



Semistrukturierte Daten

Sommersemester 2011

Teil 1: XML-Einführung

1.1. "Semistrukturierte" Daten

1.2. Entwicklung von XML

1.3. Aufbau von XML-Dokumenten



1.1. "Semistrukturierte Daten"

- "Wesen" von semistrukturierten Daten
- Dokumente vs. Daten

"Wesen" von Semistrukturierten Daten

- (relationales) Datenmodell ist eventuell zu starr:
flexiblere Struktur (als Tabellen mit fix vorgegebenen Spalten mit fixem Datentyp)
- Schema ist nicht vorhanden (oder nicht sichtbar für den Benutzer):
 - **Selbstbeschreibend** (Strukturinformation als Teil der Daten und nicht in explizitem Schema)
 - Ein **Schema** kann **optional** vorhanden sein.

Flexible Struktur

- Die Suche nach Modellen für flexible und unregelmäßige Datenstruktur führte zu Modellen für semistrukturierte Daten
 - wenn relationale Datenbank viele Nullwerte hätte
 - wenn Daten unterschiedlichen Typ haben können.
- Gut geeignete Datenstruktur:
 - Baum
 - oder allgemeiner: Graph
- Wir können übliche DB Formate (Tabellen, OO-DB) in diesem Format darstellen.

Selbstbeschreibende Daten

- In Datenbanken wird zuerst ein Schema definiert, d.h.: Struktur + erlaubte Typen
- Bei Daten am Web ist häufig kein Schema vorhanden (bzw. für den Benutzer nicht sichtbar).
- Semistrukturierte Daten:
 - **Selbstbeschreibende Daten**: jeder Eintrag wird explizit mit seiner Beschreibung annotiert.
 - Schema (z.B. DTD oder XML Schema) ist optional
 - Vorteile: Interoperabilität, Erweiterbarkeit
 - Nachteil: Speicherplatzverschwendung bei Standardspeicherungsart (Wiederholung der Bezeichnungen)

Dokumente vs. Daten

■ Dokumente:

- ☐ Reiner Text ist problematisch für automatische Verarbeitung von Inhalten
- ☐ Sichtbarmachen der Struktur (mittels Markup) erforderlich

■ Daten:

- ☐ starre Struktur, fixes Schema
- ☐ Selbstbeschreibende Daten (mittels Markup) geben mehr Flexibilität

■ XML: vereinigt die beiden Sichtweisen

- ☐ Ursprünglich eher Dokumenten-Sicht
- ☐ Mittlerweile Daten-Sicht mindestens gleich wichtig

Anwendungen von XML

■ Dokument-Anwendungen

- "menschlicher" Informationsaustausch
- aber auch für Maschinen interpretierbare Daten
- Klare Trennung von Struktur und Präsentation
 - => transportables und leicht wiederverwendbares Dokument
- "maßgeschneiderte" Präsentation mittels **Stylesheets**

■ Daten-Anwendungen

- Einheitliches, einfaches, robustes Daten-Austauschformat
- automatisierter Datenaustausch mit Datenbanken, Programmen,...
- native XML Datenbanken



Unterschiedliche Forschungsinteressen

■ Dokumentenwelt

- ☐ Präsentationsformate (wie HTML)
- ☐ Informationsaustauschformate
- ☐ Document/Information Retrieval

■ Datenbankwelt

- ☐ Speichertechniken
- ☐ Anfragesprachen
- ☐ Datenmodelle, Methoden zur Strukturierung von Daten
- ☐ Integrität/Konsistenz von Daten

Diese Grenzen verschwimmen.

1.2. Entwicklung von XML

- XML, SGML und HTML
- Applikationen und Standards
- Geschichte
- Tools

XML: eXtensible Markup Language

Wichtige Eigenschaften:

- Genormte, erweiterbare Auszeichnungssprache (W3C):
kein ISO-Standard sondern eine **W3C Recommendation**
- Syntax zur Beschreibung (semi)strukturierter Information
- Trennung von Struktur und Präsentation
- Bereichsspezifische Dokumenttypen ("Applikationen")
- als Datenaustauschformat sehr gut geeignet

"XML will be the ASCII of the Web – basic, essential, unexciting" (Tim Bray)

SGML

- Standard Generalized Markup Language
- ISO Standard (1986)
- Für einzelne Applikationen kann eine bestimmte Dokumentenstruktur vorgegeben werden
 - Elemente und ihre Relationen werden beschrieben
- HTML ist eine Applikation von SGML
 - HTML enthält Formatierungsanweisungen, die von Browsern interpretiert werden
 - HTML-Syntax ist relativ flexibel bzw. Browser sind fehler-tolerant (z.B. inkorrekte Schachtelung von Elementen)

Bestandteile eines SGML-Dokuments

- SGML-Deklaration
 - Definiert „Umgebung“ eines SGML-Dokuments, d.h.: Regeln für DTD und Dokument-Instanz
 - z.B.: Zeichensatz, als Markup zu interpretierende Zeichen, zulässige Länge von Tags, zulässige Schachtelungstiefe, ...
- Dokumenttypdefinition (DTD)
 - Externes/internes DTD-Subset
- Dokument-Instanz
 - d.h.: der mit Markup annotierte Inhalt

Einschränkungen von XML (vs. SGML)

- Max. 2 Bestandteile eines XML-Dokuments:
 - Dokument-Instanz, optional DTD, keine SGML-Deklaration
 - aber implizit gibt es eine fixe SGML-Deklaration für XML
 - d.h.: XML ist eine echte Teilmenge von SGML
- Wohlgeformtheit ohne Gültigkeit möglich
- Keine Namen, die mit [Xx] [Mm] [Li] beginnen
- Keine Auslassung/Abkürzung von Markup
- Attribute haben immer auch einen Attributwert
- Immer Unicode als Basiszeichensatz
- Keine Kapazitätsbeschränkungen

HTML vs. XML

■ HTML

- ☐ eine **Applikation von SGML** (eine fixe DTD)
- ☐ über 100 *fixe* Tags
- ☐ Browser sehr fehlertolerant (ignoriert DTD....)
- ☐ In erster Linie für Präsentation (z.B. boldfaced, rot) bzw. Layout-Struktur (z.B. Tabellen, Listen)
- ☐ Chaos: verschiedenste proprietäre Erweiterungen

■ XML

- ☐ eine **Teilmenge von SGML**
- ☐ Metasprache für Markup Sprachen
- ☐ keine vordefinierten Tags
- ☐ strikte Syntax muss eingehalten werden
- ☐ viele ergänzende Standards (XSD, Xpath, ...)
- ☐ leicht zu lesen und zu verarbeiten

Warum XML?

- Einschränkungen von **HTML**
 - Fix vorgegebene Menge von Elementen
 - keine Trennung von Layout und Struktur
- Komplexität von **SGML**
 - unbrauchbare Optionen für Webapplikationen
 - schwierig für Entwicklung von Tools/Browsern
- Idee von **XML**
 - Vereinfachte Version von SGML (Teilmenge)
 - Optimiert für Informationsbereitstellung im Web
 - Soll HTML/SGML ergänzen (nicht ersetzen)

Was XML nicht ist ...



- XML ist keine Programmiersprache
- XML ist kein Netzwerk-Transportprotokoll
- XML ist keine Datenbank

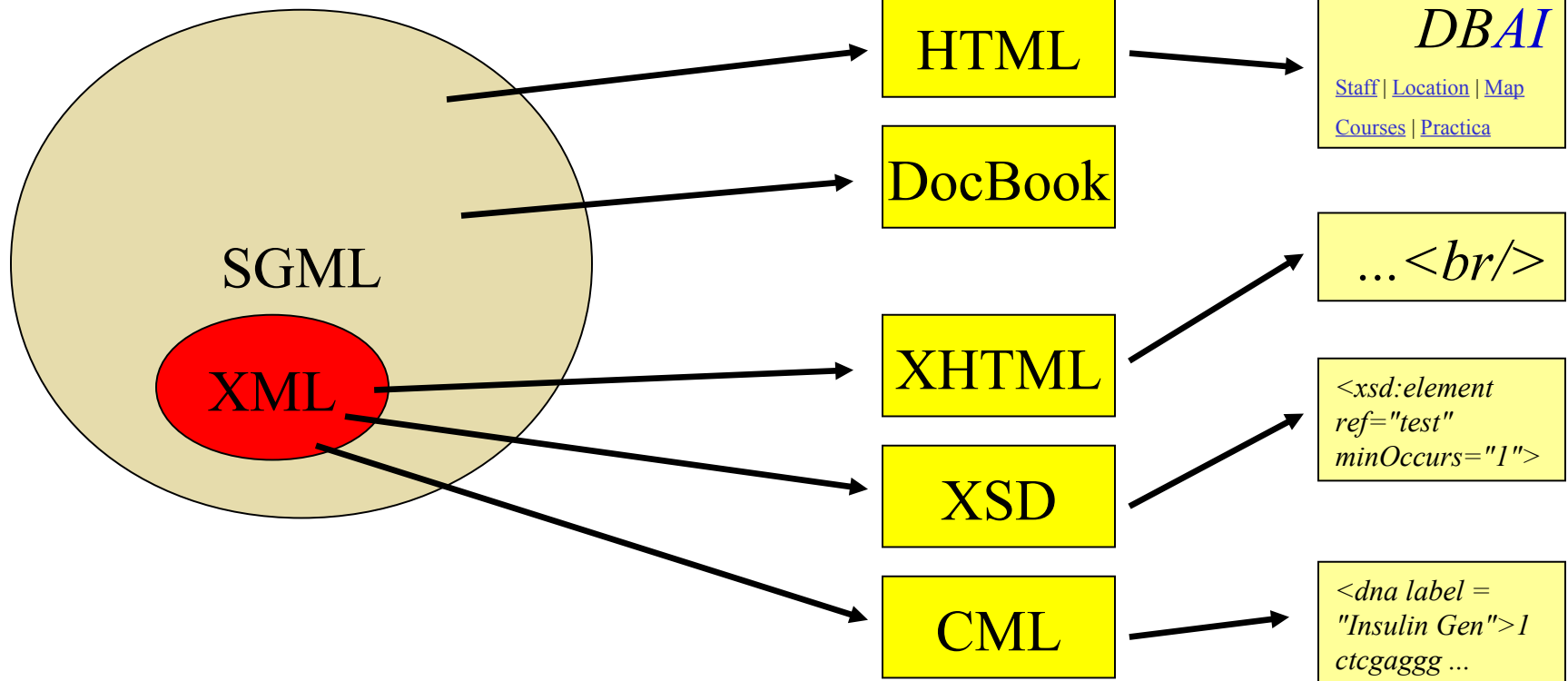
Ein XML-Dokument *existiert* einfach. Es *tut* nichts.

Applikationen und Instanzen

Meta Markup Sprache

Applikationen

Instanzen

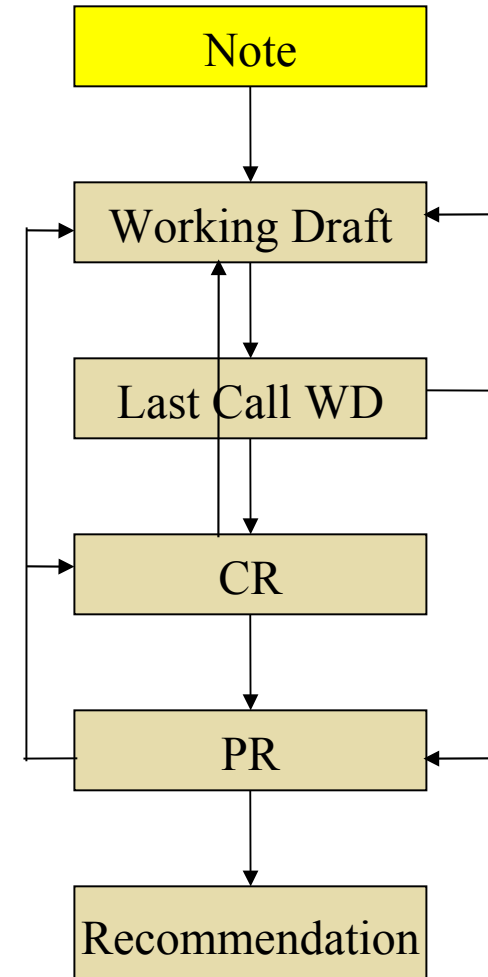


XHTML: HTML als XML Applikation
CML: Chemical Markup Language
XSD: XML Schema Definition

W3C: World Wide Web Consortium

Sechs Arten von Dokumenten

- Note:
 - ◆ kann jedes W3C Mitglied einreichen
 - ◆ noch keine Absichtserklärung des W3C
- Working Draft (WD)
 - ◆ aktueller Diskussionsstand
- Last Call WD
 - ◆ wenn festgelegte Ziele erreicht wurden
- Candidate Recommendation (CR)
 - ◆ Einreichungsbestätigung
- Proposed Recommendation
 - ◆ Implementierungen der Bestandteile sind vorhanden
- Recommendation
 - ◆ W3C "Standard" (ist kein offizieller Standard wie z.B. ein ISO Standard)



Rund um XML: Begleitende Standards

- Namespaces: Teil der XML-Recommendation
- W3C Schema-Sprachen: DTD, XSD
- Navigation: XPath
- Stylesheets: XSL
 - XSLT: Selektion, Transformation
 - XSL-FO: Präsentation
- XQuery: Abfragesprache (im Stil von SQL)
- (quasi-)standardisierte APIs
 - DOM
 - SAX
- viele weitere Standards

Geschichte

- Hypertext (1945)
 - beliebige Navigationspfade durch Dokumente
- GML (SGML Vorläufer) (1969)
- SGML ISO Standard (1986)
- HTML Tim Berners-Lee, CERN (1989)
- W3C gegründet (1994)
- SGML Subset Arbeitsgruppe (1996)
- XML 1.0 (1998)
- XSLT (1999)
- XML Schema (2001)
- laufend neue Recommendations

Einige XML Tools

- Browser
 - IE 6, Mozilla, Amaya (W3C Browser/Editor)
- XML Editoren: Altova's XMLSpy (www.xmlspy.com)
 - textbasiert vs. baumbasiert
- XML Parser:
 - Xerces (<http://xerces.apache.org>)
 - libxml-Bibliothek (<http://xmlsoft.org>)
 - ◆ xmllint – commandline tool
 - ◆ Windowsversion <http://www.zlatkovic.com/pub/libxml>
- XSLT:
 - Xalan (<http://xalan.apache.org>),
 - Saxon (www.saxonica.com).

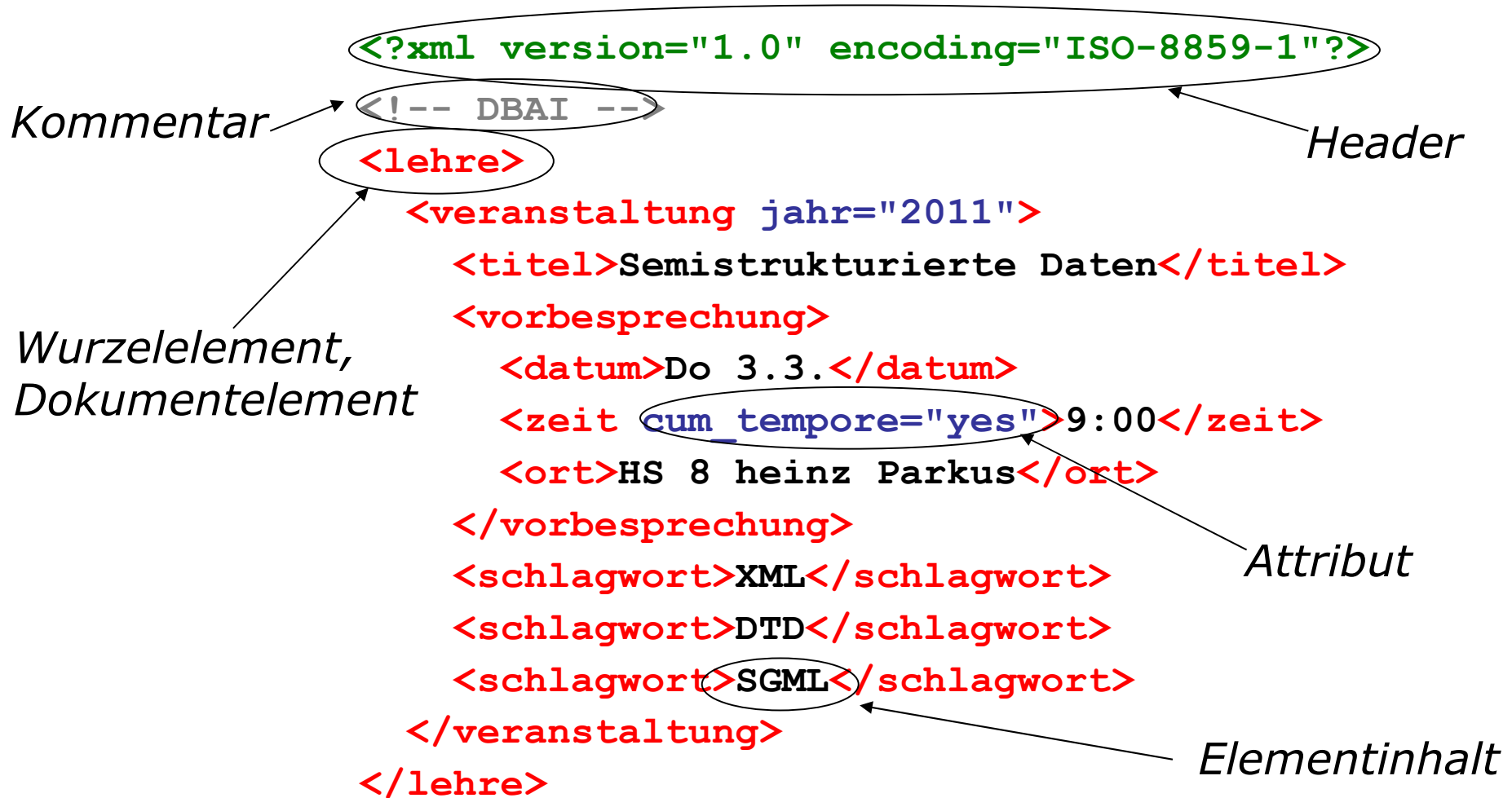
1.3. Aufbau von XML-Dokumenten

- Baumstruktur
- Elemente
- Attribute
- Header
- Kommentare
- CDATA
- Processing Instructions
- Entitäten
- Unicode

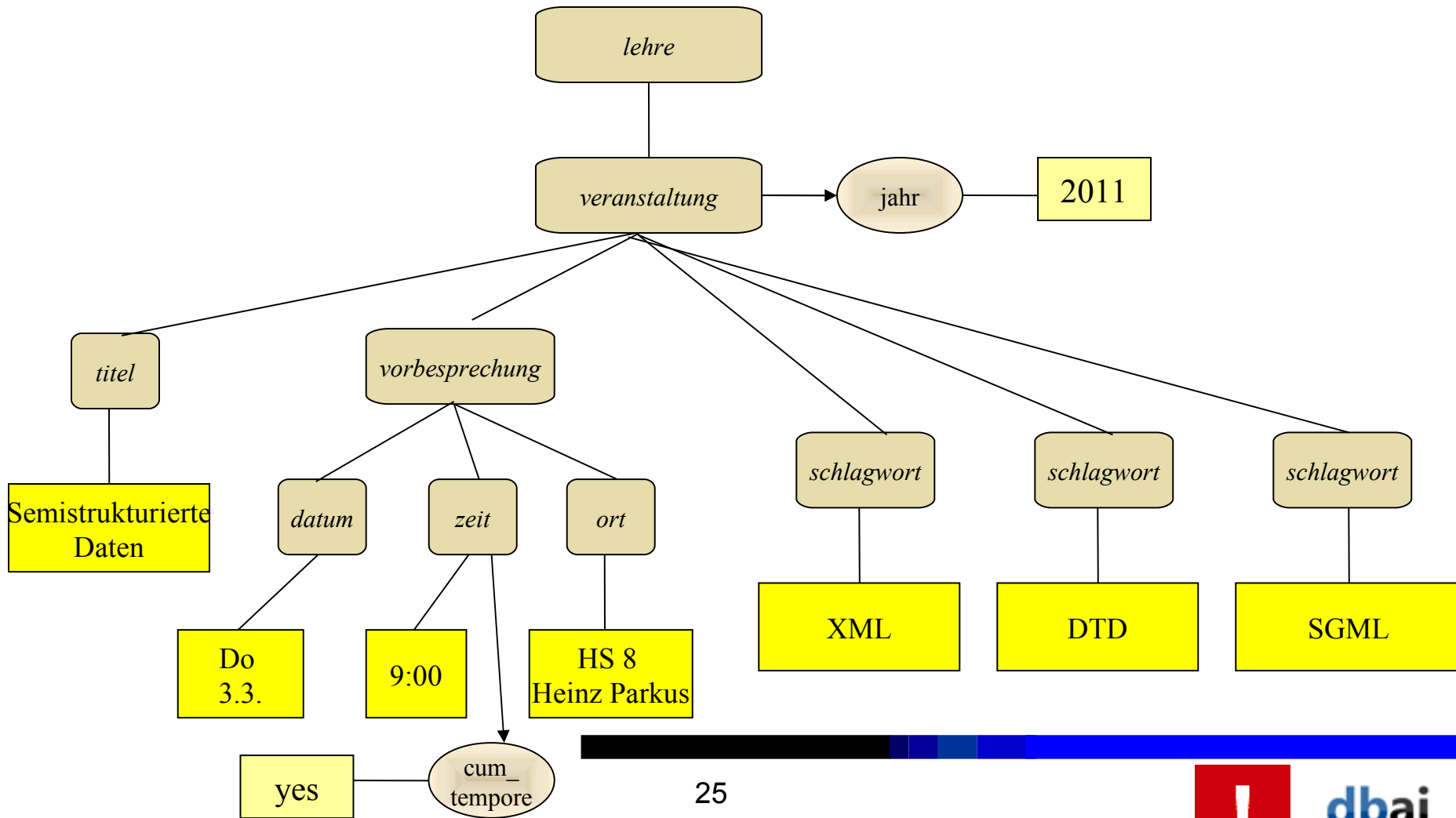
Beispiel: XML Dokument

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- DBAI -->
<lehre>
  <veranstaltung jahr="2011">
    <titel>Semistrukturierte Daten</titel>
    <vorbesprechung>
      <datum>Do 3.3.</datum>
      <zeit cum_tempore="yes">9:00</zeit>
      <ort>HS 8 Heinz Parkus</ort>
    </vorbesprechung>
    <schlagwort>XML</schlagwort>
    <schlagwort>DTD</schlagwort>
    <schlagwort>SGML</schlagwort>
  </veranstaltung>
</lehre>
```

Beispiel: Bestandteile eines XML-Dokuments



Beispiel: Dokumentenbaum



Struktur eines XML-Dokuments

■ Baumstruktur

- ☐ keine Einschränkungen der Baumstruktur durch den XML-Standard (aber natürlich mittels Schema möglich)
- ☐ ist selbstbeschreibend
- ☐ Die Ordnung der Elemente ist signifikant

■ Unterscheidung Zeichendaten vs. Markup

- ☐ Zeichendaten beherbergen die Information
- ☐ Markup beherbergt die Struktur
- ☐ beide sind einfach als Text abgelegt
- ☐ Markup steht innerhalb spitzer Klammern: `<.....>`
- ☐ Beispiel: `<lehre>`

Elemente

- beherbergen die strukturelle Information
- annotieren Text mit Markup Namen
- Elementname innerhalb spitzer Klammern

`<datum>Do 3.3.</datum>`

Starttag *Endtag*

- Gemischter Inhalt ist erlaubt, d.h.:
ein Element kann Text oder Elemente oder beides enthalten

`<datum><tag>Do</tag>3.3.</datum>`

Elementverschachtelung

- Element-Inhalt nicht auf Text beschränkt
- kann andere Elemente enthalten
- beliebige Baumtiefe und Wiederholungen

```
<ebayitem>
  <item>...</item>
  <price>...</price>
  <description>...</description>
  <description>...</description>
</ebayitem>
<yahooitem>
  <item>...</item>
  <price>...</price>
  <description>...</description>
</yahooitem>
<item>
  <item>...</item>
  <price>...</price>
  <description>...</description>
</item>
```

Regeln für Elementnamen

- XML selbst definiert keine Elementnamen
- Start mit Buchstaben oder Unterstrich
- Enthält Buchstaben, Ziffern, Unterstrich, Punkt, Bindestrich
- Buchstaben können aus beliebiger Sprache stammen (Unicode)
- Doppelpunkt erlaubt, hat aber spezielle Bedeutung
- Keine Zwischenräume erlaubt!
- Nicht beginnen mit XML (beliebig groß/klein)
- Case Sensitivity (ungleich HTML)
- Abkürzung für leere Elemente

<geblockt/> anstatt **<geblockt></geblockt>**

Einige Beschränkungen

- Genau ein Wurzelement pro XML Dokument
 - Name nicht vorgegeben
 - Bei Vereinigungen von Dokumenten: neuen Wurzelknoten hinzufügen
 - auf Ebene von Wurzelement sonst nur Kommentare und Processing Instructions erlaubt
- Start- und Endtags
 - jedes Starttag muss geschlossen werden
Beispiel: `<br>` alleine nicht erlaubt (lautet in XHTML `<br/>`)
 - keine verschränkten Tags, z.B.:
 - erlaubt: `<b>bold<i>bold-italic</i>bold</b>`
 - nicht erlaubt: `<b>bold<i>bold-italic</b>italic</i>`

Attribute

- Beschreiben üblicherweise Elementcharakteristika
- Schreibweise:
 - `<attributname> = <attributwert>`
 - Attributwert ist eine Zeichenkette in " " oder ' '.
- Ein Element kann 0 bis beliebig viele Attribute haben.
- Jeder Attributname darf nur ein Mal pro Element vorkommen

Beispiel: `<preis waehrung="$" waehrung="€">` nicht erlaubt

Attribute

- Die Reihenfolge der Attribute ist nicht signifikant
 - im Gegensatz zu Elementen
- Attribute können keine Kinder haben
 - nur Textwerte
- Datentyp von Attributen kann mittels Schema-Definition (z.B. DTD, XML Schema) beschränkt werden.
- Nicht alle Zeichen sind in Attributwerten erlaubt:
 - z.B.: "<"
 - Entitäten verwenden
 - oder (binäre) Daten codieren

Attribute

- Namensbeschränkungen wie bei Elementen
- Attribute stehen innerhalb des Starttags
- Attributwert immer in Anführungszeichen
 - können einfach ' ' oder doppelt " " sein
 - ' und " dürfen in einem Attribut nicht gemischt werden
 - jeweils andere Anführungszeichen innerhalb verwendbar
 - Attributwerte können leer sein: ""
- Beispiel: `<zeit cum_tempore="yes">`
- vordefinierte Attribute:
 - `xml:space` Whitespace erhalten oder löschen (preserve,default)
 - `xml:lang` zur Sprachspezifikation von Inhalt, z.B. "en-GB"
 - bei jedem Element möglich
 - Attributwert wird an Kindelemente vererbt

Richtlinien

■ Element vs. Attribut

- ☐ als Attribut eher systeminterne Dinge; sagt etwas darüber, wie der Elementinhalt zu interpretieren ist.
- ☐ als Element eher Dinge für den Benutzer und wo Unterstruktur nötig ist oder Daten über viele Zeilen

■ Einrückungen

- ☐ bequemere Lesbarkeit
- ☐ Whitespaces werden standardmäßig gefiltert

■ XML ungleich Semantik

- ☐ Bezeichnungen bedeuten nur für Menschen etwas
- ☐ Semantik nur durch Applikation, d.h.: die Tags haben für die jeweilige Applikation eine klar definierte Semantik, z.B. Tag `<element>` in XSD, in XSLT, in CML

Whitespaces

- Whitespaces (Leerzeichen, Tabulator, Zeilenumbruch) treten in zwei Rollen auf:
 - Formatierung des Files, Einrückungen
 - ◆ werden von Parsern ignoriert oder verschmolzen
 - als Elementinhalt
 - ◆ führende und endende Whitespaces ignorieren?
 - ◆ aufeinanderfolgende Whitespaces verschmelzen?
(je nach Parser; vordefiniertes Attribut `xml:space`)
- Folge: Repräsentation von
`<lehre><titel>SSD</titel> <nummer id="1811"/></lehre>`
hängt von diversen Faktoren ab:
 - ◆ hat Element `lehre` gemischten Inhalt?
 - ◆ ist der Parser validierend?
- Wichtig: Damit ein Element `<nummer id="1811"></nummer>` leer ist, muss der Endtag unmittelbar nach dem Starttag kommen.

Header und Kommentare

■ XML Deklaration

- ☐ erste Zeile `<?xml version="1.0"?>`
- ☐ ist optional
- ☐ weitere Information wie z.B. encoding
`<?xml version="1.0" encoding="ISO-8859-1"?>`

■ Kommentare

- ☐ `<!-- -->`
- ☐ für Menschen
- ☐ wird vom Parser nicht interpretiert (und möglicherweise ignoriert)
- ☐ entweder vor oder nach Markup
- ☐ Metacharacters (wie "<", "&") sind in Kommentaren erlaubt

CDATA und "verbotene" Zeichen

■ CDATA

- ☐ Character Data
- ☐ nicht vom Parser bearbeitet
- ☐ Entitäten und Tags innerhalb nicht erkannt
- ☐ beginnt mit `<![CDATA[`
- ☐ endet mit `]]>` (dieses Markup ignoriert Parser nicht)

■ Darstellung von beliebigen binären Zeichen

- ☐ als externe Entität (siehe 2. VL-Termin)
- ☐ codiert (z.B.: base64 oder hexBinary) im XML-Dokument

Processing Instructions

■ Einfügen von "Nicht-XML-Statements"

- Um Anweisungen an eine Applikation zu geben (Für diesen Zweck auf keinen Fall Kommentare "missbrauchen"!)

■ Typische Anwendung:

- Stylesheetverknüpfung:
 - Anweisung bzgl. Layout an den Browser.
 - `<?xml-stylesheet type="text/xsl" href="lva.xsl"?>`

■ Weiteres Beispiel:

- `<?editor href="editor" load doc?>`
- Alles hinter "editor" ist ein großer Datenblock; manche Parser versuchen, wenn es geht, diesen Teil wie Attribute zu behandeln.

Entitäten

- Deklaration von Entitäten
 - in DTD (siehe 2. VL-Termin)
 - vordefinierte Entitäten
- Entitätsreferenzen
 - innerhalb von & ... ;
 - Beispiel: `<tag>&mi;</tag>` für Mittwoch
- vordefinierte Entitäten
 - nur für folgende 5 Zeichen: `< > & ' "`
 - Entitäten: `<`; `>`; `&`; `'`; `"`;
 - **Character-Referenzen**: `Ó` (dezimal), `ó` (hex)

Zeichensätze + Zeichenkodierungen

- Basiszeichensatz für XML: **Unicode**
- ca. 100.000 Zeichen:
 - Buchstaben aller Sprachen
 - mathematische Zeichen
 - etc.
- Unicode Consortium + ISO Standard
 - <http://www.unicode.org/>
- Unicode gibt jedem Zeichen eine eindeutige Zahl
 - Unicode ist kein Font,
 - Fähigkeit Unicode zu bearbeiten ist unabhängig von Fähigkeit zur Anzeige

Zeichensätze + Zeichenkodierungen

- XML-Parser müssen zumindest folgende Kodierungen unterstützen:
 - UTF-8 (Unicode Transformation Format – 8)
Standard 7-bit ASCII als 8 Bits,
andere Zeichen mehrere Bytes lang
 - UTF-16
16 Bits Kodierung für alle "gebräuchlichen" Unicode-Zeichen,
andere Zeichen 4 Bytes lang
- Jede andere Kodierung als UTF braucht eine spezielle Deklaration im XML-Dokument
- Andere Zeichensätze + Kodierungen (Unicode-Teilmengen), z.B.: Latin-1 (ISO 8859-1):
 - Benötigt 8 bits
 - Umfasst 7bit-ASCII + Sonderzeichen für Deutsch, Französisch, ...