relational Data as XML Documents*, VLDB, 2000
- Thomas Courvoisier, Guntram Flach, *Integration relationaler Datenstrukturen in XML-Applikationen - database to XML Servlet,* 30. GI Jahrestagung, Workshop Internet-Datenbanken, 2000

Literatur / 2

Einleitung, Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Mary Fernandez, Wang-Chiew Tan, Dan Suciu, *SilkRoute: Trading between Relations and XML* , WWW9, http://db.cis.upenn.edu/DL/rxl/rxl.html, 2000
- Mary Fernandez, Yana Kadiyska, Atsuyuki Morishima, Dan Suciu, Wang-Chiew Tan, *SilkRoute a framework for publishing relational data in XML*, Unpublished, Manuscript availabe from www.cs.washington.edu/homes/suciu, 2002
- Micheal Carey, Daniela Florescu, Zachary Ives, Ying Lu,Jayavel Shanmugasundaram, Eugene Shekita, Subbu Subramanian,*XPERANTO: Publishing Object-Relational Data as XML,* WebDB Informal Proceedings, May, 2000
- Volker Turau, *Making legacy data accessible for XML applications*, http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html,2000
- Volker Turau, *Generieren von XML-Dokumenten aus relationalen Datenbanken*, 4. Darmstädter XML Kongress, XML und Datenbanken, Dezember, 2000
- Christian Ettinger, *Ein Java-Werkzeug zur Transformation von relationalen Daten-banken nach XML-Schema*, FH Wiesbaden, Fachbereich Informatik, 2000
- Ramnivas Laddad, *XML APIs for databases*, http://www.javaworld.com/javaworld/jw-01-2000/jw-01-dbxml.html, 2000

Literatur /3

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Ronald Bourret, *XML-DBMS - Middleware for Transferring Data between XML Documents and Relational Databases,* http://www.rpbourret.com/xmldbms/index.htm
- Gregor Zimmermann, *Repräsentation von Objekt-relationalen Daten in XML*, Studienarbeit, Universität Rostock, Fachbereich Informatik, 2002
- Sascha Klopp, *Automatische Generierung von virtuellen XML-Sichten aus relationalen Datenbankschemata und Übersetzung von XQuery-Anfragen nach SQL,* GI-Arbeitskreis Grundlagen von Informationssystemen, 14. Workshop über Grundlagen von Datenbanken, 2002

Kommerzielle Systeme

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- ANSI-Standard **SQL/XML**
- Hersteller wie IBM, Oracle, Microsoft, Informix und Software AG beteiligt

- Ausgabe von Relationen als XML-Dokument
- Ausgabe von Anfrageergebnissen und Sichten als XML-Dokument
- Abbildung von Datenbanktypen auf Typen von XML-Schemata und umgekehrt
- Konvertierung der Daten aus Datenbanken auf Werte in XML-Dokumenten

XML Developer's Kit (Oracle XDK)

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Grundbausteine zum Lesen, Manipulieren, Transformieren und Anzeigen von XML-Dokumenten
- XML SQL Utility (XSU) unterstützt die Ausgabe von Datenbankinhalten mit XML-Syntax
  - vollständige Inhalte relationaler Datenbanken auf XML-Dokumente,
  - Abbildung der Ergebnisse von SQL-Anfragen
  - individuell angepasste XML-Dokumente: Einsatz von XSLT
- Anfrageergebnisse: XML-Text oder DOM-Graphen

Ab Version 9*i* auch XML-Schema für die XML-Ergebnisse der SQL-Anfragen (Darstellung der Typinformationen)

IBM DB2 UDB

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- XML-Extender
- Generieren von XML-Dokumenten aus DB2-Datenbanken
- Syntax der Dokumente wird durch DAD-Dateien (- Data Access Definition) festgelegt

- Forschungsprojekt: XPERANTO
- erzeugt XML-Dokumente für XML-QL-Anfragen an DB2-Datenbanken
- noch nicht in das Produkt integriert

Microsoft SQL-Server

Einleitung,
Anwendungen
XML
Architektur

XML-Anfragen

XML-Speicherung
XML-Generierung

Indizierung
Systeme

- Ausgabe der Ergebnisse von SQL-Anfragen als XML-Dokumente
- Erweiterung der SQL-Anfrage um die Spezifikation FOR XML { RAW, AUTO, EXPLICIT }
- drei verschiedene Ausgabeformate
  - verschiedene Formate mit feststehender Syntax
  - individuelle Festlegung der XML-Syntax durch speziell erweiterte Anfragen

XML und Datenbanken — Systeme

Holger Meyer Meike Klettke

Database Research Group
University of Rostock
E-Mail: {hme|meike}@informatik.uni-rostock.de

7. Oktober 2002

Inhalt

1. Überblick zur Datenbankunterstützung für XML
2. DB2 XML und TextExtender
3. Oracle9i
4. MS SQL-Server
5. eXcelon
6. Tamino
7. Infonyte
8. POET, . . .

1. XML — Systeme

- IBM DB2 UDB, Text-Extender, XML-Extender
- ähnliches bei anderen RDBMS-Herstellern: Oracle9i, MS SQL-Server, Informix DS
- Software AG Tamino, erweitertes XQL, diverse Speicherungsmodelle
- ObjectDesign Inc $\rightarrow$ eXcelon, XQL, ObjectStore als Basis
- Infonyt-DB
- POET
- zahlreiche weitere XML-Server, DBMS-Erweiterungen, . . .

2. DB2 XML Extender

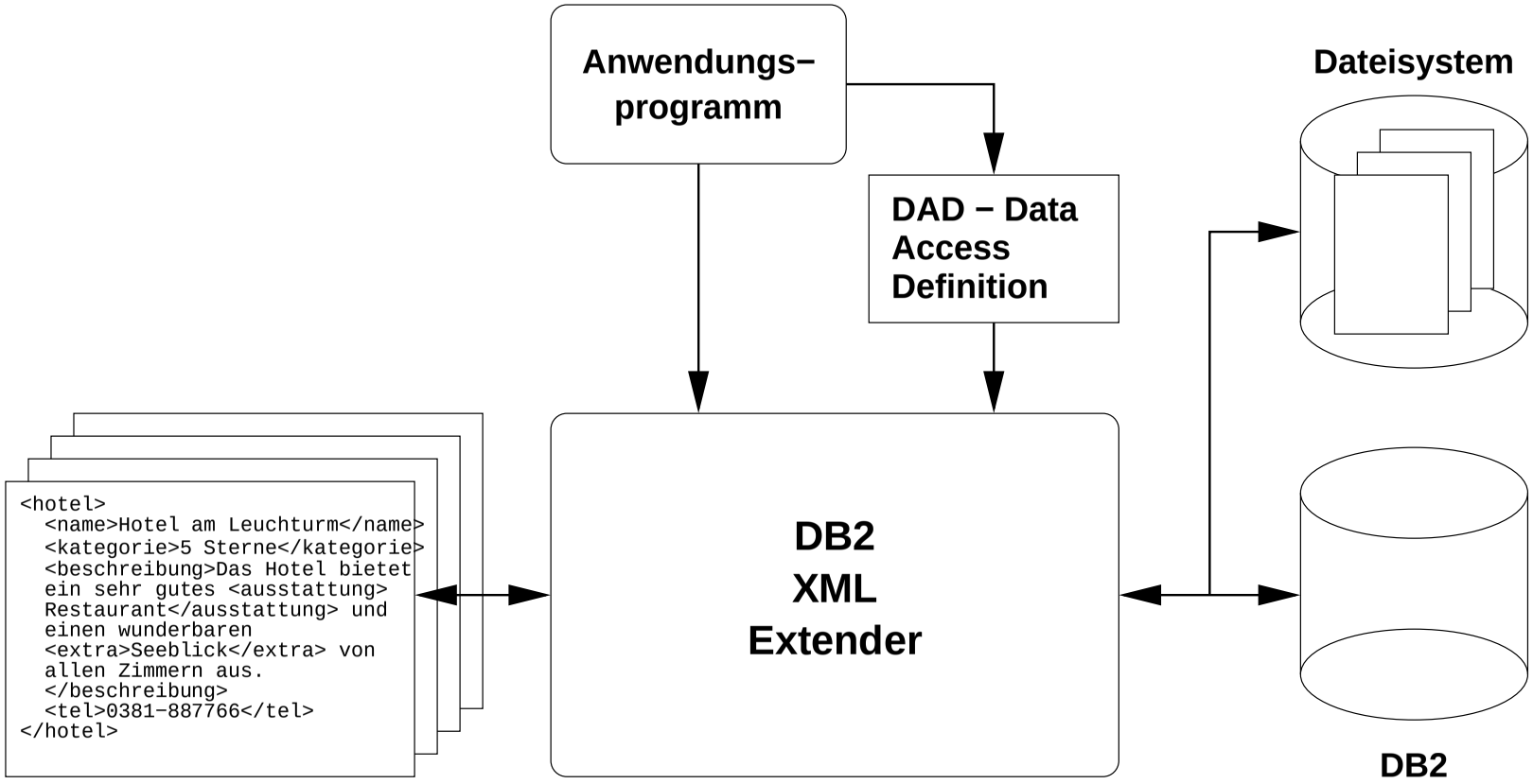
- vereinigt Objekt-relationale Datenbanktechnologie und XML
- Bestandteil von DB2 Universal Database (ab IBM DB2 UDB V7.1)
- Verfügbar für IBM AIX, Windows NT, Linux, SUN Solaris, auch für AS/400 und OS/390
- aktuelles unter:

  `http://www.software.ibm.com/data/db2/extenders/xmlext/`

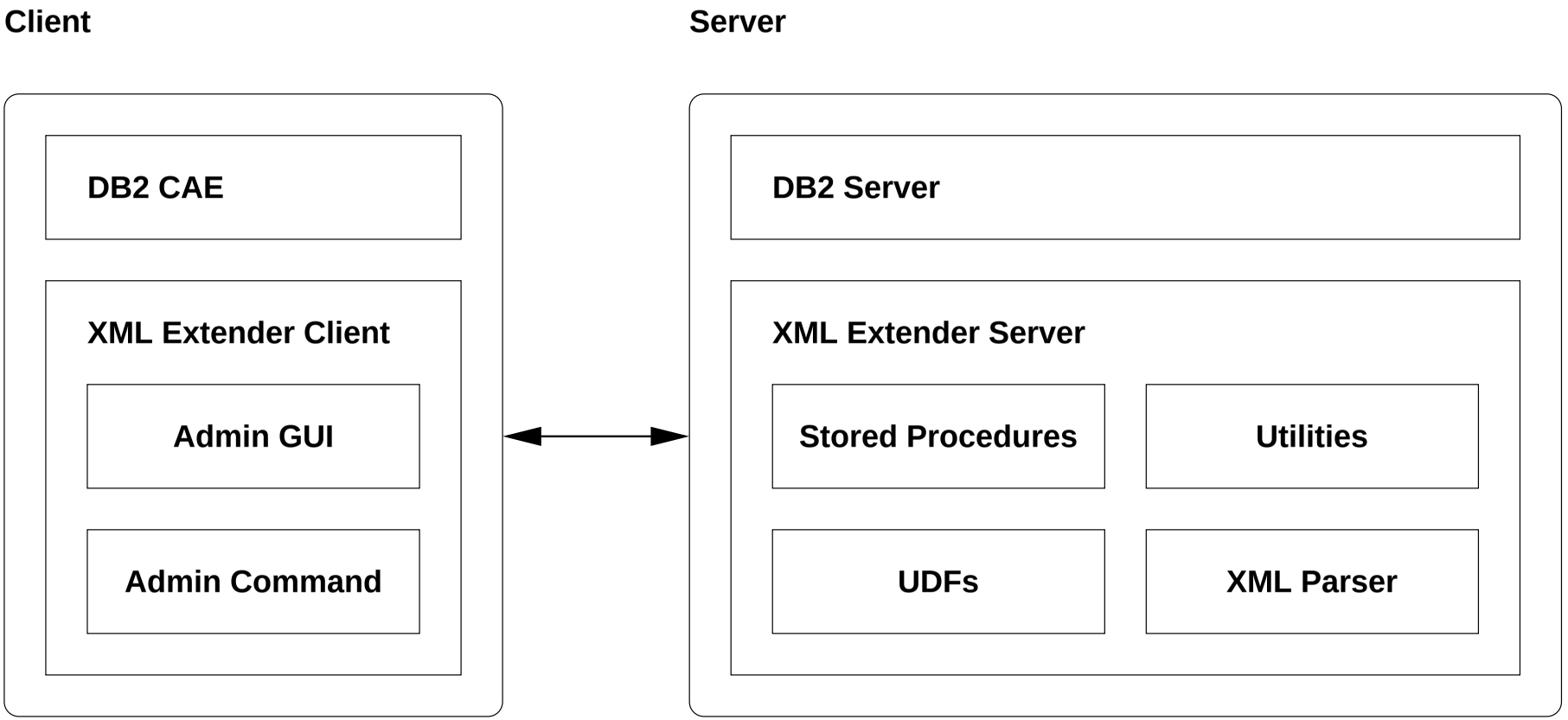
2.1. Überblick

- Import von XML, Abbildung auf relationale Strukturen
- Export von XML, Konstruktion von XML-Dokumenten aus Tabellen
- zwei Speicherungstechniken:
  - *XML Collection*: Abbildung auf relationale Strukturen
  - *XML Column*: XML-Datentyp, Speicherung von ganzen XML-Dokumenten oder -Fragmenten, Indizierung über Zusatztabellen (*Side tables*)
- Beschreibung der Speicherung mit DAD (*Data Access Definition*)
- Nutzung der Anfragemöglichkeiten von SQL, XPath-Ausdrücken und Information Retrieval des DB2 *TextExtenders*

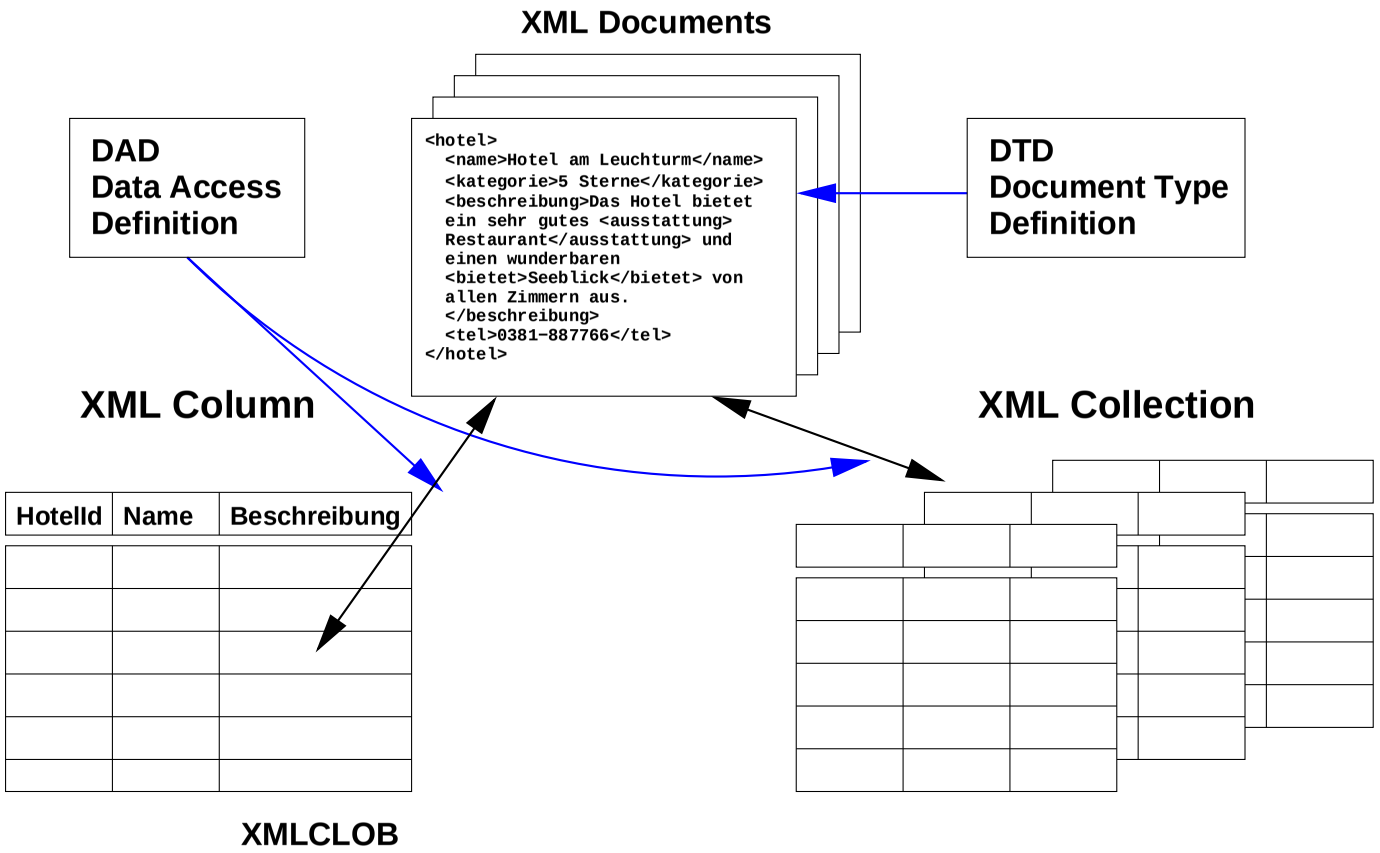
2.2. Überblick XML-Extender



2.3. XML-Extender Komponenten



2.4. XML-Speicherungtechniken



2.5. DAD — Beschreibung der XML-Speicherung

- selbst in Form eines XML-Dokumentes
- definiert Abbildung zwischen XML-Dokumenten und Datenbankrelationen
- für beide Speicherungsformen (XML Column und Collection)
- verwendet *XPath*-Standard,
- verkürzte Pfadausdrücke werden zur Beschreibung von XML-Elementen und Attributen genutzt
- Nutzung von XPath-Datenmodell (Node sets) für XML Collection-Abbildung
- zwei Abbildungsmethoden: *SQL*, *RDB_node*

SQL zur Komposition von XML-Dokumenten

- SQL-Syntax
- Spezifikation SQL **`select`**-Anweisung
- Spezifikation der Abbildung von Datenbankattributen (Tabellenspalten) auf XML-Daten als Attributwerte (*attribut_node*) und Elementinhalt (*text_node*)
- eine Reihe von Einschränkungen (Schachtelung, . . .)

RDB_node zur Komposition und Dekomposition

- XML-Syntax
- Spezifikation von Tabellen und Beziehung zwischen Tabellen im *RDB_node* des Wurzelknotens (*element_node*)
- pro *RDB_node* Angabe von Tabellennamen, Spaltennamen und wahlweise Bedingungen für *attribute_node* und *text_node*

2.6. Dynamische Erzeugung von DADs

Überschreiben der Anfragebedingungen zur Dokumenterzeugung

- *SQL_OVERRIDE*:
  - Ersetzen der **`select`**-Anweisung, bzw.
  - Modifikation der **`where`**-Klausel
- *XML_OVERRIDE*:
  - Spezifikation von Bedingungen an Elemente und Attribute,
  - XPath-Syntax Element- und Attributdefinitionen

```
/hotel/name = 'Hotel am Leuchturm' AND /hotel/@sterne = '5'
```

2.7. DAD-Beispiel

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "/usr/lpp/db2/dxx/dtd/dad.dtd">
<DAD><validation>NO</validation>
<Xcollection>
  <SQL_stmt>select band.name, album.title, album.label, album.year
            from band, album where band.name = album.band
            order by band.name, album.title, album.year</SQL_stmt>
  <prolog>?xml version="1.0"?</prolog>
  <doctype>!DOCTYPE band-info SYSTEM "band-info.dtd"</doctype>
  <root_node>
    <element_node name = "band">
      <attribute_node name = "name">
        <column_name = "band.name"/>
      </attribute_node>
      <element_node name = "album">
        ...
    </element_node>
  <root_node>
</Xcollection>
</DAD>
```

2.8. Speicherungs- und Zugriffstechniken

- *XML Column*, XML-Datentyp, UDTs + XPath, SQL, UDFs
- *XML Collection*, relationale Speicherung von XML + SQL, Stored procedures
- *DB2 TextExtender*, Volltextfunktionalität

2.9. *XML Column* — XML-Datentypen

- Speicherung als Datenbankattribut vom Typ *XMLCLOB*
- Abstrakter Datentyp: Struktur (UDT) und Verhalten (UDF)
- XPath-basierte Anfragen an Einzeldokumente mit UDFs
- XPath-Datenmodell (Node sets) dient zur Abbildung von XML-Strukturen auf Tabellenstrukturen
- Update von Elementen und Attributen mittels XPath-Ausdrücken
- Identifikation von mehrfach auftretenden Elementen über Sequenznummern (bildet Dokumentordnung ab)
- schnellere Suche durch Indizierung über *Side tables*

2.10. *XML Column* — UDTs

- *UDT*, DB2 User Defined Type, entspricht Struktur von ADT
- *XMLFile*, Referenz auf externe Datei, DB2 DataLink
- *XMLVarchar*, für interne, kurze Dokumente
- *XMLCLOB*, für interne, große Dokumente

2.11. *XML Column* — UDFs

- *UDF*, DB2 User Defined Function, entspricht Verhalten von ADT
- Funktionen/Methoden für: Import, Retrieval, Update, Extract
- *Extract*:
  - Extraktion von XML Element- und Attributwerten aus Dokument
  - Konvertierung von Zeichenfolgen aus XML-Dokumenten in Werte entsprechender SQL-Datentypen
  - skalare und Tabellen-UDFs für jeden Datentyp

2.12. *XML Column* — Extract UDF Beispiele

```
select db2xml.extractDouble(zimmertyp, '/zimmertyp/preis')
from zimmertyp
where hname = 'Hotel am Leuchturm'

select *
from table(db2xml.extractStrings(
  db2xml.XMLFile('/home/hme/hotel/hotels.xml'), '/hotel/*/name'))
  as hotelname
```

2.13. *XML Collection* — Abbildung auf Tabellen

- Dekomposition von XML-Dokumente in Tabellenstrukturen
- Speicherung der Elementinhalte und Attribute als Datenbankattribute
- direkte Abbildung von Relationen auf XML-Strukturen
- *Stored procedures* zur Verarbeitung
  - Zerlegung von XML-Dokumenten in neue oder existierende Tabellen
  - Generierung von XML-Dokumenten aus existierenden Daten oder zerlegten Dokumenten
  - liefern Typspezifikation und Werte für dynamische Abbildungsvorschriften (DADs)

2.14. Volltextsuche mit DB2 TextExtender

- orthogonal zu XML-Speicherung ist Volltextindizierung und Nutzung der Information Retrieval-Funktionalität des TextExtenders
- TextExtender: strukturierte Volltextsuche basierend auf Abschnitten
- Abschnitte durch Pfadausdrücke beschreibbar

```
select beschreibung
from hotel
where contains(beschreibungHandle,
    'MODEL order SECTION(/beschreibung/feature) "Seeblick"') = 1
```

- Nutzung des TextExtender ist unabhängig vom XML Extender und nicht zwingend

2.15. Weiteres

- DTD Repository in DB2, Referenzierung über *DTDID* in DAD
- Validierung gegen DTD beim Import von XML-Dokumenten durch Angabe der *DTDID* möglich
- Angabe allgemeiner Integritätsbedingungen für einzulesende oder erzeugte XML-Dokumente über *DAD* möglich
- GUI-basiertes Administrationswerkzeug
- Admin tool hilft bei Einrichten der XML-Unterstützung für Datenbank, Tabellen, Spalten und Kollektionen
- Unterstützung beim Erzeugen von *DAD*
- für Web-Anwendungen Integration mit DB2 Net.Data
- Programmierung mit *Embedded SQL*, *ODBC*, *JDBC* und *SQLJ*

Eigenschaften der XML-Speicherung in DB2 UDB

XML-Speicherung	
Modell	objektrelationales Modell
Schemabeschreibung	Validierung bei Import (DTD, Schema)
Art der Speicherung	sowohl vollständig als auch fragmentierend
Ordnungserhaltung	nur XML-Datentyp (XML Column)
XML-Datentyp	$\surd$
Indizierung	
Pfade	nur TextExtender
Werte	$\surd$
Volltext	nur TextExtender
XML-Import, XML-Export	
Unterstützung von Im- und Export, Abbildung spezifiziert durch DAD-Dateien	

Eigenschaften der XML-Speicherung in DB2 UDB (cont'd)

Anfrage- und Manipulationsschnittstellen	
Anfragen	SQL und XPath 1.0
Volltextfunktionalität	$\surd$(Information Retrieval mit TextExtender)
Änderungsoperationen	SQL, spezielle Methoden
Transformationen	–
Werkzeuge, Besonderheiten	
XML-sensitiver Volltextindex	
Speicherung basiert auf dem objektrelationalen Modell	

3. Oracle 9i

XML SQL Utility (XSU)

- die Ausgabe von Datenbankinhalten als XML-Dokumente (XML-Export),
- die native Speicherung von XML als **`XMLType`** und Abfragen mit XPath sowie
- die strukturierte Abbildung von Inhalten aus XML-Dokumenten in Relationen und Attributen.

Export von Datenbankinhalten mit XML-Syntax

- Spalten werden auf Elemente der ersten Ebene (top level) abgebildet.
- Einfache Typen (mit skalaren Werten) werden als Elemente mit PCDATA dargestellt.
- Strukturierte Typen und ihre Attribute werden auf Elemente mit Subelementen für die Attribute abgebildet. Damit werden komplexe Attribute als hierarchisch geschachtelte Elemente dargestellt.
- Aus Kollektionstypen werden Listen von Elementen.
- Objektreferenzen und referenzielle Integritätsbedingungen werden auf ID/IDREF innerhalb eines Dokuments abgebildet.

Beispielschema, objekrelational

```
CREATE TYPE Adresse_Type AS OBJECT (
     ort     VARCHAR2(30),
     plz     NUMBER,
     strasse VARCHAR2(40),
     nummer  CHAR(4)
);
CREATE TYPE preise_Type AS OBJECT (
    zimmertyp VARCHAR2(20),
    waehrung VARCHAR2(2),
    betrag REAL
);
CREATE TYPE preiseListType AS TABLE OF preise_Type;
CREATE TABLE hotel (
    hotel_name VARCHAR2(20),
    hotel_adr  Adresse_Type,
    preise preiseListType
) NESTED TABLE preise STORE AS nested_preise;
```

Anfrage SELECT * FROM hotel

```
<ROWSET>
  <ROW num="1">
    <hotel_name>Strand Hotel Huebner</hotel_name>
    <hotel_adr>
      <ort>Rostock-Warnemuende</ort>
      <plz>18119</plz>
      <strasse>Strandstr.</strasse>
      <nummer>12</nummer>
    </hotel_adr>
    <preise>
      <preise_ITEM num="1">
        <zimmertyp>einzelzimmer</zimmertyp>
        <waehrung>EUR</waehrung>
        <betrag>99</betrag>
      </preise_ITEM> ...
    </preise>
  </ROW> ...
  <ROW num="2"> ...
</ROWSET>
```

XML-Speicherung mit XMLType

```
create table hotels (
    name            VARCHAR2(40),
    ort             VARCHAR2(35),
    beschreibung    SYS.XMLTYPE,
    ausstattung     SYS.XMLTYPE,
    preise          SYS.XMLTYPE,
    telefonnr       VARCHAR(30)
);
```

XML-Speicherung mit XMLType (cont'd)

```
insert into hotels (name, ort, beschreibung, ausstattung,
  preise, telefonnr)
values ('Hotel Neptun', 'Warnemuende',
  sys.XMLType.createXML('In Warnemuende ankommen
    und sich zu Hause fuehlen'),
  sys.XMLType.createXML('<schwimmbad>Original-
    Thalassozentrum. Wir haben fuer Sie frisches
    Ostseewasser in das Hotel geholt!</schwimmbad>'),
  sys.XMLType.createXML('<DZ waehrung="Euro">186</DZ>
    <EZ waehrung="Euro">135</EZ>'),
  '0381-54370');
```

Selektion und Extraktion von XMLType-Instanzen

```
select ausstattung.extract
  ('schwimmbad/text()').getStringVal()
  "Swimmingpool"
from hotels;

Swimmingpool
-----------------
Original-Thalassozentrum. Wir haben fuer Sie
frisches Ostseewasser in das Hotel geholt!
```

Selektion und Extraktion von XMLType-Instanzen (cont'd)

```
select name, ausstattung.extract
  ('schwimmbad/text()').getStringVal() "Swimmingpool",
  preise.extract('/DZ/text()').getNumberVal()
  "Doppelzimmer"
from hotels
where ort = 'Warnemuende';
```

```
Name              Swimmingpool                Doppelzimmer
------------------------------------------------------------
Hotel             Original-Thalassozentrum    186
Neptun            Wir haben fuer Sie
                  frisches Ostseewasser
                  in das Hotel geholt!
```

Speicherung von XML-Dokumenten mit intermedia Text

- Volltextanfragen und XPath
- Kombination mit strukturierten SQL-Anfragen

```
select score(1) as score, beschreibung
from hotels
where contains(beschreibung, 'stem(Strand)', 1) > 0
order by score desc;
```

Eigenschaften der XML-Speicherung mit Oracle

XML-Speicherung	
Modell	objektrelationales Modell
Schemabeschreibung	nicht notwendig, Validierung möglich
Art der Speicherung	vollständige Speicherung als **`XMLType`**, fragmentierte Speicherung ist ebenfalls möglich
Ordnungserhaltung	bei **`XMLType`**
XML-Datentyp	**`XMLType`** mit Anfragemöglichkeit *XPath 1.0*
Indizierung	
Struktur, Pfade	–
Werte	√
Volltext	√

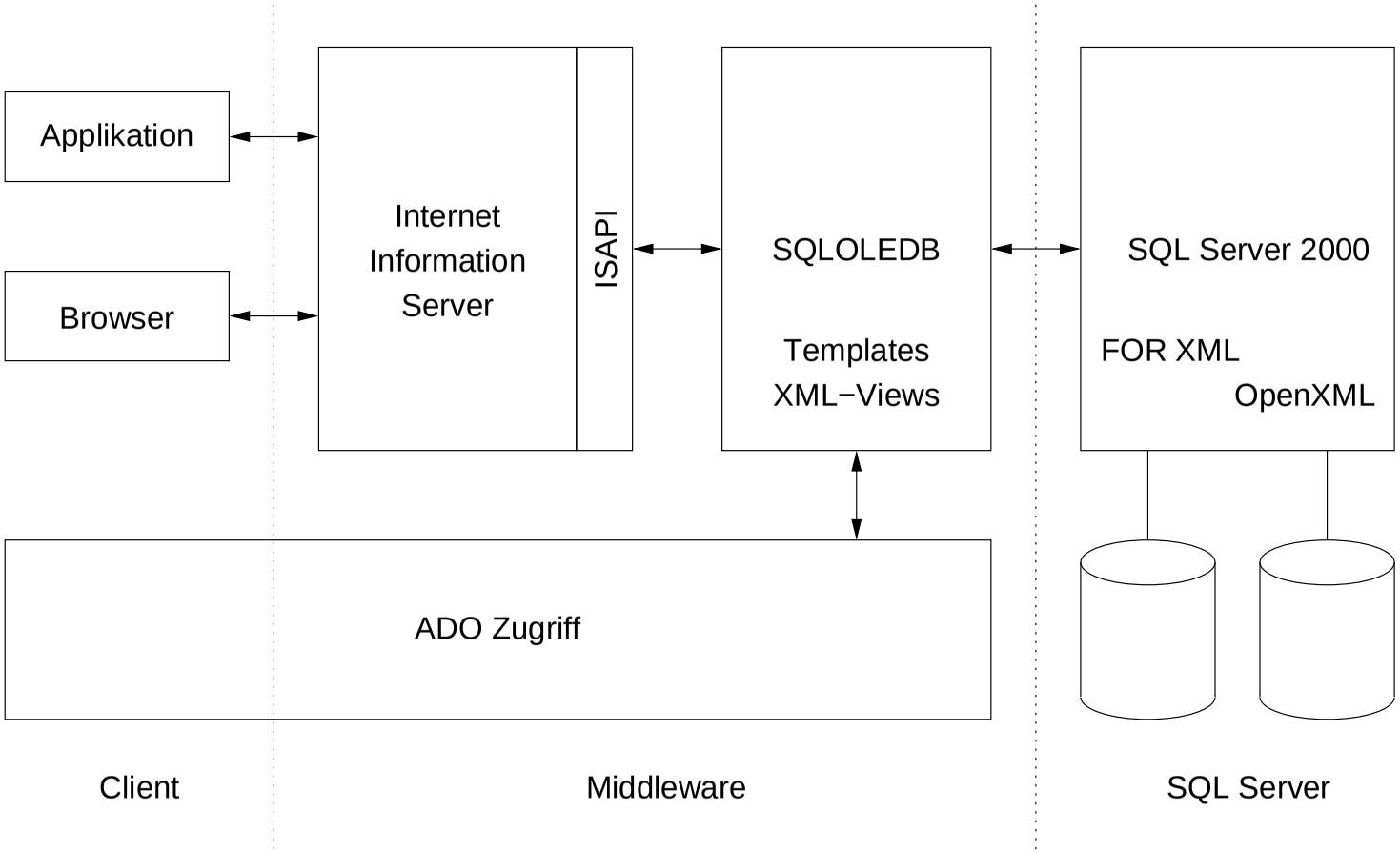
Eigenschaften der XML-Speicherung mit Oracle (cont'd)

XML-Import, XML-Export	
Export in festes Format, Import aus feststehendem Format, weitere Anpassungen mit XSLT	
Anfrage- und Manipulationsschnittstellen	
Anfragen	XPath 1.0 auf **`XMLType`**, SQL bei Speicherung in Oracle
Volltextfunktionalität	$\surd$(Information Retrieval mit Intermedia Text)
Änderungsoperationen	eingeschränkt über Löschen und Hinzufügen vollständiger Werte für **`XMLType`**
Transformationen	XSLT
Werkzeuge, Besonderheiten	
XSLT-Prozessor, XML-Prozessoren (DOM, SAX), XML Schema Validator	

4. Microsoft SQL Server 2000 und XML-Unterstützung

- Import und Export von XML-Dokumenten
- XPath-Ausdrücke über XML-Dokumente mit **`OpenXML`**-Funktion
- Anfrageergebnisse als XML-Dokumente ausgeben mit **`FOR XML`**-Klausel
- Abfrage von relationalen Daten über XML-Sichten
- auch Manipulation mit *Updategramme* von XML aus
- XML-Sichten mit *XDR* definiert

XML-Unterstützung und SQL Server 2000



Darstellung von SQL-Anfrageergebnissen als XML

1. **`RAW`**: erzeugt für jedes Ergebnistupel ein XML-Element ohne Subelemente.
2. **`AUTO`**: erzeugt ein geschachteltes XML-Dokument mit dem Anfrageergebnis.
3. **`EXPLICIT`**: erlaubt, die Form und Schachtelung des Ergebnisdokumentes in der Anfrage selbst zu bestimmen.

RAW-Modus

```
SELECT Hotel.Name, Zimmertyp.Typ, Zimmertyp.Preis
FROM Hotel, Zimmertyp
WHERE Hotel.Name = Zimmertyp.HName
FOR XML RAW
```

```
<row Name="Hotel Neptun" Typ="EZ" Preis="120"/>
<row Name="Hotel Neptun" Typ="DZ" Preis="170"/>
<row Name="Hotel Neptun" Typ="Suite" Preis="350"/>
```

AUTO-Modus

```
SELECT Hotel.Name, Zimmertyp.Typ, Zimmertyp.Preis
FROM Hotel, Zimmertyp
WHERE Hotel.Name = Zimmertyp.HName
FOR XML AUTO
```

```
<Hotel Name="Hotel Neptun">
  <Zimmertyp Typ="EZ" Preis="120"/>
  <Zimmertyp Typ="DZ" Preis="170"/>
  <Zimmertyp Typ="Suite" Preis="350"/>
</Hotel>
```

EXPLICIT-Modus

Generierung in zwei Schritten:

1. der Bearbeitung einer Anfragesicht und
2. einem Serialisierungsprozeß zur Ergebnisaufbereitung.

Kodierung der Spaltennamen:

Name!*Ebene*!*Subname*[!*Type*]

EXPLICIT-Modus — Anfrage

```
SELECT 1 AS Tag,
  NULL AS Parent,
  Hotel.Name AS "Hotel!1!Name",
  NULL AS "Zimmertyp!2!Typ"
FROM Hotels
UNION ALL
SELECT 2, 1, Hotel.Name, Zimmertyp.Typ
WHERE Hotel.Name = Zimmertyp.HName
ORDER BY [Hotel!1!Name], [Zimmertyp!2!Typ] FOR XML EXPLICIT
```

`EXPLICIT`-Modus — Universal Format

Tag	Parent	Hotel!1!Name	Zimmertyp!2!Typ
1	NULL	Neptun	NULL
2	1	Neptun	EZ
2	1	Neptun	DZ
2	1	Neptun	Suite
1	NULL	Am Leuchtturm	NULL
2	1	Am Leuchtturm	EZ
2	1	Am Leuchtturm	DZ

EXPLICIT-Modus — Serialisierung

```
<Hotel Name="Neptun">
  <Zimmertyp Typ="EZ"/>
  <Zimmertyp Typ="DZ"/>
  <Zimmertyp Typ="Suite"/>
</Hotel>
<Hotel Name="Am Leuchtturm">
  <Zimmertyp Typ="EZ"/>
  <Zimmertyp Typ="DZ"/>
</Hotel>
```

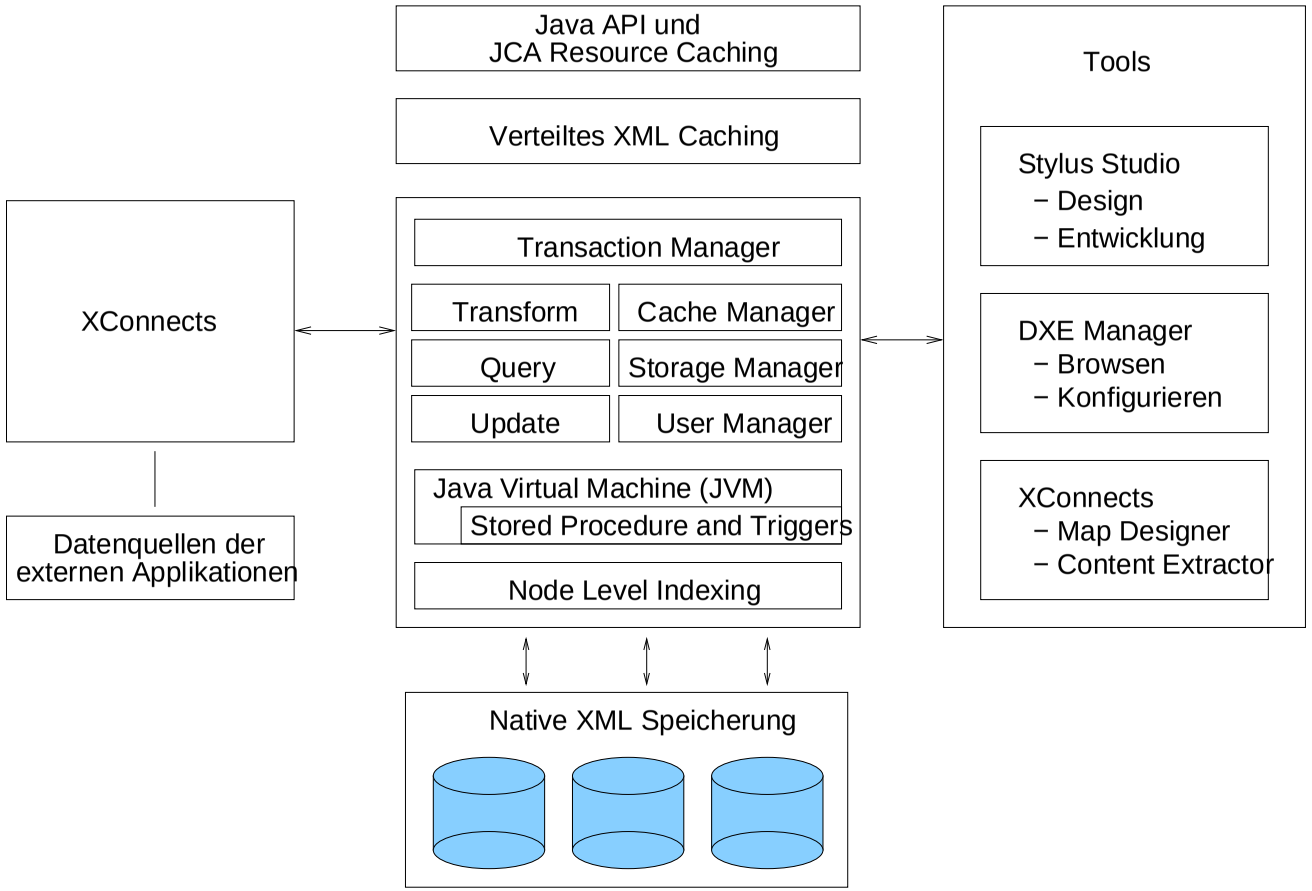
Eigenschaften der XML-Speicherung im SQL Server 2000

XML-Speicherung	
Modell	relationales Modell
Schemabeschreibung	XDR für XML-Sichten, DTD oder Schema zur Typableitung in **OpenXML**
Art der Speicherung	fragmentierte Speicherung
Ordnungserhaltung	–
XML-Datentyp	–
Indizierung	
Struktur, Pfade	–
Werte	√
Volltext	–
XML-Import, XML-Export	
Import mittels **OpenXML**, Export durch annotiertes XDR-Schema	

Eigenschaften der XML-Speicherung im SQL Server 2000

Anfrage- und Manipulationsschnittstellen	
Anfragen	SQL + Erweiterungen, XPath
Volltextfunktionalität	–
Änderungsoperationen	an XML mit *Updategrammen*
Transformationen	XSLT 1.0
Werkzeuge, Besonderheiten	
Integration mit Web-Services über Middleware	
Laden von XML-Massendaten	
basiert auf relationalem Modell oder XML	

5. eXtensible Information Server von eXcelon

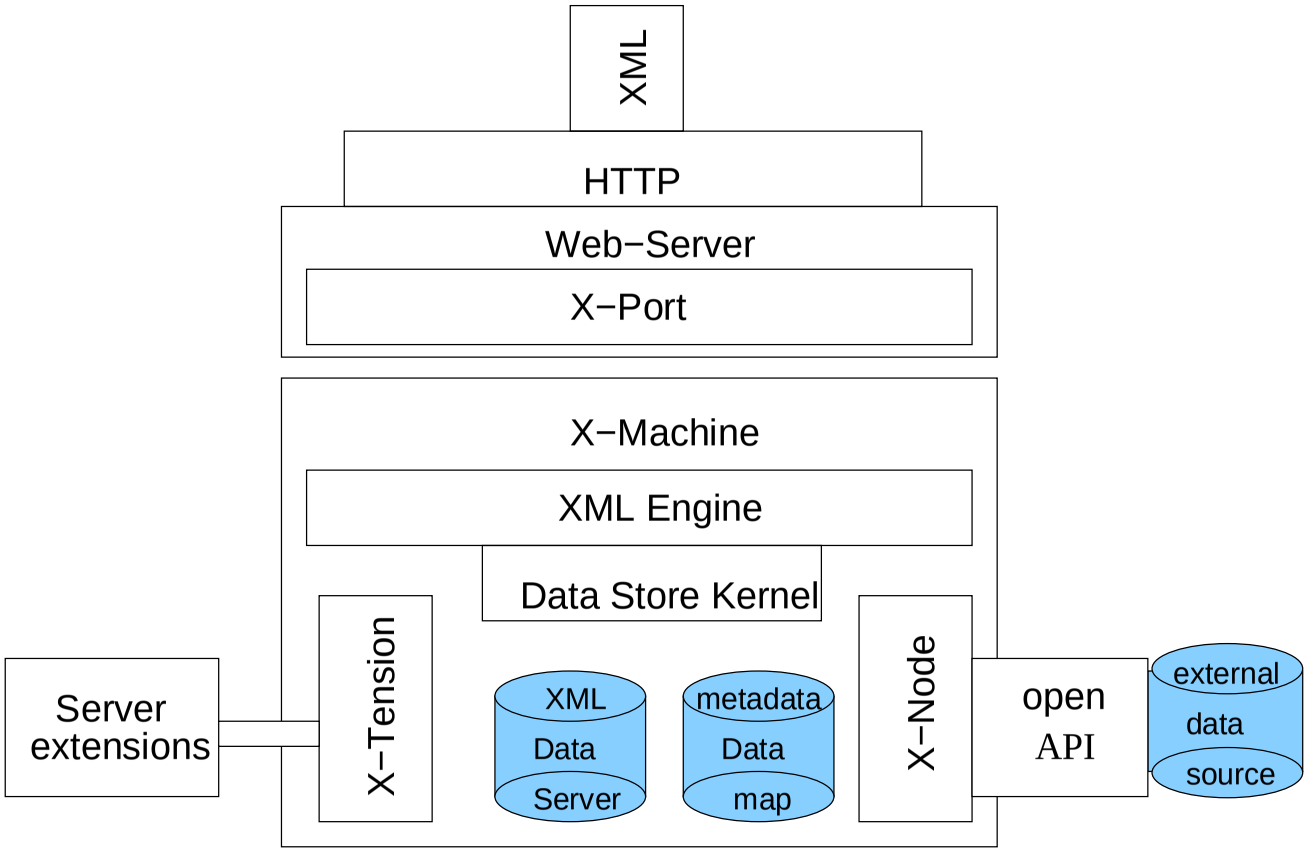


Merkmale des eXtensibel Information Server von eXcelon

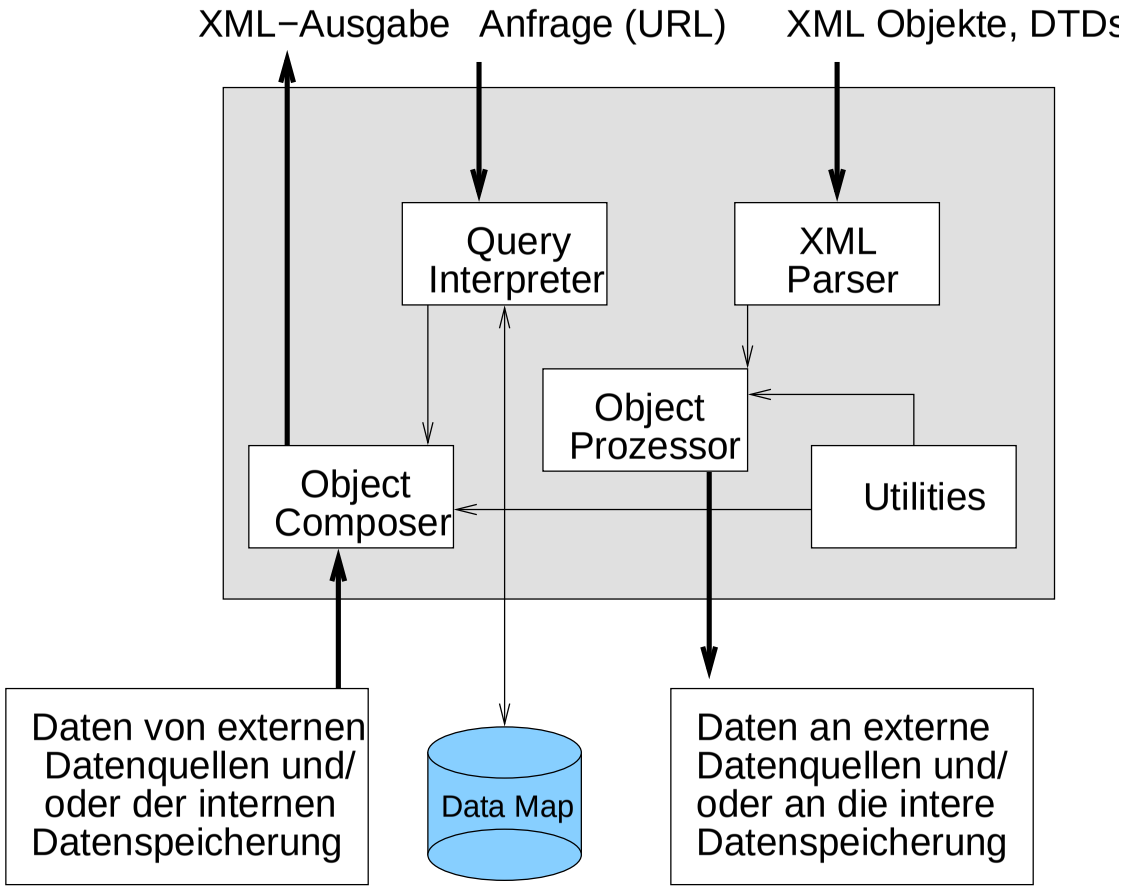
XML-Speicherung	
Modell	Struktur des DOM, Basis für die Speicherung ist ObjectStore
Schemabeschreibung	Test auf Gültigkeit, Schema ist keine Voraussetzung für die Speicherung
Art der Speicherung	vollständige Speicherung
Ordnungserhaltung	$\surd$
XML-Datentyp	$-$
Indizierung	
Struktur, Pfade	$\surd$
Werte	$\surd$
Volltext	$\surd$

XML-Import, XML-Export	
Transformation in über 80 verschiedene Formate Spezifikation der Abbildung über Mappingvorschriften	
Anfrage- und Manipulationsschnittstellen	
Anfragen	XPath 1.0, Teilrealisierung von XQuery 1.0
Volltextfunktionalität	Volltext-Retrieval
Änderungsoperationen	eigene Updatesprache XUpdate (an XPath und SQL angelehnt)
Transformationen	XSLT 1.0
Werkzeuge, Besonderheiten	
Mehrnutzerbetrieb, Caching Editoren für XML-Dokumente, XML Schema, für Anfrage- und Update-Sprache sowie für Mapping-Angaben	

6. Tamino


XML
HTTP
Web−Server
X−Port
X−Machine
XML Engine
Data Store Kernel
Server extensions
X−Tension
XML Data Server
metadata Data map
X−Node
open API
external data source

Speicherung und Retrieval von XML-Dokumenten



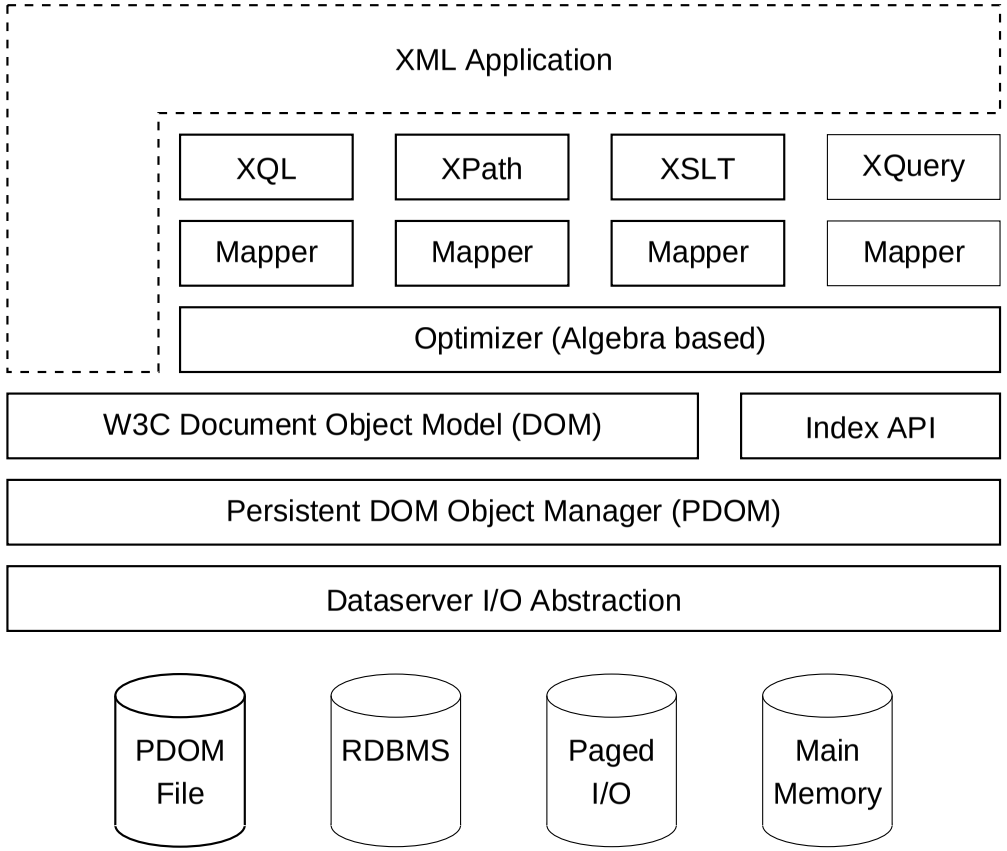
Merkmale des Tamino Server

XML-Speicherung	
Modell	native Speicherung, Basis adabas-Komponenten
Schemabeschreibung	DTD + Datentypen XML Schema, Test auf Gültigkeit, Schema ist zur Speicherung von XML-Dokumenten nicht erforderlich
Art der Speicherung	vollständige Speicherung
Ordnungserhaltung	$\surd$
XML-Datentyp	$-$
Indizierung	
Struktur, Pfade	$\surd$
Werte	$\surd$
Volltext	$\surd$

Merkmale des Tamino Server (cont'd)

XML-Import, XML-Export	
Verarbeitung externer Informationen aus Datenbanken und Spezifikation von Zuordnungsvorschriften, Einbeziehung weiterer Formate	
Anfrage- und Manipulationsschnittstellen	
Anfragen	XPath 1.0, zukünftig XQuery auf XPath 2.0
Volltextfunktionalität	$\surd$
Änderungsoperationen	?
Transformationen	XSLT 1.0
Werkzeuge, Besonderheiten	
Transaktionsverwaltung, Mehrnutzerbetrieb graphische Unterstützung der Angabe des Mappings	

7. Infonyte-DB



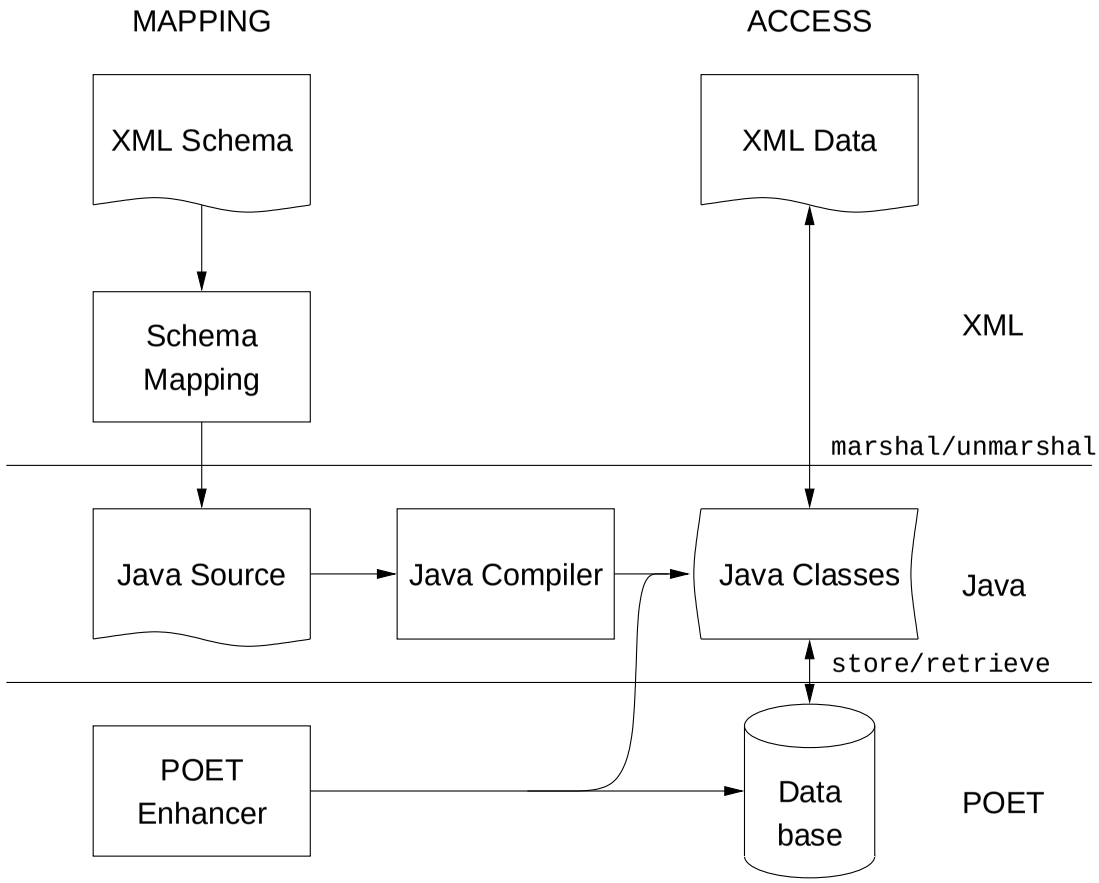
Merkmale der Infonyte-DB

XML-Speicherung	
Modell	Struktur des DOM
Schemabeschreibung	Schema ist keine Voraussetzung für die Speicherung
Art der Speicherung	vollständige Speicherung
Ordnungserhaltung	√
XML-Datentyp	√
Indizierung	
Struktur, Pfade	√
Werte	√
Volltext	√
XML-Import, XML-Export	
	nein

Merkmale der Infonyte-DB (cont'd)

Anfrage- und Manipulationsschnittstellen	
Anfragen	XPath 1.0, Teilrealisierung von XQuery 1.0
Volltextfunktionalität	–
Änderungsoperationen	über DOM-Methoden
Transformationen	XSLT 1.0

8. POET Object Server Suite



Merkmale der XML-Speicherung mit POET

XML-Speicherung	
Modell	strukturierte Speicherung auf Basis des objektorientierten Modells, alternativ Speicherung der DOM-Informationen
Schemabeschreibung	für strukturierte Speicherung erforderlich
Art der Speicherung	fragmentiert oder vollständig
Ordnungserhaltung	bei Speicherung der DOM-Informationen gegeben
XML-Datentyp	nein
Indizierung	
Struktur, Pfade	$\surd$
Werte	$\surd$
Volltext	–

Merkmale der XML-Speicherung mit POET (cont'd)

XML-Import, XML-Export	
über Java-Methoden	
Anfrage- und Manipulationsschnittstellen	
Anfragen	OQL
Volltextfunktionalität	–
Änderungsoperationen	OQL
Transformationen	–

9. Content Management Suite — CMS

XML-Speicherung	
Modell	strukturiert, objektorientiertes Modell
Schemabeschreibung	(Gültigkeit, Voraussetzung für Mapping)
Art der Speicherung	fragmentierte Speicherung
Ordnungserhaltung	$\surd$
XML-Datentyp	–
Indizierung	
Struktur, Pfade	$\surd$
Werte	$\surd$
Volltext	$\surd$
XML-Import, XML-Export	
über Java-Methoden	

Merkmale der XML-Speicherung mit CMS (cont'd)

Anfrage- und Manipulationsschnittstellen	
Anfragen	Navigations-API
Volltextfunktionalität	boolesche Suche, Phrasen, Proximity, linguistische Techniken, Stoppworte, Grundform, Stammform, Thesauri
Änderungsoperationen	Navigations-API?
Transformationen	
Werkzeuge, Besonderheiten	
Multi-User-Betrieb, Locking auf Komponentenebene, Versionsverwaltung, Rechtevergabe auf Objektebene, Einbindung von Word, Excel, PDF Editoren zur Erstellung von XML-Dokumenten und Schemata	

9.1. Weitere Systeme

- Natix
- X-Hive
- Bluestone
- Xyleme
- Xindice
- . . .
- im Bereich nativer XML-Server nicht mehr überschaubar