181135 VU Semistrukturierte Daten 1

Semistrukturierte Datenmodelle und XML

6.10.2005

Reinhard Pichler

Inhalt

- Struktur von Dokumenten
- Semistrukturierte Daten
- Entwicklung von XML
- Aufbau von XML

- Definition:

  Die Struktur eines Gegenstandes umfasst
  - die Elemente aus denen er besteht und
  - die Art und Weise, wie diese Elemente zusammenhängen

- Struktur ist ein allgegenwärtiger Begriff:

  Wirtschaftsstruktur, Organisationsstruktur, Persönlichkeitsstruktur, Sprachstruktur, Datenstruktur, Dokumentenstruktur, etc.

- Dokumentenbezogene Strukturbegriffe:

  Layoutstruktur,
  (konkrete) Inhaltsstruktur,
  Verweisstruktur,
  abstrakte Inhaltsstruktur

Layoutstruktur (visuelle Struktur)

- Struktur:
  - Visuelle Einheiten: Absätze, Tabellen, Abbildungen
  - Schrift: Schriftfamilien, Schriftgrade, Auszeichnung
- Layout (Formatierung, Typographie):
  - Layout ist immer „strukturell“
  - Layoutstruktur ist wiederholbar
    (z.B. Formatvorlagen, Stylesheets)
- Dokumentenaustausch zwischen Systemen:
  - Bewahrung des Layouts
  - HTML vs. XML

(Konkrete) Inhaltsstruktur

- Logische Beziehungen:
  - Verknüpfung von elementare Aussagen
  - Mittels logischer Operatoren (und, oder, $\rightarrow$, $\leftrightarrow$)
- Sachliche (z.B. pragmatische) Beziehungen:
  - Handlungsanweisungen
  - Handlungsoptionen
- Konventionelle Beziehungen:
  - Anrede und Gruß in einem Brief
  - Einleitung - Hauptteil - Schluss
- Layout vs. Inhalt:
  - nicht 100%ig zu trennen

Verweisstruktur

- Verweise durchkreuzen die Linearität von Dokumenten
- Mögliche Funktionen von Verweisen:
  - Orientierung (Verzeichnisse)
  - das Auffinden erleichtern (Stichwörter)
  - Redundanzverringerung (n:1-Relationen)
  - schaffen Bezüge (=> besseres Verständnis)
  - Modularisierung
- Verweise vs. Layout vs. Inhalt:
  - Verweise haben semantische Aspekte
  - und darstellungstechnische Aspekte

Abstrakte Inhaltsstruktur

Inhaltstypen (Strukturelemente, abstrakte Sinneinheiten)
und ihre Beziehungen (Ordnung)

- 3 Sichtweisen eines Dokuments:
  - Konkreter Inhalt: Daten, Information
  - Layout des Inhalts
  - **Abstrakte Inhaltsstruktur**: kommentiert, gruppiert die Inhalte, verleiht ihnen eine sinnvolle Ordnung
- Sprachgebrauch: (logische, formale) Struktur
- Freiheitsgrade des Strukturierens:
  - Strukturierungstiefe (fein/grob)
  - Strukturierungsbreite (Aspekte berücksichtigen/weglassen)
- Dokumenttyp: Dokumente mit identischer/ähnlicher Inhaltsstruktur

- Dokumentenwelt
  - Intra- und Interdokumentstruktur
  - Präsentationsformate wie HTML
  - Informationsaustauschformate
  - Document/Information Retrieval
- Datenbankwelt
  - Speichertechniken, Abfragesprachen
  - Datenmodelle, Methoden zur Strukturierung von Daten
  - Integrität/Konsistenz von Daten
- Zusammenwachsen der zwei Welten:
  - Dokumentenwelt: XML
  - Datenbankwelt: semistrukturierte Datenmodelle

Semistrukturierte Daten

- Die Suche nach Modellen für flexible und unregelmäßige Datenstruktur führte zu Modellen für semistrukturierte Daten
  - wenn kein festes bekanntes explizit gegebenes Schema
  - wenn Datenbanken mit vielen Nullwerten
  - wenn große Datenbankschemata
  - wenn Daten nicht sehr typisiert
    (i.e. können von verschiedenem Typ sein)

- Modellierung als Graph

Selbstbeschreibende Daten

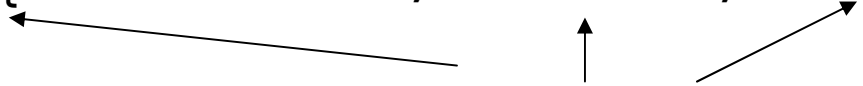
- In Datenbanken wird zuerst die Struktur beschrieben: Ein Schema und erlaubte Typen werden definiert

- Semistrukturierte Daten (SSD) oft als selbstbeschreibend bezeichnet
  - keine eigene Beschreibung der vorgeschriebenen Struktur
  - keine Beschreibung der vorgeschriebenen Typen
  - Darstellung als Label - Value Paare

- Wir verwenden hier die allgemein verbreitete SSD Syntax
  - basierend insbesondere auf Lore (Stanford, 1997)

SSD Beispiel

```
{name: "Gottlob", tel: 18420, email: "gottlob@dbai.tuwien.ac.at"}
```



Menge von Paaren mit Bezeichnungen (Label) und Werten.

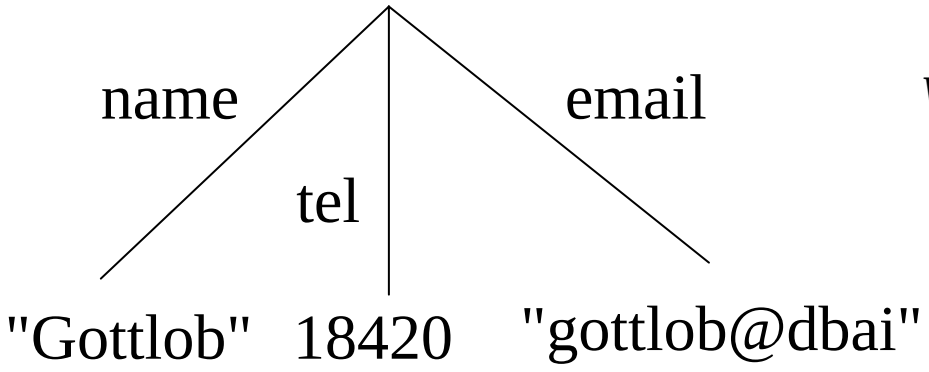
```
{name: {first: "Georg", last: "Gottlob"}, tel: 18420, email:
   "gottlob@dbai.tuwien.ac.at"}
```

Werte selbst können weitere Struktur beherbergen.

```
{name: {first: "Georg", last: "Gottlob"}, tel: 18420, tel: 18403,
   email: gottlob@dbai.tuwien.ac.at}
```

Wir können auch nicht-eindeutige Bezeichnungen erlauben.

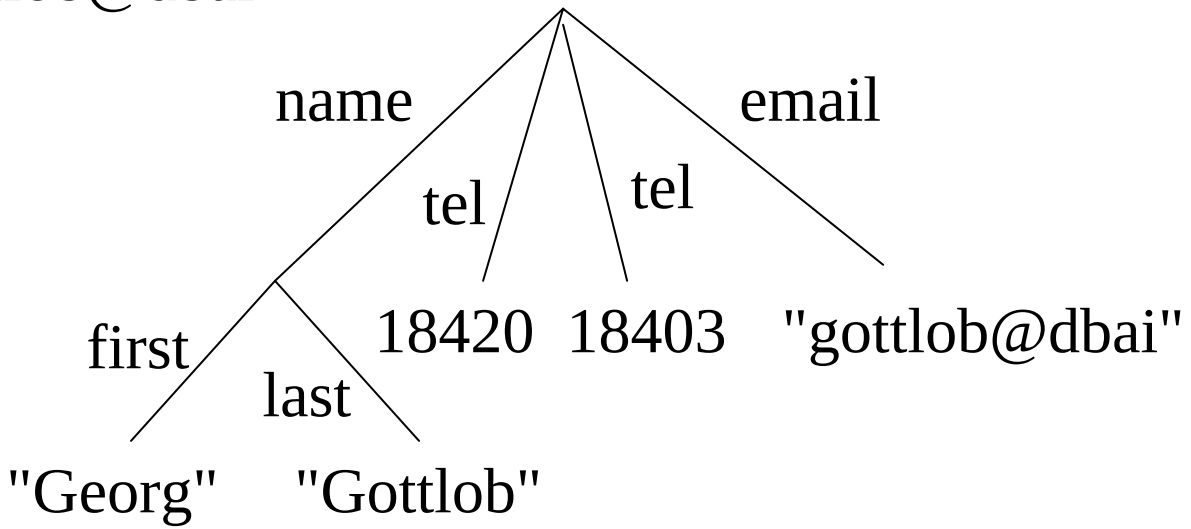
Graphische Darstellung



Label als Bezeichnung der Kanten

Werte als Bezeichnung von Blattknoten

Darstellung als Baum, generell als Graph



Beispiel: Personenliste

```
{person:
{name: {first: "Georg", last: "Gottlob"}, tel: 18420,
   tel: 18403, email: "gottlob@dbai.tuwien.ac.at"},
person:
{name: {first: "Jürgen", last: "Dorn"}, tel: 18431,
   web: "/staff/dorn/"},
person:
{name: "Slany", email: "wsi@dbai.tuwien.ac.at", tel:
   "refer to webpage", web: "/staff/slany/"}
}
```

Nicht alle Personen haben dasselbe Format. Telefonnummer hat Zahlenwert oder Stringwert. Einige Werte tauchen nur bei einigen Personen auf.

Datenstruktur

- Ein Typenschema für solch halbstrukturierte Daten könnte festgelegt werden, ist aber sehr instabil bei kleinen Änderungen
- Lösung: anstelle eines Typenschemas wird jeder Eintrag explizit mit seiner Beschreibung annotiert
- Vorteile: Interoperabilität, Erweiterbarkeit
- Nachteil: Speicherplatzverschwendung in der Standardspeicherungsart (Wiederholung der Bezeichnungen)
- Wir können übliche DB Formate in diesem Format darstellen

Darstellung relationaler Datenbanken

Beispiel: Datenschema durch zwei Relationen beschrieben. r(a,b,c), s(c,d)

r,s ... Namen der Relationen
a,b,c,d ... Spaltennamen (Variablen);
Typen müssen festgelegt werden

a	b	c
a1	b1	c1
a2	b2	c2

c	d
c2	d2
c3	d3
c4	d4

Darstellung in der ssd-Notation:

```
{r: {zeile: {a: a1, b: b1, c: c1},
     zeile: {a: a2, b: b2, c: c2}
   },
{s: {zeile: {c: c2, d: d2},
     zeile: {c: c3, d: d3},
     zeile: {c: c4, d: d4}
   }
}
```

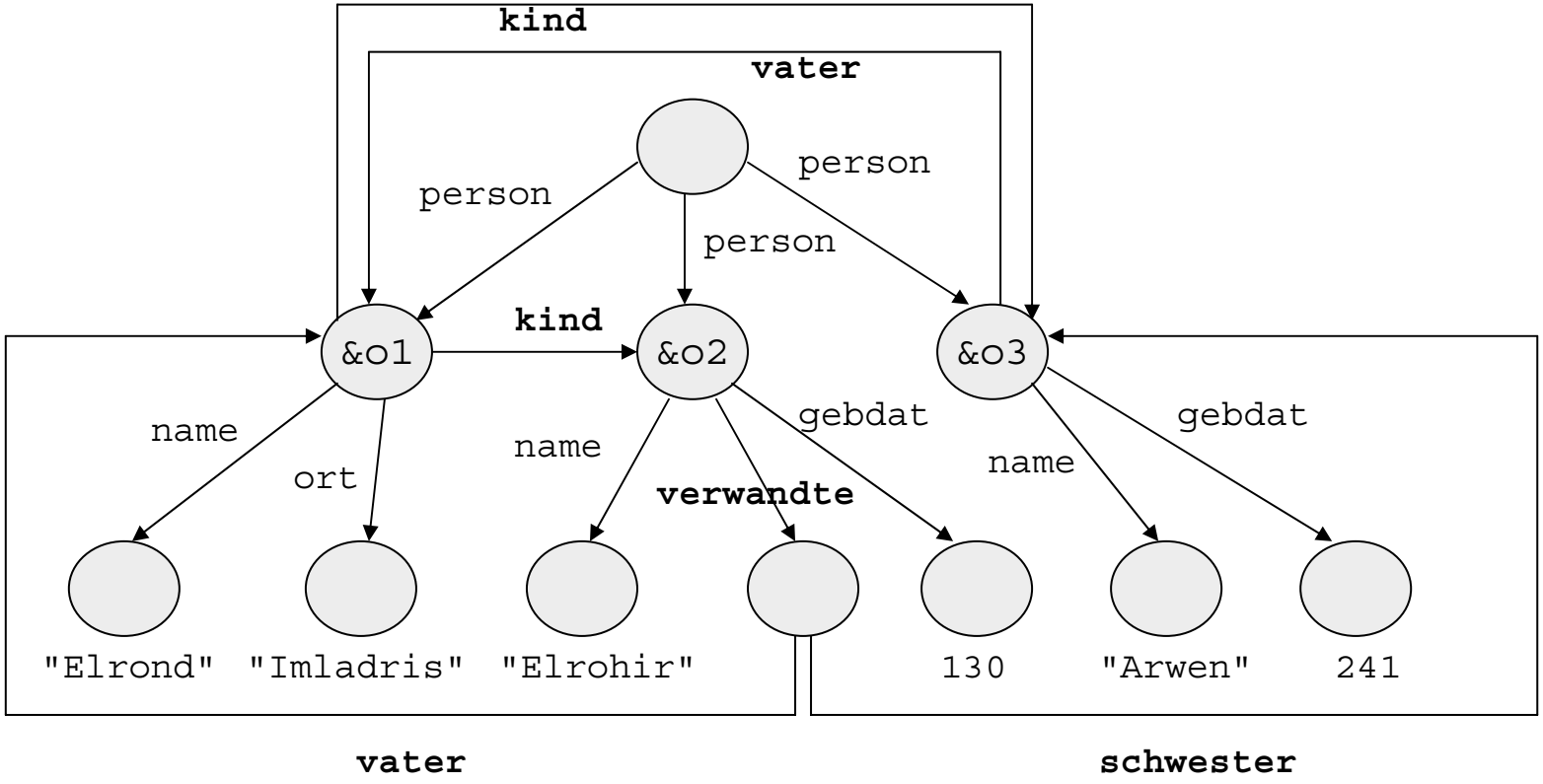
Verschiedene Darstellungsmöglichkeiten:

z.B. `spalten` statt `zeilen`
z.B. `r,s` ganz weglassen

Darstellung zyklischer Strukturen (OO-DB)

Modellerweiterung:

- Zuweisung von **Objektidentitäten** zur Referenzierung durch andere Objekte (als Knotenbezeichnung)
- **generelle Graphstruktur** mit gerichteten Kanten



Darstellung zyklischer Strukturen (OO-DB)

```
{
 person: &o1{name: "Elrond", ort: "Imladris", kind:
   &o2, kind: &o3},
 person: &o2{name: "Elrohir", gebdat: 130, verwandte
   {vater: &o1, schwester: &o3}},
 person: &o3{name: "Arwen", gebdat: 241, vater: &o1}
}
```

Anstelle eines Baumes nun Graphen, in diesem Fall sogar zyklischer Graph. Substrukturen werden miteinander geteilt.

&oI ... **Objektidentitäten** (oids). Jeder Knoten kann eine eindeutige oid haben.

oids: physikalisch Pointer auf bestimmte Speicherstelle, URI, ...

SSD und ODMG

- Objektorientiertes Datenmodell ODMG (Cattell, 1994)
- ODMG stellt Mengen, Listen, etc. zur Verfügung
- Handling von Datenbankobjekten
  - Verwendung von extent
  - z.B. cities ist Collection aller Instanzen von City, in dem Fall set<City>
- Objekte haben Attribute und Beziehungen.
  - Beziehungen sind wie Eigenschaften für Abfragen
  - erlauben inverse Einschränkungen (inverse Keyword)

```
class State (extent states)
{
  attribute string scode;
  attribute string sname;
  attribute City capital;
  relationship set<City> cities-in
      inverse City::state-of;
}
```

```
class City (extent cities)
{
  attribute string ccode;
  attribute string cname;
  relationship State state-of
      inverse State::cities-in;
}
```

SSD und ODMG

states-city Beispiel mit Beispieldaten:

```
{states:
  {state: &s1{scode: "ID", sname: "Idaho", capital: &c1,
   cities-in {City: &c1, City: &c3, .....},
  {state: &s2{scode: "NE", sname: "Nevada", capital: &c2,
   cities-in: {City: &c2, ...}
  .....
  }
}
{cities:
  {city: &c1{ccode: "BOI", cname: "Boise", state-of: &s1},
   city: &c2{ccode: "CCN", cname: "Carson City", state-of: &s2},
   city: &c3{ccode: "MOC", cname: "Moscow", state-of: &s1},
   .....
   }
}
```

In ssd keine direkte Aussage über inverse Constraints und Typen (das wären eigene weitere Einschränkungen).

SSD Syntax

- Bisher als Typen: Zahlen, Strings, oids
- weitere Typen durch Angabe eines Typentags definierbar,
  z.B.: (date, „28. April 2003")
- Unterscheidung atomare Werte (Strings,Zahlen,...) und komplexe Werte
- ssd Ausdruck:

```
<ssd-Ausdruck>    ← <wert> | oid<wert> | oid
<wert>            ← atomarerwert | <komplexerwert>
<komplexerwert>   ← {label: <ssd-Ausdruck>, ...,
                     label: <ssd-Ausdruck>}
```

Weitere Ansätze

- OEM Object Exchange Model (1994)
  - Bezeichnungen sind den Knoten des Graphen zugewiesen
  - Nachteil: Hilfsknoten nötig für Referenzen
- F-Logik (1995)
  - Komplexes Datenmodell (als Graph darstellbar)
  - Abfragen durch Logikprogrammierung-Formalismus
- UnQL (1995)
  - Datenmodell ähnlich wie SSD (anderer Gleichheitsbegriff)
  - Abfragesprache UnQL verwendet strukturelle Rekursion (wie XSLT)
- ACeDB A *C.elegans* Database (1992)
  - sehr spezifische Anwendung, aber sehr allgemeines Datenmodell
  - flexibles Datenschema (ähnlich wie DTD)

eXtensible Markup Language

Wichtige Eigenschaften:

- Genormte, erweiterbare Auszeichnungssprache (W3C)
- Syntax zur Beschreibung (semi)strukturierter Information
- Trennung von Struktur und Präsentation
- Bereichsspezifische Dokumenttypen („Applikationen“)
- Datenaustauschformat
- Datenmodellierung

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

Was ist Markup?

- prozedurales Markup
  - diverse Zusatzanmerkungen für Publishing
  - /farbe0000FF/groesse42 Was ist Markup
  - i.a. nur optisch strukturiert
  - unflexibel und fehleranfällig
- Erweiterung: Generisches Codieren
  - Makros (z.B. T_EX)
  - \emph{Was ist Markup}
  - portabler, beschreibt schon "eher" Struktur
  - flexibel und portierbar

SGML

- Standard Generalized Markup Language
- Erweiterung von generischem Codieren
- Strukturbeschreibendes Markup
  (beschreibt die abstrakt-inhaltliche Struktur)
- 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 erzwingt keine strikte Struktur
  - viele Formatting Tags die von Browsern interpretiert werden
  - Makros: Stylesheets

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

Warum XML?

- Einschränkungen von HTML
  - Fix vorgegebene Menge von Elementen
  - keine Trennung von Layout und Struktur
  - Wiederverwendbarkeit?
- 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)

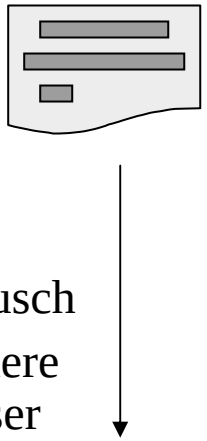
Ziele beim Design von XML

1. Im Internet leicht einsetzbar
2. Für breites Spektrum von Anwendungen
3. Mit SGML kompatibel
4. Programme für XML-Verarbeitung leicht zu entwickeln
5. (Fast) keine Optionen

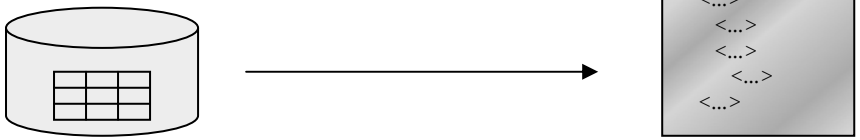
Ziele beim Design von XML

6. Für Menschen lesbar
7. Rasches Design von XML
8. Formale Definition von XML (EBNF)
9. XML-Dokumente leicht zu erstellen
10. Keine Abkürzungen beim Markup

Anwendung von XML



- Dokument-Anwendungen
  - "menschlicher" Informationsaustausch, B2B, B2C
  - reine Strukturbeschreibung generiert transportables und leicht wiederverwendbares Dokument; einfacherer Informationsaustausch
  - verschiedenste Konvertierungen durch Stylesheets; z.B. einfachere Aufrechterhaltung großer Websites, die für verschiedene Browser optimiert sind



- Daten-Anwendungen
  - einfacher "maschineller" Informationsaustausch mit derselben Technologie
  - *"application as document"*
  - automatisierter Datenaustausch mit Datenbanken und Clients
  - XML Datenbanken

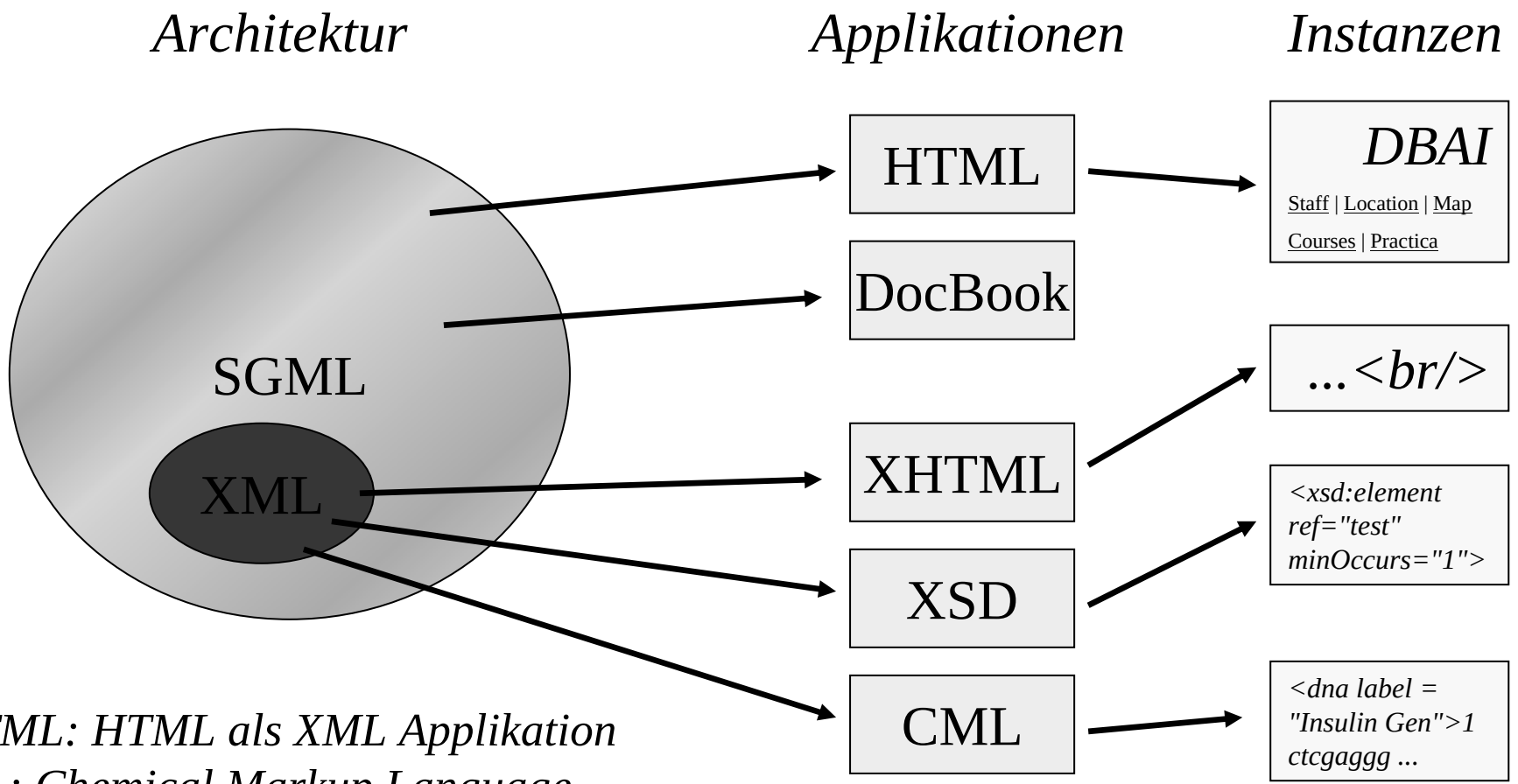
HTML vs. XML

- HTML
  - Hypertext Markup Language
  - eine Applikation von SGML (eine fixe DTD)
  - HTML 4.0, CGI, Javascript, Flash,....
  - über 100 *fixe* tags
  - Browser sehr fehlertolerant (ignoriert DTD....)
  - Präsentation (z.B. boldfaced, rot) und Struktur (z.B. Tabellen, Listen)
  - Chaos: verschiedenste proprietäre Erweiterungen
  - Semantische Information nur in Metatags
- XML
  - eXtensible Markup Language
  - eine Teilmenge von SGML
  - Metasprache für Markup Sprachen
  - keine prädefinierten Tags
  - strikte Syntax muss eingehalten werden
  - Abfragesprachen
  - viele ergänzende Standards, die in Folge vorgestellt werden
  - leicht zu lesen und zu verarbeiten

Einschränkungen von XML (vs. SGML)

- Max. 2 Bestandteile eines XML-Dokuments:
  - Document Instance, 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] [Ll] beginen
- Keine Auslassung/Abkürzung von Markup
- Angabe von Attributname + Attributwert zwingend
- Einige Deklarationen nicht möglich
  - z.B.: SHORTREF, USEMAP, LINKTYPE, USELINK, LINK
- Immer Unicode als Basiszeichensatz
- Keine Kapazitätsbeschränkungen

Applikationen und Instanzen



XHTML: HTML als XML Applikation
CML: Chemical Markup Language
XSD: XML Schema Description (siehe später)

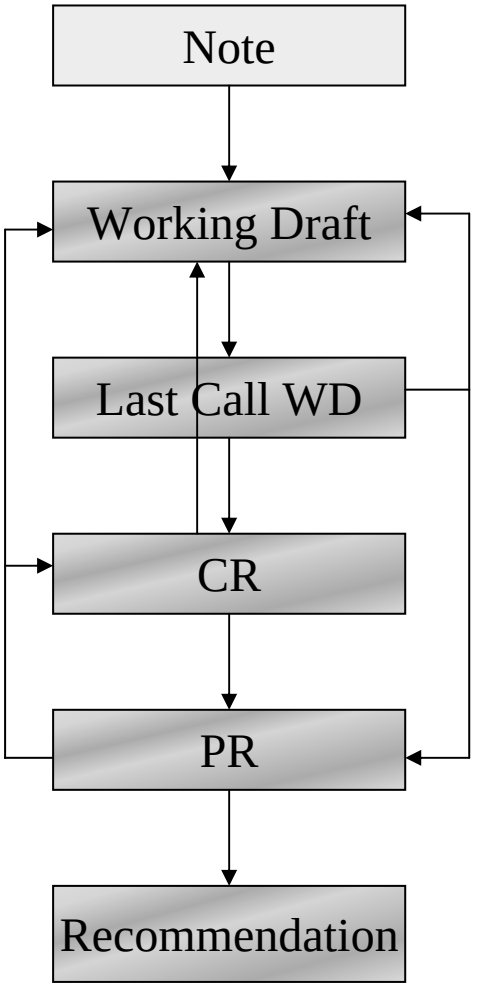
Geschichte

- Hypertext (1945)
  - beliebige Navigationspfade durch Dokumente
- GML (SGML Vorläufer) (1969)
- SGML ISO Standard (1986)
- HTML Tim Berners-Lee, CERN (1989)
  - Ziel: einfacher Dokumentaustausch
- W3C gegründet (1994)
- SGML Subset Arbeitsgruppe (1996)
- XML 1.0 (1998)
- XSLT (1999)
- XML 2nd Edition (2000)
  - kleine Änderungen und beinhaltet Namespaces
- XML Schema (2001)
- XQuery (mittlerweile > 10 working drafts)

W3C

- World Wide Web Consortium
  - gegründet von CERN, MIT und anderen
  - ca. 200 Mitglieder (Firmen)
  - Aufgabe: Standardisierung von Webformaten
  - nicht normativ: gibt nur "recommendations" heraus, keine ISO artigen Standards

- Sechs Arten von Dokumenten
  - Note
    - ist nicht Bestandteil des Standardisierungsprozesses
    - keine Absichtserklärung des W3C steht dahinter
  - Working Draft (WD)
    - dokumentieren aktuellen Diskussionsstand
  - Last Call WD
    - wenn festgelegte Ziele erreicht
  - Candidate Recommendation (CR)
    - Einreichungsbestätigung
  - Proposed Recommendation
    - Erweiterung; Implementierungen der Bestandteile
  - Recommendation
    - offizieller W3C Standard



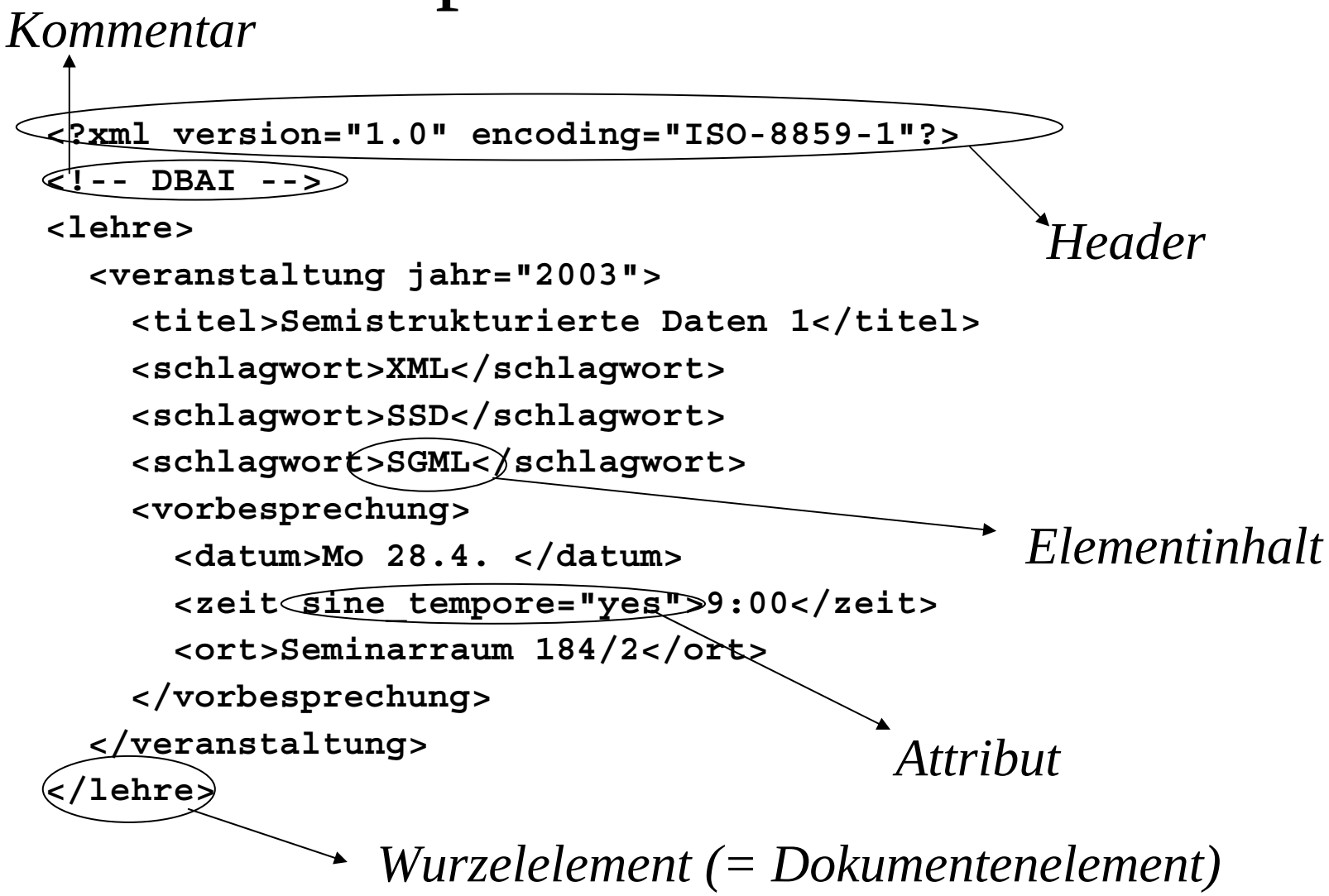
Rund um XML: Companion Standards

- XML selbst "tut nichts"
  - das "Rundherum" macht es aus
- Namespaces
- DTD, XSD
  - Datenmodellierung für Markup Sprachen
- Stylesheet Transformationen
  - Navigation, Transformationen, Darstellung und Layout (XPath, XSLT, XSL-FO)
- XLink, XPointer
  - Links, Relationen zu anderen Dokumenten
- XQuery
  - Abfragesprachen
- u.v.a.m.
  - DOM, Parser, APIs, RDF, etc.

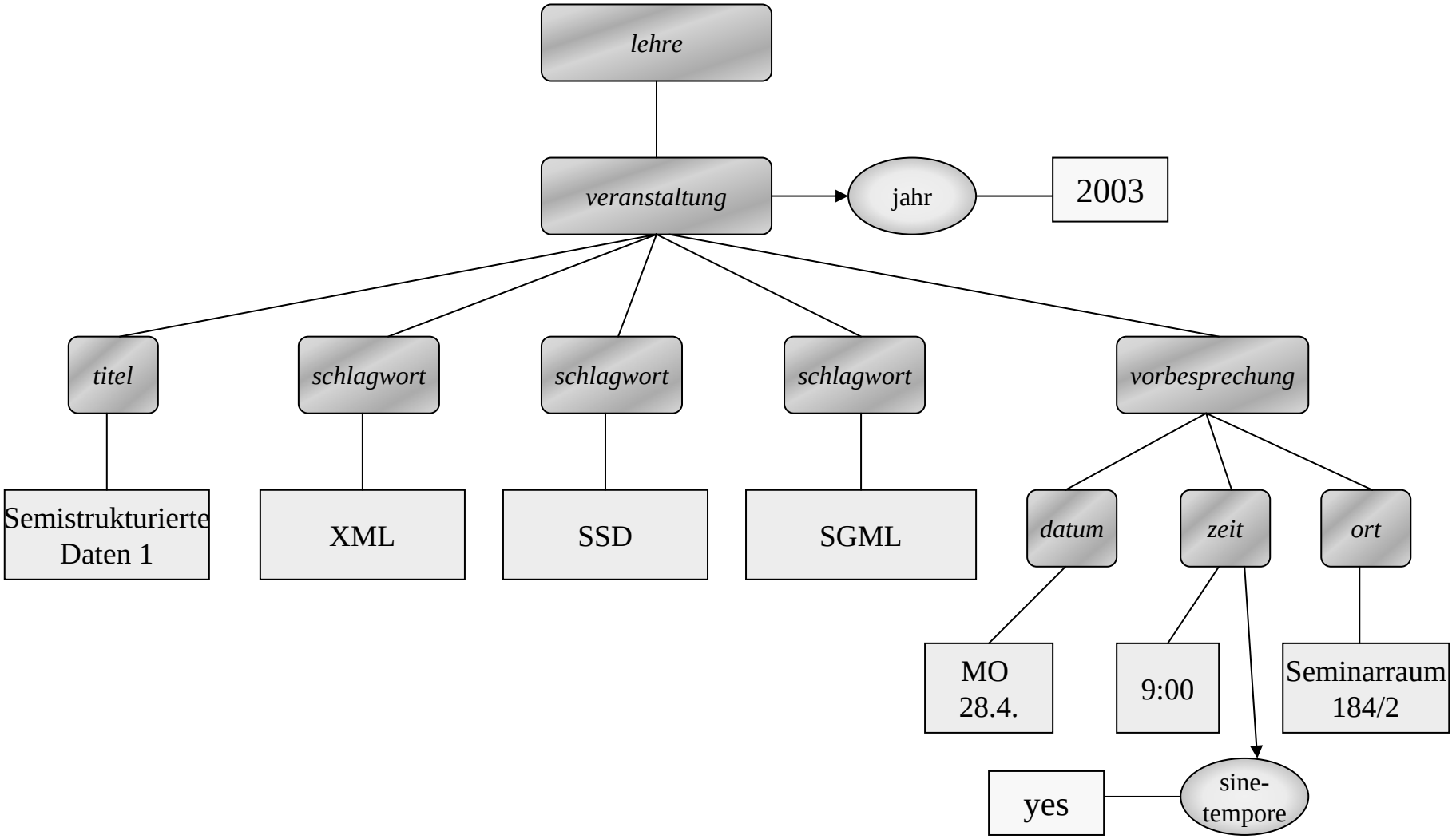
Beispiel: XML Dokument

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- DBAI -->
<lehre>
  <veranstaltung jahr="2003">
    <titel>Semistrukturierte Daten 1</titel>
    <schlagwort>XML</schlagwort>
    <schlagwort>SSD</schlagwort>
    <schlagwort>SGML</schlagwort>
    <vorbesprechung>
      <datum>Mo 28.4. </datum>
      <zeit sine_tempore="yes">9:00</zeit>
      <ort>Seminarraum 184/2</ort>
    </vorbesprechung>
  </veranstaltung>
</lehre>
```

Beispiel: Dokumentaufbau



Beispiel-Dokumentenbaum (DOM)



XML Dokumentaufbau

- „Objekte“ bilden baumartige Struktur
  - Baumstruktur ist simpel und flexibel
  - keine fixe Vorgaben für Anordnung der Baumelemente (vgl. Datenbanken: zuerst Typendefinition), daher *semistrukturiert*
  - Das Dokument selbst hat eine innere Struktur (im Gegensatz zu "plain text "), gut maschinell lesbar
  - ist *selbstbeschreibend*
  - Ungleich ssd ist die Ordnung der Elemente relevant
- Unterscheidung Character Data vs. Markup
  - Character Data beherbergt die Information
  - Markup beherbergt die Struktur
  - beide sind einfach als Text abgelegt
  - Markup findet sich innerhalb spitzer Klammern: <.....>
  - Beispiel: `<lehre>`

XML Dokumentaufbau

- Im Normalfall werden führende und endende Whitespaces ignoriert
  - je nach Parser adaptierbar (bzw. vordefiniertes Attribut)
  - daher viele Repräsentationen desselben Dokuments, kanonische Darstellung

```
<titel>Semistrukturierte
Daten 1</titel>
<nummer
id="181135"></nummer>
```

Kanonische Darstellung:

```
<titel>Semistrukturierte Daten 1</titel>
<nummer id="181135"/>
```

- Wohlgeformtheit
  - Ein XML Dokument ist wohlgeformt wenn es syntaktisch korrekt gebildet wurde (Beschränkungen s.u.)
  - später: verschiedene Arten von Gültigkeit

Unicode Unterstützung

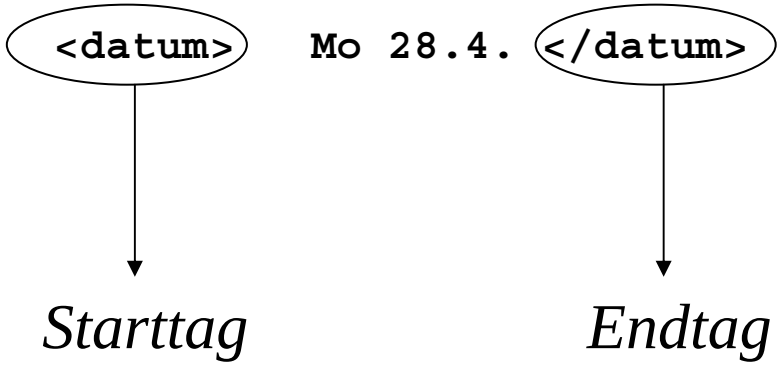
- Unicode an jeder Stelle des XML Dokuments erlaubt
- Erweiterung von ASCII
- Unicode Consortium + ISO Standard
  - http://www.unicode.org/
- ca. 65000 Zeichen (16 bits pro Character)
- Unicode gibt jedem Zeichen eine eindeutige Zahl
  - Unicode ist kein Font, Fähigkeit Unicode zu bearbeiten ist unabhängig von Fähigkeit zur Anzeige
- Alle Sprachen und mathematische Zeichen

Unicode

- Transportkodierungen:
  - Latin-1 (ISO 8859-1):
    - (= ASCII = 8 bits-Teilmenge von Unicode)
  - UTF-8 (Unicode Transformation Format – 8)
    - Standard 7-bit ASCII als 8 bits, andere Zeichen mehrere Bytes lang
  - UTF-16
    - 16 bits Codierung
- Jede andere Codierung als UTF braucht eine spezielle Deklaration im XML-Dokument

Elemente

- beherbergen die strukturelle Information
- annotieren Text mit Markup Namen
- Elementname innerhalb spitzer Klammern (wie in SGML)



- Gemischter Inhalt erlaubt
  - Ein Element kann Text oder Elemente, und sogar Text und Elemente enthalten

```
<datum><tag>Mo</tag>28.4.</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>
```

Beschränkungen

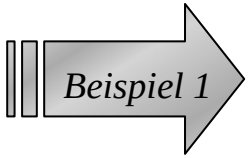
- Genau ein Wurzelelement pro XML Dokument
  - Name nicht vorgegeben
  - Bei Vereinigungen von Dokumenten: neuen Wurzelknoten hinzufügen
  - auf Ebene von Wurzelelement sonst nur Kommentare/PIs erlaubt

- Endtags
  - jedes Starttag muß geschlossen werden
    Beispiel: **`<br>`** alleine nicht erlaubt (muß in XHTML **`<br/>`** lauten)
  - keine verschränkten Tags,
    eindeutige Kinder/Geschwisteridentifikation
    Beispiel: **`<b>bold<i>bold-italic</b>italic</i>`** nicht erlaubt!
    Aber erlaubt: **`<b>bold<i>bold-italic</i>bold</b>`**

Regeln für Elementnamen

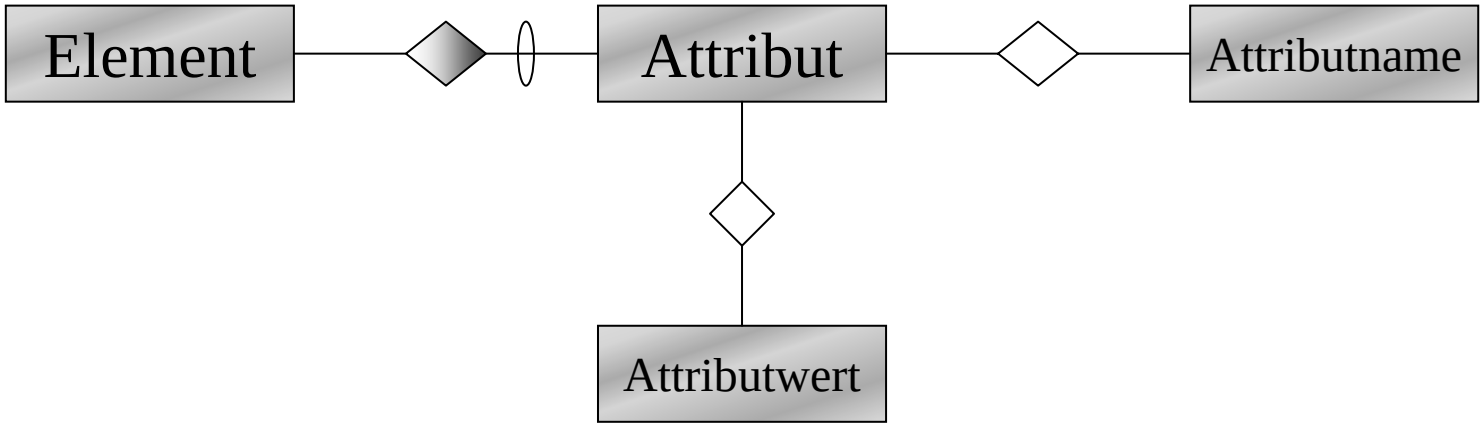
- XML selbst definiert keine Elementnamen
- Start mit Buchstaben oder Unterstrich
- Enthält Buchstaben, Ziffern, Unterstrich, Punkt, Bindestrich
- Doppelpunkt erlaubt, hat aber spezielle Zwecke
- beliebige Sprache da Unicode
- Keine Zwischenräume erlaubt!
- Nicht beginnen mit [Xx] [Mm] [Ll] (reserviert)
- Case Sensitivity (ungleich HTML)
- Abkürzung für leere Elemente

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



Attribute

- beschreiben Elementcharakteristika
- Attributname, Attributwert



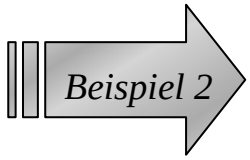
- jeder Attributname darf nur einmal pro Element vorkommen
  Beispiel: `<preis waehrung="$" waehrung="€">` nicht erlaubt

Attribute

- Die Reihenfolge der Attribute ist ungeordnet
  - im Gegensatz zu Elementen, bei denen die Ordnung eine Rolle spielt
- Attribute können keine Kinder haben
  - nur Textwerte
- Ein Attribut kann keine Liste als Wert haben
  - nur „manuell“ möglich: **`attr1="x" attr2="y" attr3="x"`**
  - aber ein Attributwert kann ein Listenwert sein: **`attr="x y z"`**
- Attribute können in DTD beschränkt werden, mehr als Elemente
  - typisiert sind Attribute also nur mit DTD (siehe 2. VL-Termin)
  - können dann auch für Objektreferenzen verwendet werden
- Nicht alle Zeichen sind in Attributwerten erlaubt:
  - Entitäten verwenden (siehe 2. VL-Termin)
  - oder (binäre) Daten codieren

Attribute

- Namensbeschränkungen wie bei Elementen
- Attribute stehen innerhalb des Starttags
- Attributwert immer in Anführungszeichen
  - können single oder double sein
  - nicht mischen in einem Attribut
  - jeweils andere Anführungszeichen innerhalb verwendbar
  - Attributwerte können leer sein
- Beispiel: **`<zeit sine-tempore="yes">`**
- vordefinierte Attribute:
  - **`xml:space`** wenn spaces erhalten bleiben sollen (preserve,default)
  - **`xml:lang`** zur Sprachspezifikation von Inhalt, z.B. "en-GB"
  - bei jedem Element möglich
  - Wert wird von Subelementen geerbt (oder überschrieben)



Richtlinien

- Element vs. Attribut
  - als Attribut eher systeminterne Dinge, insbesondere dann wenn kleine Auswahlliste gewünscht ist, die für Attribute in DTD angegeben werden kann
  - als Element eher Dinge für den Benutzer und wo Unterstruktur nötig ist oder Daten über viele Zeilen
- Indentierung
  - bequemere Lesbarkeit
  - Whitespaces werden standardmäßig gefiltert
- XML ungleich Semantik
  - Bezeichnungen bedeuten nur für Menschen etwas
  - Semantik nur durch Applikation, z.B. durch XSL (Präsent.sem.), RDF (Relationship zwischen Dokumenten)

Header und Comments

- XML Deklaration
  - erste Zeile **`<?xml version="1.0"?>`**
  - ist optional
  - ist eine Processing Instruction (s.u.)
  - weitere Information wie z.B. encoding

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

- XML Comments
  - <!-- -->
  - für Menschen, wird vom Parser ignoriert, für den Objektbaum uninteressant
  - entweder vor oder nach Markup
  - Metacharacters sind in Comments erlaubt

PCDATA vs CDATA

- CDATA
  - Character Data
  - nicht vom Parser bearbeitet
  - Entitäten und Tags innerhalb nicht erkannt
  - beginnt mit **`<![CDATA[`**
  - endet mit **`]]>`** (dieses Markup ignoriert Parser nicht)
- PCDATA
  - parsable character data
  - wird vom Parser bearbeitet
  - Elementinhalt ist gewöhnlich PCDATA, da darin andere Elemente enthalten sein können
- Darstellung von beliebigen binären Zeichen
  - als externe Entität (siehe 2. VL-Termin)
  - codiert (z.B.: base64 oder hexBinary) im XML-Dokument selbst

Processing Instructions

- nicht Teil des Dokuments
- Einfügen von Nicht-XML-Statements
  - i.a. möchte man "processing from structure"
  - manchmal sind solche Dinge aber einfacher (Skripts ausführen...)
- Beispiele
  - **`<?xml ... ?>`** ist eine PI
  - **`<?xml-stylesheet ....?>`** für Stylesheetverwendung
- keine Attributwerte
  - **`<?editor href="editor" load doc?>`**
  - Alles hinter "editor" ist ein großer Datenblock; manche Parser versuchen wenn es geht wie Attribute zu behandeln

Entitäten

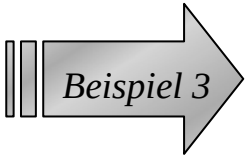
- Deklaration von Entitäten
  - in DTD (siehe 2. VL-Termin)
  - vordefinierte Entitäten
- Entitätsreferenzen
  - innerhalb von **`& ... ;`**
  - Beispiel: **`<tag>&do;</tag>`** für Donnerstag
- vordefinierte Entitäten
  - nur für folgende 5 Zeichen: **`< > & ' "`**
  - Entitäten: **`< > & &apos; &quot;`**
  - Character-Referenzen: **`&#211;`** (dezimal), **`&#xF3;`** (hex)

Präsentation von XML Dokumenten

- Stylesheet Verknüpfungen
  - CSS (Cascading Stylesheet)
  - XSL (eXtensible Stylesheet Language) (5.+ 6. VL-Termin)
- Datenbindung
  - in HTML Dokument einbinden
- Skripts
  - XML mit HTML-Seite verknüpfen
  - mit Javascript auf XML-Inhalt zugreifen

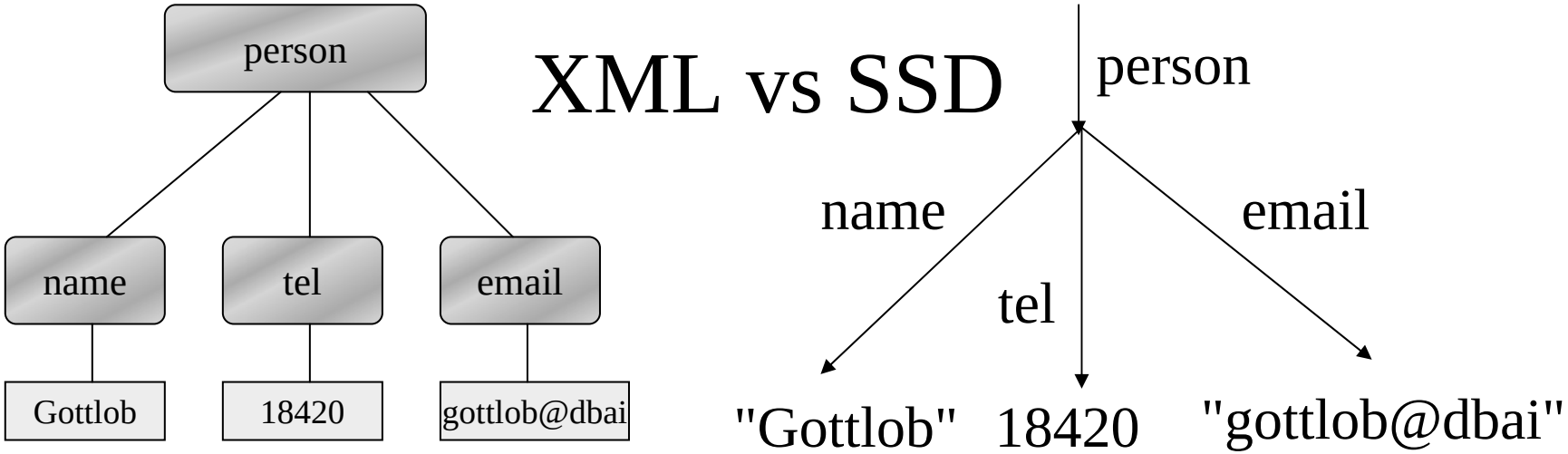
Präsentation von Elementen mit CSS

- Schriftarten-Eigenschaften:
  - font-family, font-size, font-style, font-weight
- Textabstände, Textausrichtung:
  - letter-spacing, vertical-align, text-align, text-indent, ...
- Weitere Eigenschaften:
  - display, colour, margin-left, margin-top



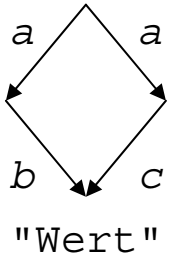
Einige XML Tools

- XML Browser
  - IE 6 (unterstützt XSLT)
  - Mozilla (Open Source), Netscape 6 (leider kein XSLT)
  - InDelv XML Browser
  - Opera 6.0
  - Amaya (W3C Browser/Editor) (auch MathML Unterstützung!, kein XSLT)
- XML Editoren Arten
  - textbasiert mit Syntaxhighlighting
  - baumbasiert
  - pseudo-wysiwyg (zB XMetaL verwendet Stylesheets zur Darstellung)
- XML Editoren
  - XMLSpy (www.xmlspy.com)
    - u.a. visuelles DTD/Schema Erstellen
  - XMetaL
  - XML Authority
  - Adobe Framemaker (SGML)
  - XML Notepad (einfach, MS)
  - Microstar (XML Modelling Tools)
- XML Parser
  - Apache Xerces
- XPath
  - XPathTester
- XSLT
  - Apache Xalan, XT, Saxon
- Datenbanken
  - Tamino, Poet, Cache, ...
  - native vs. xml-enabled



- Basis XML Syntax ideal für Beschreibung von SSD.
- Für Bäume einfache Übersetzung:
  - T(atomarerwert) = atomarerwert
  - $T(\{l_1:v_1,....,l_n:v_n\}) = <l_1>T(v_1)</l_1>....<l_n>T(v_n)</l_n>$
- XML Attribute sind Objekteigenschaften. Möglichkeit Information in Attributen (nur einmal pro tag, mehr Einschränkungen möglich) oder Elementen (kann Subelemente enthalten) zu spezifizieren.
- Baumdarstellung von XML: Tagnamen mit Knoten assoziiert. Im Fall von Bäumen leicht XML Bäume in SSD Bäume abbildbar (Bezeichnungen auf Kanten geben, neue Ursprungskante).

XML vs SSD



- XML Referenzen:
  - Identifier als spezieller Attributwert.
  - Im Fall von Graphen nicht eindeutige Übersetzung SSD nach XML.
  - Obiger Graph kann verschieden dargestellt werden
    `<a><b id="&o1">Wert</b></a> <a c="&o1"/>` (c als Referenzattribut)
    `<a b="&o1"/> <a><c id="&o1">Wert</c></a>` (b als Referenzattribut)
- Ordnung der Elemente:
  - In XML spielt die Ordnung eine Rolle, in SSD nicht.
  - In der Dokumentenwelt Reihenfolge gewünscht, in Datenbankwelt nicht
- Aber: Attribute in XML sind ungeordnet!
  - **`<person first="Georg" last="Gottlob"/>`** ist gleich zu
    `<person last="Gottlob" first="Georg"/>`
- XML Dokument hat 1 Wurzelelement, SSD hat i.a. mehrere

XML vs SSD

- XML erlaubt gemischte Elemente ("mixed elements")
  - PCDATA und Elemente dürfen gemischt werden
  - daher auch Dokumentformat, von DB-Perspektive unnatürlich
  - für Übersetzung in SSD müßte man ein Tag rundherum dazufügen

```
<info>So etwas<speziell>geht nicht</speziell> in SSD </info>
```

- Kommentare sind Teil des XML Dokumentes
- Processing Instructions und Header
- CDATA und Entitäten
- Namensräume (2.VL-Termin)
- Möglichkeiten eines Schemas (2.+ 3.VL-Termin)
  - ähnlich wie Typenschemata für semistrukturierte Daten
  - Referenzierung des Schemas im XML Dokument

Ausgewählte Literatur

- Data on the Web
  - Serge Abiteboul, Peter Buneman, Dan Suciu
- XML by Example
  - Benoit Marchal
- XML Bible
  - Eliotte Rusty Harold
- Semistructured Data Tutorial
  - Peter Buneman, PODS 1997
- Describing Semistructured Data
  - Luca Cardelli, Sigmod Record 4/2001

Ausgewählte Webseiten

- http://www.db-stanford.edu/tsimmis/tsimmis.html
  - Tsimmis Mediatorsystem
- http://www.inf.fu-berlin.de/~faulstic/bib/semistruct/
  - Über Projekte zu semistrukturierten Daten
- http://xml.darmstadt.gmd.de
  - XML Kompetenzzentrum (Fankhauser Unterlagen, XQL Engine,...)
- http://www.w3.org
  - W3C Consortium
- www.w3schools.com
  - Diverse Tutorials
- www.jeckle.de
  - Umfangreiche XML Vorträge und mehr

Übungsbeispiel A

Erstellen Sie eine XML-Homepage für die Lehrveranstaltung
„Semistrukturierte Daten 1“
und überprüfen Sie die Wohlgeformtheit in einem Editor/Browser.

Verwenden Sie folgende XML-Sprachmittel:
Leeres Element, geschachtelte Elemente, Element mit Text-Inhalt,
Element mit gemischtem Inhalt (d.h.: Text + Subelemente),
Attribute, Header, Kommentar, CDATA-Section,
Processing Instruction (für ein einfaches CSS-Stylesheet),
vordefinierte Entitäten

Übungsbeispiel B

Betrachten Sie die relationale Uni-DB (nächste Folie) mit den folgendenTabellen:

Studenten, Professoren, Assistenten, Vorlesungen, voraussetzen, prüfen, hören

Stellen Sie diese DB auf folgende Weisen dar:

- mittels ssd (und zugehörigem Graph): Beachten Sie die Fremdschlüssel-Beziehungen zwischen den Tabellen und realisieren Sie sie mittels oids!
- in XML

Die relationale Uni-DB

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Studenten		
MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen			
VorlNr	Titel	SWS	gelesen von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die Stoiker	4	2137

voraussetzen	
Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022

Assistenten			
PersNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

prüfen			
MatrNr	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2