Kapitel 10

XML

Inhalt

- ❑ Einführung
- ❑ XML-Dokumente
- ❑ Techniken
  - ➩ DTD
  - ➩ XSL
  - ➩ XPath
  - ➩ XML-Schema
  - ➩ XQuery
  - ➩ SQL/XML
- ❑ Dokumenten-zentrierte vs. Daten-zentierte Sicht
- ❑ Zusammenfassung und Ausblick

Einführung

- ❑ ***HTML***
  - ➩ *Hypertext Markup Language*
  - ➩ basiert auf Metasprache ***SGML**:*
    Standard Generalized Markup Language
    - kompliziert
  - ➩ eher geeignet für einfache Dokumente
- ❑ Ziele des W3C (World Wide Web Consortium) bei Entwicklung von ***XML***:
  - ➩ einfach erlern- und anwendbar
  - ➩ ***Trennung von Daten und Präsentation***
  - ➩ ***Erweiterbarkeit***
  - ➩ ***Möglichkeit der strukturellen Validierung***
  - ➩ Systemunterstützung (Parser) einfach integrierbar
- ❑ ***XML***
  - ➩ *Extensible Markup Language*
  - ➩ Teilmenge von *SGML*

XML-Dokumente (1)

- ❑ Beispiel

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE Memo SYSTEM "http://www.fck.de/DTDs/memo.dtd">
```

PROLOG

```
<Memo>
    <Von>
        <Name>Andi Brehme</Name>
        <Email>Andi Brehme@fck.de</Email>
    </Von>
    <An>
        <Name>Vorstand</Name>
        <Email>Vorstand@fck.de</Email>
    </An>
        <Betreff Dringlichkeit="hoch">Spielereinkauf</Betreff>
    <Inhalt>
        <Absatz>Vorstand, ich glaube wir <Betonung>muessen
               </Betonung> unbedingt noch einige neue Spieler
               kaufen, sonst wird das wohl nichts mit der
               Meisterschaft.
        </Absatz>
    </Inhalt>
</Memo>
```

Dokumentinstanz

XML-Dokumente (2)

- ❑ Prolog
  - ⇨ XML-Deklaration
    - Versionsangabe
    - Kodierungsdeklaration (optional)
    - *standalone*-Angabe (optional)
  - ⇨ Dokumenttyp-Deklaration
    - Angabe der verwendeten *DTD*
- ❑ XML-Dokument besteht aus ***Markup-Elementen***
- ❑ Markup-Element
  - ⇨ besteht aus Tag-Paar (Begin-Tag, End-Tag)
    - Bsp: `<Name>Harry Koch</Name>`
      `<LeeresElement/>`
  - ⇨ kann leer sein
  - ⇨ Schachtelung möglich
  - ⇨ enthält Zeichendaten
  - ⇨ Elementnamen werden bzgl. Gross-/Kleinschreibung unterschieden
  - ⇨ können näher durch ***Attribute*** beschrieben werden

XML-Dokumente (3)

❑ ***Attribute***

- haben Namen (innerhalb eines Elements eindeutig) und Wert
- 3 Typen
  - Zeichendaten
  - Aufzählungen
  - über Schlüsselwörter, wie ID oder IDREF, ausgezeichnete Zeichendaten
- es ergeben sich verschiedene Möglichkeiten der Darstellung derselben Information
  - Unterelementen

```
<Spieler>
   <Nummer>24</Nummer>
   <Tore>5</Tore>
   <Name>Harry Koch</Name>
</Spieler>
```

  - Attribute

```
<Spieler Nummer="24" Tore="5"
Name="Harry Koch"/>
```

  - gemischt

```
<Spieler Nummer="24" Tore="5">Harry
Koch</Spieler>
```


XML-Dokumente (4)

❑ ***Entities***

- bilden physikalische Dokumentstruktur
- Name
- Inhalt
  - parsed
  - unparsed
- Unterscheidung
  - extern
  - intern
  - allgemein
  - Dokument
  - ...
- Entity-Referenzen
- Beispiele

```
<!ENTITY FckLogo
SYSTEM "http://www.fck.de/wappen.gif"
NDATA GIF>

<!ENTITY FCK "1. FC Kaiserslautern">
<Text>Andi Brehme ist Trainer des
&FCK;.</Text>
```

DTD (1)

- *DTD*: *Document Type Definition*

```
<!DOCTYPE Memo[
  <!ELEMENT Memo (Von, An, Betreff, Inhalt)>
  <!ELEMENT Von (Name, Email)>
  <!ELEMENT An (Name, Email)>
  <!ELEMENT Name (#PCDATA)>
  <!ELEMENT Email (#PCDATA)>
  <!ELEMENT Spieler ANY>
  <!ATTLIST Spieler
     Dringlichkeit (niedrig| mittel| hoch)
        "niedrig"
  <!ELEMENT Inhalt (Absatz+)>
  <!ELEMENT Absatz (#PCDATA|Betonung)*>
  <!ELEMENT Betonung (#PCDATA)>
]>
```


DTD (2)

- Syntax
  - dient struktureller Validierung
    - ***well-formed***: konform zu XML-Standard
    - ***valid***: konform zu angegebener DTD
  - kann extern
    `<!DOCTYPE Memo`
    `SYSTEM "http://www.fck.de/DTDs/memo.dtd">`
    oder intern deklariert sein
  - Inhaltsspezifikationen für Elemente
    - Text (parsed character data)
      `<!ELEMENT Email (#PCDATA)>`
    - leerer Inhalt
      `<!ELEMENT Email EMPTY>`
    - Beliebiger Inhalt
      `<!ELEMENT Email ANY>`
    - Kombinierter Inhalt
      `<!ELEMENT Email`
      `(#PCDATA | Element1 | Element2)*>`
    - Elementeninhalt
      `<!ELEMENT Email (Element1,`
      `(Element2, Element3)+, Element4?)>`
      (Indikatoren für das Auftreten: ?, *, +)

DTD (3)

- Syntax (Forts.)
  - Attributlistendeklaration
    - Attributtyp
      - Aufzählungstyp
      - **CDATA**
      - **ID**
      - **IDREF**
      - **IDREFS**
      - **ENTITY**
      - ...
    - optional: Status, Defaultwert
      - **#REQUIRED**
      - **#IMPLIED**
      - **[#FIXED] <default value>**
    - Beispiele:

```
<Spieler Nummer="24" Tore="5">Harry
   Koch</Spieler>
```

```
<!ATTLIST Spieler
   Nummer CDATA #REQUIRED
   Tore   CDATA "0">
```


XSL (1)

- *XSL*: *Extensible Stylesheet Language*
  - darstellungsunabhängiges Markup von XML
  - medienunabhängige Präsentation
  - ***XSLT***: XSL Transformation Language
    - Formatvorlage (*stylesheet*)
    - Transformationsregeln
      - jeweils bestehend aus einem *Pattern* und einem *Template*
    - Ursprungsbaum
    - Ergebnisbaum

XSL (2)

❑ Beispiel

➪ ein Fragment eines XML-Dokuments:

```
<Inhalt>
   <Absatz>XSLT <Fremdwort> (XSL Transformation Language) </Fremdwort>
            ist eine <Betonung> phantastische</Betonung> Sprache um
            XML-Dokumente in XHTML <Fremdwort> (Extensible HTML)
            </Fremdwort> zu konvertieren.
   </Absatz>
</Inhalt>
```

➪ das gewünschte Ergebnis-Dokument:

```
<html>
   <head>
      <title>Ein XSLT-Beispiel</title>
   </head>
   <body>
      XSLT <i>(XSL Transformation Language)</i> ist eine
      <b>phantastische</b> Sprache um XML-Dokumente in XHTML
      <i>(Extensible HTML)</i> zu konvertieren.
   </body>
</html>
```

➪ Präsentation:

XSLT *(XSL Transformation Language)* ist eine **phantastische** Sprache um XML-Dokumente in XHTML *(Extensible HTML)* zu konvertieren.

XSL (3)

❑ Beispiel (Forts.)

➪ das XSLT-Stylesheet:

```
<xsl:stylesheet    xmlns:xsl="http://w3.org/XSL/Transform/1.0"
                   xmlns="http://w3.org/TR/xhtml1"
                   indent-rules="yes">
<!-- Regel 1 -->   <xsl:template match="/">
                       <html>
                       <head>
                           <title>Ein XSLT-Beispiel</title>
                       </head>
                       <body>
                           <xsl:apply-templates/>
                       </body>
                       </html>
                   </xsl:template>
<!-- Regel 2 -->   <xsl:template match="Absatz">
                       <xsl:apply-templates/>
                   </xsl:template>
<!-- Regel 3 -->   <xsl:template match="Betonung">
                       <b><xsl:apply-templates/></b>
                   </xsl:template>
<!-- Regel 4 -->   <xsl:template match="Fremdwort">
                       <i><xsl:apply-templates/></i>
                   </xsl:template>
```

XPath

- *XPath*: *XML Path Language*
  - Beschreibung der *Pattern* in XSLT
  - Angabe der Position eines Knotens
    - absolut bzgl. Wurzelknoten
    - relativ bzgl. eines Kontextknotens
  - Pfadausdruck
    - *steps* (durch “/” getrennt)
      - *basis*
        - *axis (descendant, child, following, following-sibling, ancestor, parent, preciding, preciding-sibling, self, descendant-or-self, ancestor-or-self)*
        - *node test*
      - optionale Liste von *predicates*
    - Beispiel:
      `id("Mannschaft")/descendant::Spieler/child::Name[position()=last()]`
    - abgekürzte Notation:
      `id("Mannschaft")//Spieler/Name[position()=last()]`

XML-Schema (1)

- Erweiterung der Möglichkeiten zur Beschreibung von Dokumenttypen im Vergleich zu DTD
  - Beschreibung von Dokumenten-Modellen
    - Kardinalitätsrestriktionen
    - Vererbung
  - Datentypen
    - vielfältige vordefinierte Datentypen, z. B. Integer, Date
    - benutzerdefinierte, komplexe Datentypen
    - Constraints
  - Referenzen
    - bisher ‘nur’ Wertübereinstimmung von ID- und IDREF(S)-Attributen
    - nun getypte Referenzen auf bestimmte Elemente/Attribute
- seit Mai 2001 *W3C Recommendation*

XML-Schema (2)

- Beispiel
  - DTD

```
<!ELEMENT description
     ((content, date*)+, author?)>
<!ELEMENT content (#PCDATA)>
<!ATTLIST content info CDATA #REQUIRED>
<!ELEMENT date (#PCDATA)>
<!ELEMENT author (#PCDATA)>
```

  - XML-Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns=
  "http://www.w3.org/2001/XMLSchema">
  <element name="description"
type="mycontent"/>
  <complexType name="mycontent">
    <sequence>
      <sequence maxOccurs="unbounded">
        <element name="content"
               type="mytype"/>
        <element name="date" type="date"
             minOccurs="0"
               maxOccurs="unbounded"/>
      </sequence>
      <element name="author"
             type="string" minOccurs="0"/>
    </sequence>
  </complexType>
  <complexType name="mytype">
    <simpleContent>
      <extension base="string">
        <attribute name="info"
           type="string" use="required"/>
      </extension>
    </simpleContent>
  </complexType>
</schema>
```


Dokumenten- vs Datensicht

- Datenzentrierte Sicht
  - fein-granulare Daten
  - Reihenfolge nicht signifikant
  - Beispiele: Flugpläne, Speisekarten, ...
  - maschinelle Verarbeitung
- Dokumentzentrierte Sicht
  - grob-granulare Daten
  - Reihenfolge signifikant
  - Beispiele: Bücher, Email, Werbung, ...
  - Nutzung durch Personen
- XML-Datenbanken
  - XML-generierende (XML-enabled) DB
  - XML-Dokument-DB (native XML-DB)
  - XML-Komponenten-DB (*XML awareness*)

XQuery

- Motivation
  - Anfragen über XML
    - physisches XML Dokument (z.B. als Datei gespeichert)
    - logische XML Sicht (z.B. über relationaler Datenbank)
  - soll sowohl Daten- als auch Dokument-zentrierte Sicht unterstützen
  - Public Working Draft des W3C
- Grundlagen
  - funktionale Sprache
    - basiert auf Komposition von funktionalen Ausdrücken
  - Erweiterung von XPath
    - XPath 2.0 wird von XQuery, XSLT WGs weiterentwickelt
  - XQuery Datenmodell erweitert XPath 1.0 DM
    - collections of documents
    - XQuery ist abgeschlossen bzgl. DM
- Wichtige Erweiterungen
  - FLWR-expressions (For-Let-Where-Returns)
    - Iteration
    - Binden von Variablen
  - Konstruktoren
    - XML Syntax

XQuery Beispiele

- Beispiel 1

```
FOR $b IN document("bib.xml")//book
WHERE $b/publisher = "Morgan Kaufmann"
  AND $b/year = "1998"
RETURN $b/title
```

- Beispiel 2

```
FOR   $p   IN   distinct(document("bib.xml")//
publisher)
LET $a := avg(document("bib.xml")
     /book[publisher = $p]/price)
RETURN
  <publisher>
    <name> { $p/text() } </name>
    <avgprice> { $a } </avgprice>
  </publisher>
```

- Beispiel 3

```
<author_list>
   { FOR $a IN distinct(document("bib.xml")//
author)
     RETURN
       <author>
         <name> { $a/text() } </name>
             { FOR $b IN document("bib.xml")//
book[author = $a]
               RETURN $b/title        }
       </author>    }
</author_list>
```

SQL/XML

- Teilprojekt der SQL Standardisierung
  - Part 14 “XML-related Specifications”
  - derzeitiger Zustand: Final Committee Draft

- Ziel: Standardisierung verschiedener Aspekte der Interaktion/Integration von SQL und XML
  - Wie werden SQL-Daten (Tabelleninhalte, Anfrageresultate, ...) in XML dargestellt (und umgekehrt)?
  - Wie werden SQL Schemata in XML Schema Definitionen abgebildet (und umgekehrt)?
  - ....
- Mögliche Einsatzbereiche
  - Einfache Umwandlung von SQL Daten in XML
  - Integration von XML Daten in SQL Datenbanken
  - XML zum Austausch von SQL Daten
  - Erzeugen von XML “Views” auf relationalen Daten
    - mögl. Basis für XQuery

SQL/XML (2)

- Abbildungs-Infrastruktur
  - SQL Character Sets <-> Unicode
  - SQL identifier <-> XML name
    - problematisch sind bspw. “_”, “ “, “:” in SQL delimited identifiers
  - SQL Datentypen <-> XML Schema Datentypen
    - die beste Entsprechung, die alle erlaubten Werte des jeweiligen SQL Typs in XML repräsentieren kann
    - entsprechend der Typabbildung ist auch die Wertabbildung definiert, z.B.

SQL data type	SQL literal	XML value
VARCHAR (10)	'Smith'	Smith
INTEGER	10	10
DECIMAL (5,2)	99.52	99.52
TIME	TIME'12:30:00'	12:30:00
INTERVAL HOUR TO MINUTE	INTERVAL'2:15'	PT02H15M

- Abbildung von SQL Tabellen, Schemas, Katalogen in
  - XML Dokument (Daten)
  - XML Schema Dokument (Metadaten)

SQL/XML Mapping

- Abbildungsbeispiel
  - SQL Tabelle “EMPLOYEE”
  - XML Dokument:

```
<EMPLOYEE>
 <row>
       <EMPNO>000010</EMPNO>
       <FIRSTNME>CHRISTINE</FIRSTNME>
       <LASTNAME>HAAS</LASTNAME>
       <BIRTHDATE>1933-08-24</BIRTHDATE>
       <SALARY>52750.00</SALARY>
 </row>
 <row>
       <EMPNO>000020</EMPNO>
       <FIRSTNME>MICHAEL</FIRSTNME>
       <LASTNAME>THOMPSON</LASTNAME>
       <BIRTHDATE>1948-02-02</BIRTHDATE>
       <SALARY>41250.00</SALARY>
 </row>
...
</EMPLOYEE>
```


SQL/XML Mapping (2)

- XML Schema Dokument:

```
<xsd:schema>

      <xsd:simpleType name="CHAR_6">
            <xsd:restriction base="xsd:string">
                  <xsd:length value="6"/>
            </xsd:restriction>
      </xsd:simpleType>
...
      <xsd:simpleType name="DECIMAL_9_2">
            <xsd:restriction base="xsd:decimal">
                  <xsd:totalDigits value="9"/>
                  <xsd:fractionDigits value="2"/>
            </xsd:restriction>
      </xsd:simpleType>

      <xsd:complexType name="RowType.HR.ADMINISTRATOR.EMPLOYEE">
            <xsd:sequence>
              <xsd:element name="EMPNO" type="CHAR_6"/>
              <xsd:element name="FIRSTNME" type="VARCHAR_12"/>
              <xsd:element name="LASTNAME" type="VARCHAR_15"/>
              <xsd:element name="BIRTHDATE" type="DATE" nillable="true"/>
              <xsd:element name="SALARY"
                  type="DECIMAL_9_2" nillable="true"/>
            </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="TableType.HR.ADMINISTRATOR.EMPLOYEE">
            <xsd:sequence>
                  <xsd:element name="row"
                  type="RowType.HR.ADMINISTRATOR.EMPLOYEE"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
      </xsd:complexType>

      <xsd:element name="EMPLOYEE"
            type="TableType.HR.ADMINISTRATOR.EMPLOYEE"/>

</xsd:schema>
```

SQL/XML “publishing functions”

- SQL/XML definiert neue SQL Funktionen/Operatoren, aus SQL Daten in XML Konstrukte (Elemente, attribute, ...) erzeugen.
  - XMLCONCAT konkateniert XML Werte
  - XMLELEMENT erzeugt ein XML Element
  - XMLFOREST erzeugt einen Wald von Elementen
  - XMLGEN erzeugt Element basierend auf XQuery Konstruktorsyntax
  - XMLAGG aggregiert XML über Tupel hinweg
- Beispiel

```
SELECT e.id,
     XMLELEMENT ( NAME "Emp", e.fname || ' ' || e.lname)
          AS "result"
FROM employees e
WHERE ... ;
```

  - Resultat:

ID	**result**
1001	<Emp>John Smith</Emp>
1206	<Emp>Mary Martin</Emp>

- Geplante Erweiterungen:
  - Abbildung von benutzerdefinierten Datentypen
  - Funktionen XMLParse, XMLSerialize, XMLExtract

Zusammenfassung und Ausblick

- XML
  - Datenaustausch
    - Integration durch “gemeinsame Sprache”
  - Datenspeicherung
    - Flexibilisierung der Dokument- und Datenverwaltung
- Ausblick