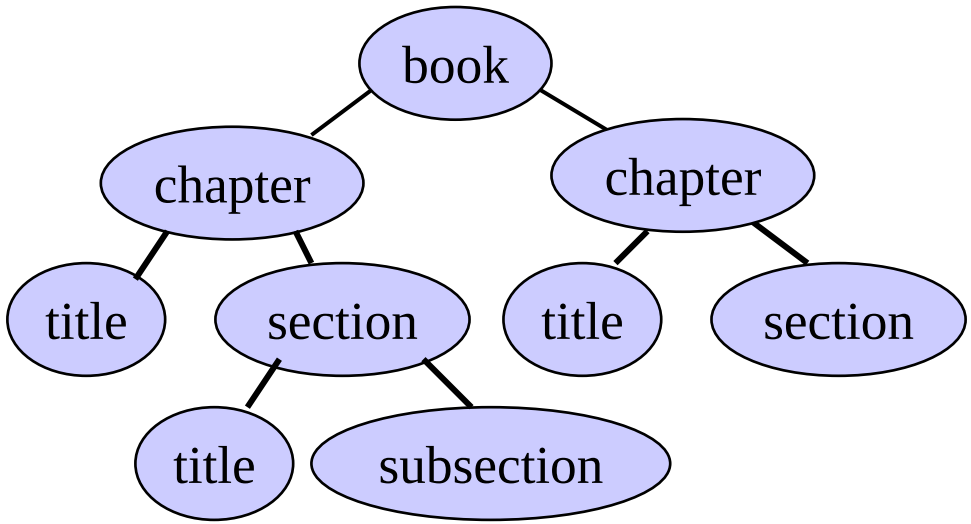
Strukturelle Anfragen

Viele Folien in diesem Abschnitt sind eine deutsche Übersetzung der Folien von Raymond J. Mooney (http://www.cs.utexas.edu/users/mooney/ir-course/).

Strukturelle Anfragen

- Für Dokumente, die eine Struktur haben, die in einer Suche verwertet werden kann.
- Eine Struktur könnte sein:
  - Feste Menge von Feldern, z.B. Titel, Autor, Abstract, etc.
  - Hierarchische (rekursive) Baumstruktur (z.B. XML):



Anfragen mit Struktur

- Erlaube Anfragen nach Text, der in spezifischen Feldern erscheint:
  - Suche nach “nuclear fusion”, erscheinend in einer Kapitelüberschrift
- SFQL: Relationelle Datenbank-Anfragesprache: SQL erweitert mit “Volltext”-Suche.
  - SELECT Abstract FROM journals.papers

    WHERE author contains “Teller” AND titel like “nuclear fusion %” AND date < 1/1/1950
  - Standard der Air Transport Associatoin/Aircraft Industry Association

Z39.50-Protokoll

- Ursprünglich nur für bibliographische Informationen.
- Inzwischen für andere Datentypen erweitert.
- Anfragesprache zwischen Client- und Server-DBMS.
- Nicht für menschlichen Gebrauch konzipiert.
- ANSI- und NISO-Standard seit 1995
- Version von 2003 unter http://www.loc.gov/z3950/agency/Z39-50-2003.pdf
- Weit verbreitet, insbes. bei Bibliotheken.
- Beispiele:
  - Z39.50-Gateway der Deutschen Bibliothek: http://z3950gw.dbf.ddb.de/
  - Karlsruher Virtueller Katalog: http://www.ubka.uni-karlsruhe.de/kvk.html

Metadaten

- Metadaten enthalten Informationen *über* ein Dokument, die nicht Teil des Dokuments selbst sein müssen (Daten über Daten).
- *Beschreibende* Metadaten liegen außerhalb der Bedeutung des Dokuments:
  - Autor
  - Titel
  - Quelle (Buch, Illustrierte, Zeitung, Zeitschrift)
  - Datum
  - ISBN
  - Verleger
  - Länge

Forts. Metadaten

- *Semantische* Metadaten betreffen den Inhalt:
  - Abstract
  - Schlüsselwörter
  - Klassifikationssysteme
    - Library of Congress
    - Dewey Decimal Classification
    - UMLS (Unified Medical Language System)
- Klassifikationsterme können von spezifischen *Ontologien* kommen (Taxonomie eines standardisierten Vokabulars).

Web-Metadaten

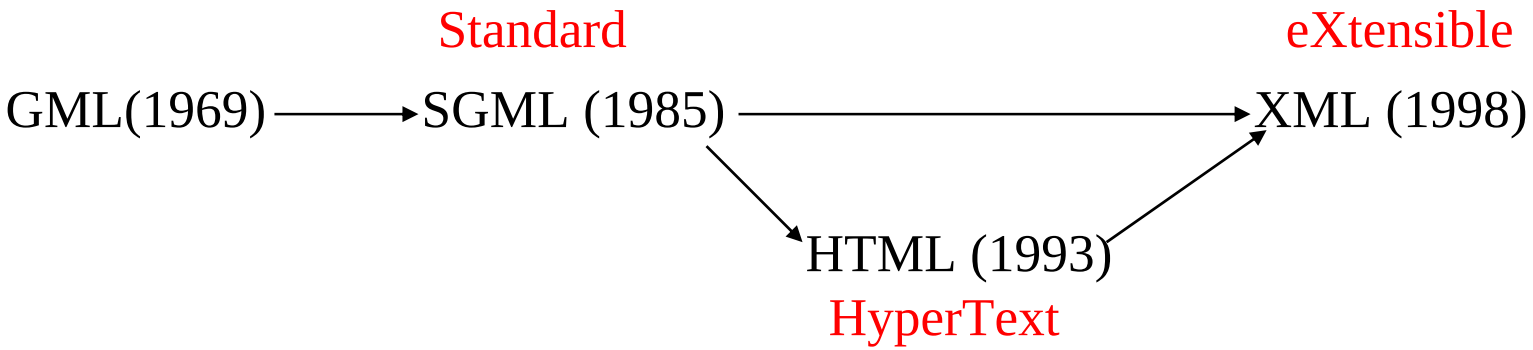
- META-Tag in HTML
  - <META NAME="keywords" CONTENT="pets, cats, dogs">
- META-Attribut "HTTP-EQUIV" erlaubt dem Server oder Browser, auf Informationen zuzugreifen:
  - <META HTTP-EQUIV="content-type" CONTENT="text/tml; charset=EUC-2">
  - <META HTTP-EQUIV="expires" CONTENT="Tue, 01 Jan 02">
  - <META HTTP-EQUIV="creation-date" CONTENT="23-Sep-01">

Inhaltsbewertung durch Metadaten

- PICS (Platform für die Selektion von Internet-Inhalten)
- http://www.w3.org/PICS/
- Bewertungssystem, um das Zensieren basierend auf sexueller oder gewalttätiger Sprache, Inhalt, etc. zu ermöglichen.
  - <META HTTP-EQUIV="PICS-label" CONTENT="PG13: SEX, VIOLENCE">

Markup-Sprachen

- Sprache, die verwendet wird, um Dokumente mit “Tags” zu annotieren, die Layout- oder semantische Informationen beinhalten.
- Die meisten Dokumentsprachen (Word, RTF, Latex, HTML) definieren primär das *Layout*.
- Historie der verallgemein. Markup-Sprachen:



Grundlegende SGML-Syntax

- Textblöcke, die von Start- und End-Tags umgeben sind.
  - <Tagname Attribut=Wert Attribut=Wert …>
  - </Tagname>
- Markierte Textblöcke können verschachtelt sein.
- Im HTML ist das Endtag nicht immer notwendig, aber in XML.

HTML

- Entwickelt für Hypertext im Web.
  - <a href="http://www.cs.utexas.edu">
- Kann einen Code wie Javascript in dynamischem HTML (DHTML) enthalten.
- Eingermaßen separates Layout durch die Verwendung von Style sheets (Cascade Style Sheets, CSS) möglich.
- Jedoch definiert es primär Layout und Format.

XML

- Wie SGML eine Metasprache zur Definition spezifischer Dokumentsprachen.
- Vereinfachung des ursprünglichen SGML für das Web, die vom WWW Consortium (W3C) unterstützt wird.
- Vollständige Trennung von semantischen Informationen und Layout.
- Liefert strukturierte Daten (wie relationale DB) in einem Dokumentformat.
- Ersatz für ein explizites Datenbankschema.

Forts. XML

- Ermöglicht *Programmen*, leicht auf den Inhalt eines Dokuments zuzugreifen, im Gegensatz zu HTML, was als Layoutsprache zur Formatierung von Dokumenten für den menschlichen Leser bestimmt ist.

- Neue Tags werden nach Bedarf definiert.
- Strukturen können beliebig tief verschachtelt sein.

- Separate (optionale) *Dokument Typ Definition* (DTD) definiert Tags und Dokumentgrammatik.
- Ausdrucksstärkere Alternative: XML Schema

XML-Beispiel

```
<Person>
  <Name> <Vorname>John</Vorname>
         <zweiter Name/>
         <Nachname>Doe</Nachname>
  </Name>
  <Alter> 38 </Alter>
  <email> jdoe@austin.rr.com</email>
</Person>
```

<tag/> ist die Kurzschrift für leeres Tag <tag></tag>

Tag-Namen unterscheiden zwischen Groß- und Kleinschrift (im Gegensatz zu HTML)

*Ein markiertes Textstück wird **Element** genannt.*

XML-Beispiel mit Attributen

```
<Produktart=“food”>
    <Name Sprache=“Spanish”>arroz con pollo</Name>
    <Preis-Währung=“peso”>2.30</Preis>
</Produkt>
```

Attributwerte müssen in Anführungszeichen gesetzte Strings sein.

Für ein gegebenes Tag kann jeder Attributname nur einmal erscheinen.

XML - Diverses

- XML Dokumente müssen mit einem speziellen Tag beginnen.
  - <?XML VERSION="1.0">
- Tag "id" und "idref" Attribute ermöglichen es, graphisch strukturierte Daten sowie baumartig strukturierte Daten zu spezifizieren.

```
<state id="s2">
  <abbrev> TX</abbrev>
  <name>Texas</abbrev>
</state>
<city id="c2">
   <aircode> AUS </aircode>
   <name> Austin </name>
   <state idref="s2"/>
 </city>
```

Document Type Definition (DTD)

- Grammatik oder Schema zur Definition von Tags und Strukturen einer speziellen Dokumentart.
- Ermöglicht es, durch Verwendung eines regulären Ausdrucks, die Struktur eines Dokumentelements zu definieren.
- Ein Ausdruck, der ein Element definiert, kann rekursiv sein, was die Ausdrucksstärke einer kontextfreien Grammatik erlaubt.

DTD-Beispiel

```
<!DOCTYPE db [
   <!ELEMENT db (Person*)>
   <!ELEMENT Person (Name,Alter,(Elternteil |
    Vormund)?>
   <!ELEMENT Name (#PCDATA)>
   <!ELEMENT Alter   (#PCDATA)>
   <!ELEMENT Elternteil (Person)>
   <!ELEMENT Vormund (Person)>
]>
```

*: 0 oder mehr Wiederholungen

?: 0 oder 1 (optional)

| : Alternation (or)

PCDATEN: Parsed Character Daten (kann Tags enth.)

Beispiel: ein gültiges Dokument für die DTD

```
<db>
 <Person>
    <Name> <Vorname>John</ Vorname > <Nachname>Doe</ Nachname >
    </Name>
    <Alter> 26 </Alter>
    <ELtern>
        <Person>
          <Name><Vorname>Robert</ Vorname > <Nachname>Doe</
    Vorname >
          </Name>
          <Alter> 55</Alter>
        </Person>
    </Elternteil>
 </Person>
</db>
```

Forts. DTD

- Tag-Attribute werden auch definiert:

  <!ATTLIS Name Sprache CDATA #REQUIRED>

  <!ATTLIS Preiswährung CDATA #IMPLIED>

  CDATA: Eigenschaftsdaten (String)

  IMPLIZIERT: Optional

- Man kann die DTD in einer separaten Datei definieren:

  <!DOCTYPE db SYSTEM “/u/doe/xml/db.dtd”>

XSL (Extensible Style Sheet Language)

- Definiert das Layout für XML-Dokumente.
- Definiert, wie man XML in HTML übersetzt.
- Das Style Sheet wird in einem Dokument wie folgt aufgerufen:
  - <?xml-stylesheet href="mystyle.css" type="text/css">

Standardisierte DTD's

- MathML: für mathematische Formeln.
- SMIL (Synchronized Multimedia Integration Language): Scheduling-Sprache für web-basierte Multi-Media-Präsentationen.
- RDF
- TEI (Text Encoding Initiative): für literarische Werke.
- NITF: für Nachrichten-Artikel.
- CML: für Chemikalien.
- AIML: für astronomische Instrumente.

Parsen von XML

- Wandelt XML-Datei in ein internes Datenformat zur weiteren Verarbeitung um.
- SAX (Simple API for XML): Liest den XML-Text, wobei *Ereignisse* (z.B. Start- und End-Tag) erfasst und zur Verarbeitung an die Anwendung zurückgegeben werden.
- DOM (Document Object Model): Parst XML-Text in eine baumstrukturierte objektorientierte Datenstruktur.

DOM

- Darstellung eines XML-Dokuments als ein Baum von Knoten-Objekten (z.B. Java Objekten).
- Knoten-Klasse hat Unterklassen:
  - Element
  - Attribute
  - CharacterData
- Knoten hat Methoden:
  - getParentNode()
  - getChildNodes()

Weitere Knoten-Methoden

- Element-Knoten
  - getTagName()
  - getAttribute()
  - getAttribute(Stringname)
- CharacterData-Knoten
  - getData()
- Es gibt auch Methoden für das Hinzufügen und das Entfernen von Knoten und Text im DOM-Baum, für das Bestimmen von Attributen, etc.

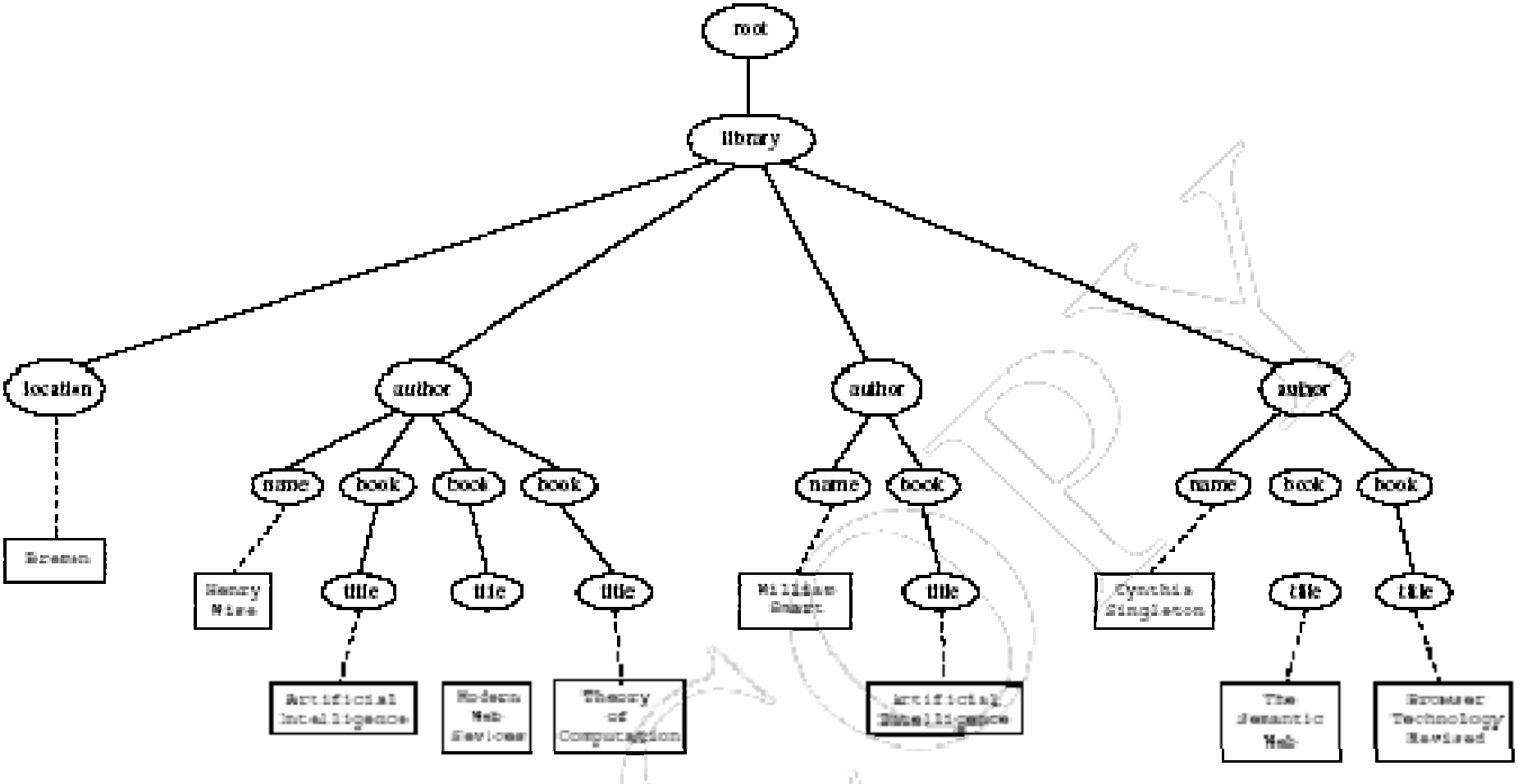
Apache Xerxes XML Parser

- Parser für das Bilden des DOM-Baums eines XML-Dokuments.
- Java-Version verfügbar unter:
  - http://xml.apache.org/xerxes-j/
- Voll-Javadoc verfügbar unter:
  - http://xml.apache.org/xerxes-j/apiDocs/

Ein XML-Beispiel

```
<library location="Bremen">
    <author name="Henry Wise">
        <book title="Artificial Intelligence"/>
        <book title="Modern Web Services"/>
        <book title="Theory of Computation"/>
    </author>
    <author name="William Smart">
        <book title="Artificial Intelligence"/>
    </author>
    <author name="Cynthia Singleton">
        <book title="The Semantic Web"/>
        <book title="Browser Technology Revised"/>
    </author>
</library>
```

DOM-Darstellung



Anfragen an XML-Dokumente

- Das zentrale Konstrukt einer XML-Anfragesprache sind Pfadausdrücke.
  - Sie legen fest, welche Knoten des DOM erreicht werden können.

- XPath ist die Basis für XML-Anfragesprachen.
  - Sprache, um Teile eines XML-Dokuments zu adressieren.
  - Operiert im Baummodell von XML.
  - Hat eine Nicht-XML-Syntax.

Beispiele für Pfadausdrücke in XPath

/library/author

- Addressiert alle **author**-Elemente, die Kinder von **library**-Elementen sind, die direkt unter der Wurzel hängen.
- Allgemein sucht **/t1/.../tn** einen Pfad von der Wurzel bis zu Element tn.

Beispiele für Pfadausdrücke in XPath

//author

- // gibt an, dass wir alle Elemente im Baum darauf testen sollen, ob sie vom Typ **author** sind.
- Dies sucht alle **author-**Elemente irgendwo im Dokument.

Beispiele für Pfadausdrücke in XPath

- Suche alle Bücher mit dem Titel “Artificial Intelligence”

  /book[@title="Artificial Intelligence"]

- Der Test in den eckigen Klammern ist ein Filterausdruck, der aus der Menge der Treffer selektiert.

Resource Description Framework (RDF)

- kompatibel mit XML-Syntax.
- Hat ein Graphmodell anstelle eines Baummodells als Semantik. (Ist also inkompatibel zur XML-Semantik.)
- Neuer Standard für Web-Metadaten.
  - Beschreibung des Inhalts
  - Beschreibung der Sammlung
  - Datenschutz-Informationen
  - Intellektuelle Besitzrechte (z.B. Copyright)
  - Rating des Inhalts
  - Digitale Unterschriften für Berechtigungen

Beispiel

```
<uni:lecturer rdf:ID="949352">
    <uni:name>Grigoris Antoniou</uni:name>
</uni:lecturer>

<uni:professor rdf:ID="949318">
            <uni:name>David Billington</uni:name>
</uni:professor>

<rdfs:Class rdf:about="#professor">
    <rdfs:subClassOf rdf:resource="#lecturer"/>
</rdfs:Class>
```

- Eine Anfrage nach den Namen aller Dozenten (lecturer) sollte sowohl Grigoris Antoniou als auch David Billington ergeben.

Beispiele

- Select-from-where ähnlich wie in SQL:

- Suche nach den Telephonnummern aller Angestellten:

  select X,Y

  from {X}phone{Y}

- **X** und **Y** sind Variablen, und **{X}phone{Y}** repräsentiert ein Objekt-Attribut-Wert-Tripel.

Beispiele

- Suche alle Dozenten und ihre Telefonnummern (mit implizitem Join):

select X,Y
from lecturer{X}.phone{Y}

Beispiele

- Gib die Namen aller Vorlesungen aus, die von dem Dozenten mit ID 949352 gehalten werden (mit explizitem Join):

select N
from course{X}.isTaughtBy{Y}, {C}name{N}
where Y="949352" and X=C

Schemaanfragen in RDF

- Schema-Variablen beginnen mit **$** (für Klassen) und **@** (für Attribute).
- Gib alle Ressourcen und Tripel mit dem Attribut **phone** aus (eingeschlossen die jeweiligen Unterklassen und Unterattribute):

```
select X,$X,Y,$Y
from {X:$X}phone{Y:$Y}
```

Web Ontology Language (OWL)

- OWL erweitert RDF um logische Konstrukte.
- Ist ausdruckstärker als RDF.
- Entspricht einem entscheidbaren Fragment der Prädikatenlogik erster Stufe.
- Basiert auf der Beschreibungslogik SHOIN(D).
- Anfragen werden als logische Formeln gestellt.

- Weitere Details zu RDF und OWL:
  - http://www.w3.org/TR/rdf-concepts/
  - http://www.w3.org/2004/OWL/
  - Master-Vorlesung “Künstliche Intelligenz”