Prof. Dr.-Ing. Stefan Deßloch
AG Heterogene Informationssysteme
Geb. 36, Raum 329
Tel. 0631/205 3275
dessloch@informatik.uni-kl.de

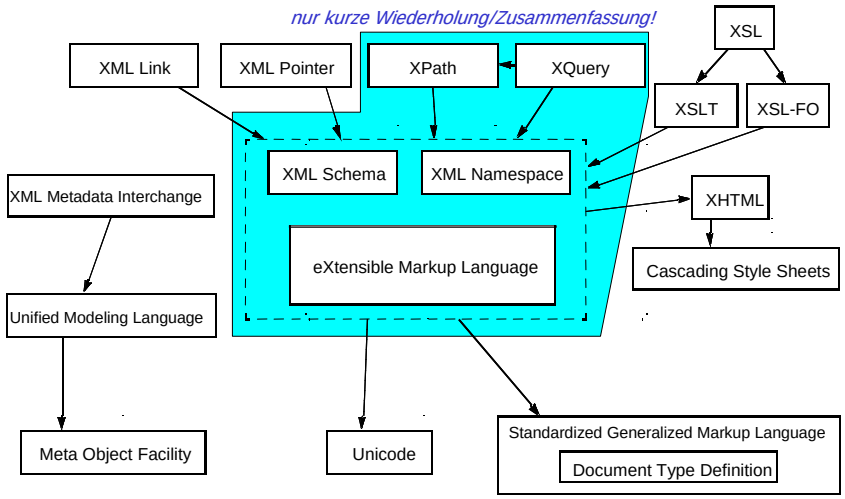
Technische Universität Kaiserslautern

Kapitel 12: XML

Anwendungsbereiche von XML

- **Dokument-zentriert:** tief strukturierte Dokument-Hierarchie für viele Dokumentinstanzen
  - grob-granulare Daten
  - Reihenfolge signifikant
  - Beispiele: Bücher, Email, Werbung, ...
  - Nutzung durch Personen
- **Daten-zentriert:** große Bestände in Datenbanksystemen von dort aus in ein XML-Format umwandeln
  - fein-granulare Daten
  - Reihenfolge nicht signifikant
  - Beispiele: Flugpläne, Speisekarten, ...
  - maschinelle Verarbeitung
- [**EDI**] (Electronic Data Interchange): Datenformat zur Kommunikation (Austausch von Informationen) zwischen Anwendungen

XML-Sprachspezifikationen (W3C)

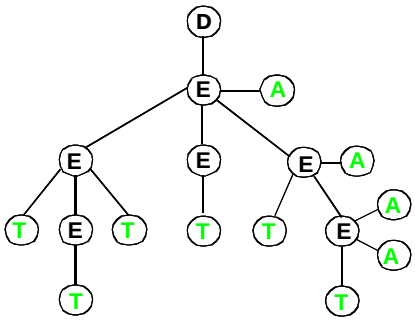


XML-Dokumente - Aufbau

- Prolog
  - XML-Deklaration
  - Dokumenttyp-Deklaration
    - Angabe der verwendeten DTD
- XML-Dokument besteht aus Markup-Elementen
- Markup-Element
  - besteht aus Tag-Paar (Begin-Tag, End-Tag)
    - enthält Zeichendaten
    - kann leer sein
    - Schachtelung möglich
  - Elementnamen werden bzgl. Gross-/Kleinschreibung unterschieden
  - können näher durch Attribute beschrieben werden
    - haben Namen (innerhalb des Elements eindeutig) und Wert
    - 3 Typen
      - Zeichendaten
      - Aufzählungen
      - über Schlüsselwörter, wie ID oder IDREF, ausgezeichnete Zeichendaten
- Dokumente müssen wohlgeformt (well-formed) sein

XML-Datenmodell

- Es existiert kein einheitliches Datenmodell für XML
  - Verschiedene Ansätze mit unterschiedlichem Ziel
    - XML Information Set, DOM Structure Model, XQuery-Datenmodell (enthält XPath), ...
  - Gemeinsame Sicht: XML-Dokument als Baumstruktur mit unterschiedlichen Knotentypen
    - Document, Element, Attribute, Text, Comment, ...



Schemadefinition für XML-Dokumente

- XML-Dokument kann (optional) Schema besitzen
  - standardisierter Datenaustausch, ...
- Schema schränkt die für das Dokument erlaubten Strukturen und Datentypen ein
  - Dokument heisst gültig (valid), falls es die Anforderungen des Schemas erfüllt
  - rekursive Schemadefinitionen erlaubt
    - Bauteil besteht aus weiteren Bauteilen ...
- Zwei wichtige Ansätze
  - Document Type Definition (DTD)
    - im Dokument enthalten, oder
    - in einer separaten Datei gespeichert, im Dokument referenziert
  - XML Schema

XML Document Type Definition (DTD)

- Definition von Elementtypen
  - Welche Elemente dürfen vorkommen
  - Welche Attribute darf bzw. muss ein Element haben
  - Welche Unterelemente dürfen bzw. müssen in einem Element auftreten, und wie oft
- DTD bietet keine Unterstützung für Datentypen
  - Daten sind, wie gehabt, nur Zeichenketten ohne weitere Einschränkungen
- DTD-Syntax

  <!ELEMENT element (subelements-specification) >

  <!ATTLIST element (attributes) >

Schemabeschreibung mit XML Schema

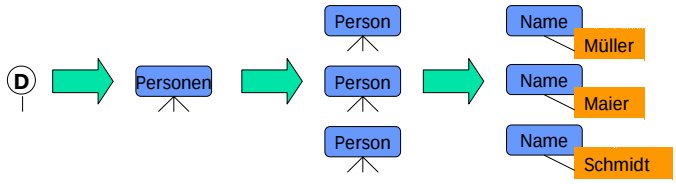
- Unterstützung von Datenmodellierungskonzepten, u.a.
  - Datentypen
    - integer, string, ...
  - Constraints
    - min/max-Werte für mgl. Anzahl von Elementwiederholungen
    - Werteindeutigkeit (unique), Fremdschlüssel
  - Getypte Referenzen
  - Namensräume in XML (name spaces)
  - Benutzerdefinierte Typen
    - lokale Definition oder Referenz auf entsprechenden Namensraum
    - mehrfache Benutzung
- Modularisierung und Wiederverwendung von Schemadefinitionen
- Schemadefinition ist wiederum ein XML-Dokument
- Deutlich komplizierter als DTDs

Namensräume (Namespaces)

- Ziel: Vermeidung von Namenkollisionen bei Nutzung von verschiedenen Vokabularen (Schemata) im gleichen Dokument
  - Beispiel
    - Ein Element Titel kommt sowohl im Kontext von Büchern (Buchtitel) als auch im Kontext von Personen (z.B. akademischer Titel) vor
    - Ein Dokument das Information über Bücher und Autoren enthält soll Bestandteile aus beiden Schemata verwenden können
- Namensraum
  - „enthält“ eine Menge von Namen
  - durch URI „weltweit“ eindeutig identifiziert
  - Kann in einem XML-Dokument oder Element genutzt werden
    - Deklaration als Default-Namespace
    - Definition und Angabe von “Kürzeln” (prefix)
      - <lit:titel>, <pers:titel>, ...

Pfadausdrücke in XPath, XQuery

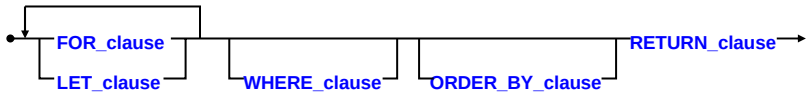
- Pfadausdruck adressiert (selektiert) ein Sequenz von Knoten in einem Dokument
  - besteht aus Schritten, durch “/” voneinander getrennt
    - Bsp.: Namen aller Personen
      - /child::Personen/child::Person/child::Name
  - wird sukzessive, von links nach rechts ausgewertet
    - Pfadanfang: Dokumentwurzel oder von außen vorgegebener Kontext
    - Jeder Schritt geht von Knotensequenz aus
      - Sucht für jeden Knoten in der Sequenz weitere Knoten auf
      - Duplikateliminierung aufgrund der impliziten Knotenidentität
      - Mglw. Sortierung der Knoten in Dokumentreihenfolge
    - Leere Resultate führen nicht zu Fehler



XQuery – Überblick

- Pfadausdrücke sind wesentlicher Bestandteil einer XML-Anfragesprache
- Weitere Sprachkonstrukte werden benötigt
  - Konstruktion neuer XML-Objekte als Anfrageresultat
    - Bsp.: Schachtelung der durch einen Pfadausdruck lokalisierten Elemente in ein neues Element zur Ausgabe
  - Binden und Nutzen von Variablen zur Iteration über mehreren Sequenzen von Knoten
  - Verbundoperationen
  - Sortierung
  - Aggregation
  - ...
- Wesentliche zusätzliche Konzepte in XQuery
  - Konstruktoren
  - FLWOR-Ausdrücke (gespr.: flower)

FLWOR – Ausdrücke



- FOR, LET binden Variablen durch
  - Iteration über eine Sequenz von Knoten (FOR)
  - Auswertung eines Ausdrucks (LET): Ausdruck liefert Sequenz von Items, welche an Variable gebunden wird
- WHERE ermöglicht Selektion von Variablenbelegungen aufgrund von Prädikaten
- ORDER BY erlaubt Sortieren von Inhalten
- RETURN generiert Resultat des Ausdrucks anhand der Variablenbelegungen
- Vergleichbar mit folgenden Konzepten in SQL:
  for ⇔ SQL from
  where ⇔ SQL where
  order by ⇔ SQL order by
  return ⇔ SQL select
  let hat keine Entsprechung

FLWOR – Beispiel

- Suche alle Personen (Name, Alter) jünger als 25
  - Variablen beginnen mit “$”-Präfix
  - Anfrage ohne LET, WHERE:

```
for     $p in //Person[Alter < 25]
order by $p/Name
return <jungePerson> {$p/Name, $p/Alter} </jungePerson>
```

  - Einfache (äquivalente) FLWOR-Ausdrücke
    (siehe Auswertungsbeispiel 1: Sequenzen Name und Alter bestehen in allen Fällen nur aus einen Element)

```
for     $p in //Person
let     $n := $p/Name, $a := $p/Alter
where   $a < 25
order by $n
return <jungePerson> {$n, $a } </jungePerson>
```

```
for     $p in //Person[Alter < 25]
let     $n := $p/Name, $a := $p/Alter
order by $n
return <jungePerson> {$n, $a } </jungePerson>
```


Anwendungsprogrammierung mit XML

- Anwendungsschnittstellen zum Verarbeiten von XML Daten/Dokumenten
  - **Parsen** von XML um relevante Informationen zu extrahieren
  - Erzeugen von XML
    - als Zeichenkette
    - Aufbau einer internen XML Dokumentrepräsentation, anschließendes **Serialisieren**
  - *Simple API for XML (SAX)*
    - “Push”-parser (ereignisbasiert)
    - Parser benachrichtigt die Anwendung bei Erreichen/Erkennen von Dokumentbestandteilen (Elemente/Fragmente, etc.)
    - Benachrichtigungen erfolgen in der durch das serlialisierte Dokument vorgegebenen Reihenfolge
    - Haupspeichereffizientes Verfahren, vorteilhaft für große Dokumente
  - *Document Object Model (DOM)*
    - Parser erzeugt Haupspeicherdarstellung des Dokuments (*parse tree*)
    - DOM spezifiziert die versch. Arten von Objekten im parse tree, Eigenschaften, Operationen
      - Programmiersprachenunabhängig
    - Anbindung an spez. Programmiersprachen definiert (z.B., Java)

DOM

- DOM-Struktur ist eine Baumstruktur mit untersch. Knotentypen

Node type	Contains
Document	Element (maximum of one), ProcessingInstruction, Comment, DocumentType
DocumentFragment	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
DocumentType	no children
EntityReference	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
Element	Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference
Attr	Text, EntityReference
ProcessingInstruction	no children
Comment	no children
Text	no children
CDATASection	no children
Entity	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
Notation	no children

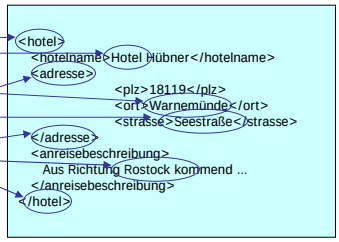
Speicherung von XML

- Anforderungen an die Speicherung
  - Effektive Speicherung
  - Effizienter Zugriff auf XML-Dokumente oder Teile davon
  - Wiederherstellbarkeit der Dokumente (oder der Informationen aus den Dokumenten)
- Mögliche Ansätze zur Speicherung
  - Speicherung der XML-Dokumente als Ganzes und Indizierung (textbasiert native Verfahren)
    - Volltextindex
    - Volltext- und Strukturindex
  - Dekomposition und generische Speicherung
    - Graphspeicherung
    - Speicherung der DOM-Informationen
  - strukturierte Abbildung auf Datenbanken
    - relationale Datenbanken
    - objekt-orientierte und objekt-relationale Datenbanken
    - Einsatz von benutzerdefinierten Mappingverfahren

Speicherung von XML Dokumenten als Ganzes

Volltextindex

Term	Verweis
Hotel	
Warnemünde	
Seestraße	
Rostock	
adresse	

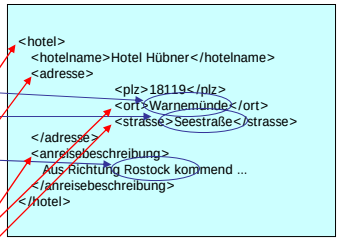


- XML Dokument wird unverändert gespeichert
- zusätzlich Volltextindex (z.B. invertierte Wortliste)
  - Volltextretrieval
  - keine Unterscheidung von Markup (Elementnamen) und Textinhalt
  - keine Strukturinformation

Speicherung von XML Dokumenten als Ganzes

Volltextindex

Term	Verweis	Element
Warnemünde		
Seestraße		
Rostock		



XML-Index

Element	Verweis	Vorgänger
hotel		
adresse		
ort		
strasse		
anreise-beschreibung		

zusätzliche Strukturinformation

Eigenschaften

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

Dekomposition und Generische Speicherung

- Speicherung der Graphstruktur
  - Knoten: Elemente und Attribute
  - Kanten: Beziehung zu Subelement/Attribut
  - oft RDBMS als Basis, generisches Schema
    - Tabelle **ELEMENTE** mit Spalten DOCID, ELEMENTNAME, ID, VORGÄNGER, POSITION, WERT
    - Tabelle **ATTRIBUTE** mit Spalten DOCID, ATTRIBUTNAME, ELEMENTID, WERT

Elemente

docID	Elementname	ID	Vorgänger	Position	Wert
H0001	hotel	001		1	
H0001	hotelname	002	001	1	Hotel Hübner
H0001	adresse	003	001	1	

- Speicherung der Informationen des Document Object Model
  - generisches Speicherungsschema orientiert sich an den DOM Node Types
    - Document, Element, Attribute, ...
  - auch hier RDBMS als Basis möglich

Eigenschaften

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

Systeme

Infonyte-DB

- verwendet zur Speicherung von XML-Dokumenten ein persistentes Document Object Model (PDOM)
- baut aber nicht auf existierende Datenbanken auf, sondern entwickelt Komponenten zur physischen Speicherung, die an die XML-Dokumente optimal angepasst sind
- Anfragesprache: Richtung XQuery

Tamino

- modellbasierte Speicherung von XML-Dokumenten
- XML-Anfragen auf den gespeicherten Dokumenten sind durch die Verwendung von XPath realisierbar
- Entwicklung geht Richtung erweitertes XPath, das den Namen Tamino XQuery trägt

eXcelon

- verwendet zur Speicherung von XML-Dokumenten das Document Object Model
- Alle Informationen, die in dieser API enthalten sind, werden in einer objektorientierten Datenbank objectstore adäquat gespeichert.
- Anfragen: OQL

Strukturierte Speicherung in Datenbanken

- XML Dokumentstruktur auf Schemaebene repräsentiert
  - Voraussetzung: explizites Schema der XML-Dokumente
- Abbildung auf ein (anwendungsspezifisches) DB-Schema
  - vom System per default festgelegt
  - benutzerdefinierbar
- Abbildungsvorschriften
  - Transformation von Anfrageresultaten
  - DB-Schemagenerierung
- Beispiel (objekt-relationales DBMS)

Hotel

id	hotelname	adresse			...
		plz	ort	strasse	
0001	Hotel Hübner	18119	Warnemünde	Seestraße 12	

Vor- und Nachteile

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

Eigenschaften

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

Systeme: Systemdefinierte Abbildung

POET

- zu jedem Element der DTD oder der XML-Schema-Beschreibung wird eine korrespondierende Java-Klasse erzeugt
- Diese wird innerhalb des Systems gespeichert, indem dazu eine Relation generiert wird
- Ist kein Schema für eine Dokumentkollektion vorhanden, so wird hier eine Datenbank mit festem Schema eingesetzt. Diese Schema entspricht den Informationen des Document Object Model. (siehe vorn)

Oracle 8i / Oracle 9i

- Zum Datenbanksystem Oracle werden seit der Version 8i Tools angeboten, die die XML-Speicherung unterstützen.
- Bestandteil des Oracle XML Developer's Kit (XDK)
- Verfügbar sind zum Beispiel XML-Parser und XSL-Transformator

Systeme: Benutzerdefinierte Abbildung

IBM DB2 XML-Extender

- Speicherung von XML-Dokumenten im Datenbanksystem DB2
- über eine Mappingvorschrift (DAD - Data Access Definition) wird angegeben, wie die Zuordnung von Elementen und Attributen eines Dokuments auf die Attribute der Datenbank erfolgen soll.
- Die Syntax der DAD-Dateien ist XML

Oracle

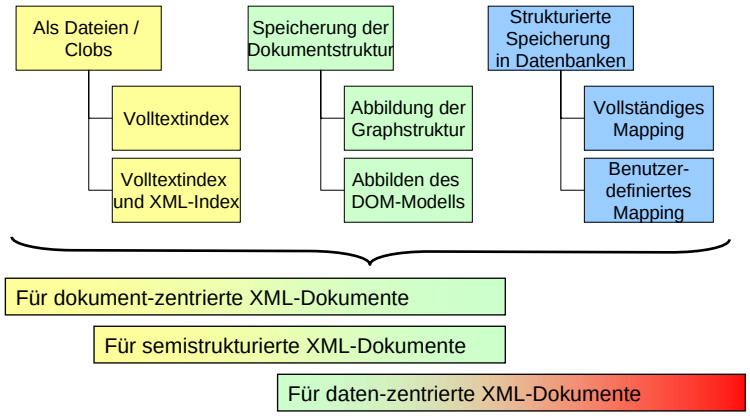
- objektrelationale Speicherung des XMLType
- Art der Speicherung wird durch ein annotiertes XML-Schema beschrieben, dieses enthält eine Zuordnung von Datenbankinformationen zu den XML-Bestandteilen
- Dadurch benutzerdefinierte Speicherung
- Zugriff auf die so gespeicherten XML-Dokumente mit XPath oder SQL

Systeme: Benutzerdefinierte Abbildung

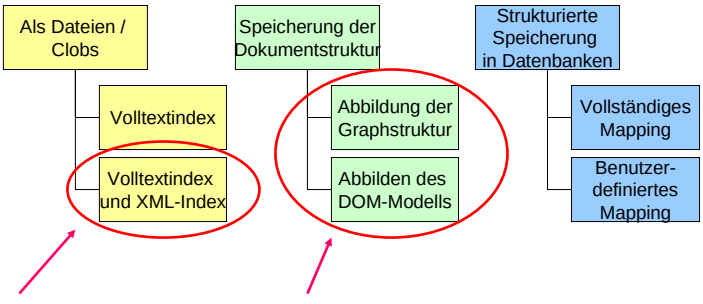
Microsoft SQL-Server

- speichert XML-Dokumente in Datenbanken
- anwenderdefiniertes Mapping wird verwendet, das durch ein annotiertes XDR-Schema festgelegt wird.
- Annotationen beschreiben die Zuordnung zwischen XML-Dokument und relationaler Datenbank, zum Beispiel: sql:relation und sql:field, bestimmen die Zuordnung zwischen Elementen und Attributen zu Datenbankrelationen und Datenbankattributen.

Speicherung von XML-Dokumenten



Was ist native Speicherung?



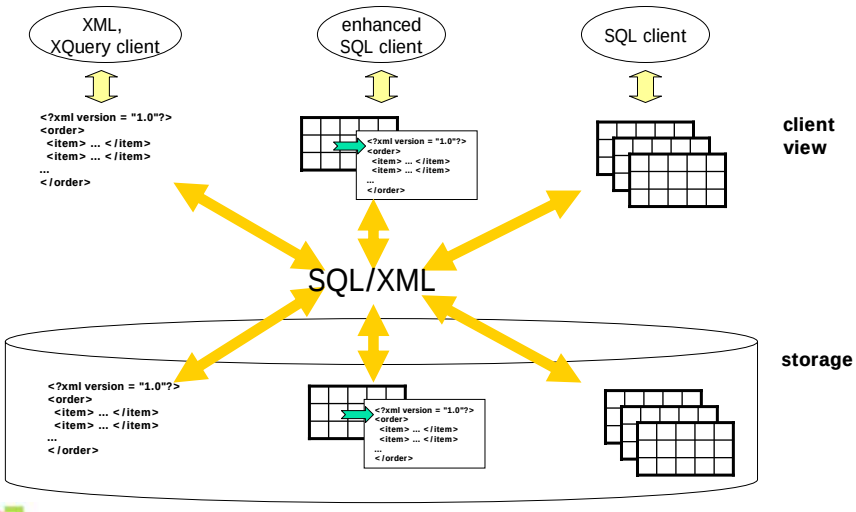
Textbasierte native Speicherung

Modellbasierte native Speicherung

SQL und (natives) XML?!

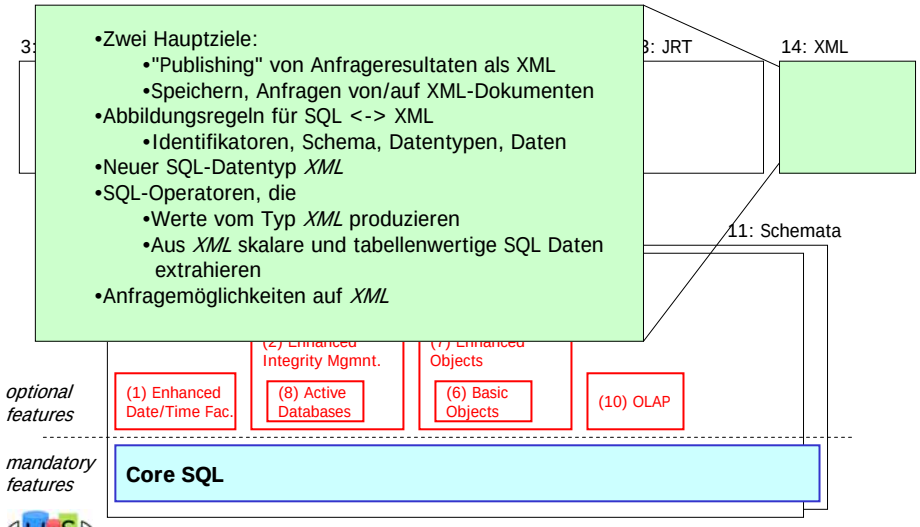
- Nutzung von (objekt-) relationaler Technologie ist problematisch
  - large objects: Granularität zu grob
    - Suche, Änderungen problematisch
  - Strukturierte Speicherung in Tabellen
    - komplex, i.A. ineffizient
- Ziel: "hybride" XML/relationale Datenverwaltung
  - unterstützt relationale und XML Daten
    - Speicherung, Zugriff
    - Anfrageformulierung
  - ermöglicht XML-Sicht auf relationalen Daten, und relationale Sicht auf XML-Daten

SQL/XML Big Picture



SQL:2003 Parts and Packages

- Zwei Hauptziele:
  - "Publishing" von Anfrageresultaten als XML
  - Speichern, Anfragen von/auf XML-Dokumenten
- Abbildungsregeln für SQL <-> XML
  - Identifikatoren, Schema, Datentypen, Daten
- Neuer SQL-Datentyp *XML*
- SQL-Operatoren, die
  - Werte vom Typ *XML* produzieren
  - Aus *XML* skalare und tabellenwertige SQL Daten extrahieren
- Anfragemöglichkeiten auf *XML*



XLink - XML Linking Language

- Hyperlinks/Verweise in XML-Dokumenten
  - separate Spezifikation
  - nutzt Uniform Resource Identifiers (URIs), Xpath, XPointer als Referenzierungsmechanismen
  - mächtiger als HTML-Hyperlinks (siehe auch Betrachtungen in Kapitel 8)
    - bidirektional, mehr als zwei beteiligte Resourcen
    - leistungsfähige Addressierung von Resourcen
      - gezielter Bezug auf Objektkomponenten
    - Link-Attributierung durch Metadaten
    - Speicherung des Links unabhängig von den Resourcen
  - aber auch direkte Unterstützung "einfacher" links (vgl. HTML) durch spezielle Syntax

XLink Elemente und Attribute

- XLink Element
  - enthält spezielle Attribute aus dem XLink-Namensraum
    - **type (simple, extended, ...)**
    - **href** (URI-Referenz oder XPointer)
    - **title, role** (Beschreibung der Linksemantik)
    - **show (new, replace, embed, undefined)** (Anzeige beim Aktivieren)
    - **actuate (onLoad, onRequest, undefined)** (Verfolgen von links)
    - **from, to** (Definition von gerichteten Kanten im Link-Graph)
- Arten von Verweisen
  - outbound: lokale Start- , entfernte End-Resource
  - inbound: entfernte Start- , lokale End-Resource
  - third-party: entfernte Start- , entfernte End-Resource
  - es können mehrere Resourcen beteiligt sein
  - multi-direktionale Verweise
  - Verweise auf ausschließlich lesbare Rssourcen

XLink - Beispiel

- in der DTD:

```
<!ELEMENT Spieler ANY>
<!ATTLIST Spieler
                    xlink:type  (simple)   #FIXED    "simple"
                    xlink:href  CDATA   #REQUIRED
                    xlink:role   NMTOKEN  #FIXED "http://www.fck.com/links/spieler"
                    xlink:title   CDATA    #IMPLIED
                    xlink:show    (new|embed|replace)   "replace"
                    xlink:actuate (onLoad|onRequest)   "onRequest"
          >
```

- in der Dokumentinstanz:

```
<Spieler xlink:href="http://www.fck.de/Spielerliste.xml"
                    xlink:title="Liste aller FCK-Spieler"
                    xlink:show="new">
                    Hier geht es zu einer Liste aller FCK-Spieler.
</Spieler>
```

XSL – Transformation und Darstellung

- **XSL**: *Extensible Stylesheet Language*
  - eigene formating engine für XML
  - darstellungsunabhängiges Markup von XML
  - medienunabhängige Präsentation
  - **XSLT**: *XSL Transformation Language*
    - Formatvorlage (stylesheet)
    - Transformationsregeln
      - jeweils bestehend aus einem Pattern und einem Template
      - Nutzung von XPath
    - Ursprungsbaum
    - Ergebnisbaum
  - **XSL-FO**: *XSL Formatting Objects*
    - Vokabular für die Spezifizierung von Formatierungsregeln
      - Wiederverwendung von komplexen Formatierungen
    - Umwandlung in beliebige Ausgabeformate (PDF, RTF, PostScript, ...)

Funktionsweise von XSLT

- XSL-Prozessor
  - durchläuft die Elemente des Ursprungsbaums, beginnend mit Wurzel
  - sucht im Stylesheet nach passender XSLT-Regel
- XSLT-Regel legt fest
  - Schablone (template): für welches Element, unter welchen Verwandtschaftsbeziehungen die Regel gilt
  - Aktion:
    - was an Ausgabe generiert werden soll
      - Bezugnahme auf bestimmte Inhalte mit Hilfe von Select-Ausdrücken
    - mit welchen Elementen der Prozessor die Ausführung fortsetzen soll
      - <xsl:apply-templates/> - fahre mit Kindelementen fort
      - erweiterte Syntax erlaubt Auswahle bestimmter Elemente, Sortierung, etc.
- Default-Regeln (falls keine Regel greift)
  - für alle Elemente (inkl. Wurzel), bearbeite die Kindelemente
  - für alle Textknoten und Attribute, nimm die Werte als Ausgabe

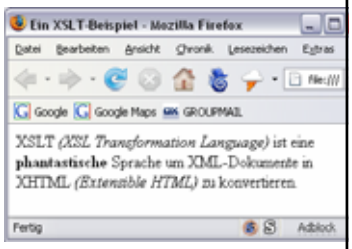
XSL - Beispielanwendung

- 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>
```



XSL - Beispiel

- 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>
```

XML and CM/DL

- XML can be used to represent documents and data
- but XML is a text-oriented language
  - not really suitable for multimedia content
- Multi-media content can be referenced in XML documents
  - URI, XLink, XPointer, XPath
- Multi-media content can be encoded in a text-based format
  - Scalable Vector Graphics (SVG)
  - Synchronized Multimedia Integration Language (SMIL)
- XML for meta-data representation
  - Meta-data standards (e.g., Dublin Core)
  - Representation in XML
    - RDF

Synchronized Multimedia Integration Language

- Creations of multi-media presentations, declarative description of
  - presentation layout
  - objects involved
  - timing aspects
- SMIL is **not** a container format
  - multi-media objects (pictures, audio, video, ...) are not included, only referenced via a URI
    - same object can be included in multiple presentations
    - integration of "remote" objects (e.g., weather chart)
- Object alternatives in SMIL
  - same object in different resolutions, dynamically selected based on available bandwidth
  - text or audio stored in different languages, selected based on user language preferences
- Interactive capabilities (limited)
  - follow presentation links

SMIL Documents

- XML documents following the SMIL DTD or Schema
  - document header
    - general presentation properties, such as layout, position of regions, …
    - based on a layout language close to cascading style sheets (CSS)
  - document body
    - actual definition of the presentation (incl. timing aspects)
    - number of XML element tags for different types of MM-objects
      - <audio>, <img>, <video>, <textstream>
      - Attributes for representing object URI, presentation region, MIME-type, presentation duration
    - objects can be "hyperlinked" to other resources on the web
    - timing aspects
      - parallel presentation (i.e., at the same time)
      - sequential presentation
      - duration attribute

Example:

```
<seq>
    <img id="a" dur="6s" begin="0s" src="..." />
    <img id="b" dur="4s" begin="0s" src="..." />
    <img id="c" dur="5s" begin="2s" src="..." />
</seq>
```


Resource Description Framework (RDF)

- Language for representing information (e.g., meta data) about resources on the web
  - identify something on the web using Uniform Resource Identifier (URI)
  - describe it using simple property/value pairs
- RDF statement can be represented using a graph
  - Example (from the RDF spec): "there is a Person identified by http://www.w3.org/People/EM/contact#me, whose name is Eric Miller, whose email address is em@w3.org, and whose title is Dr."
- RDF heritage: knowledge representation
  - semantic networks

RDF/XML

- XML-based syntax for encoding and exchanging RDF statements
  - Example

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf=...>
<contact:Person
  rdf:about=
  "http://www.w3.org/People/EM/contact#me">
   <contact:fullName>
                Eric Miller
   </contact:fullName>
   <contact:mailbox
     rdf:resource="mailto:em@w3.org"/>
  <contact:personalTitle>
                Dr.
  </contact:personalTitle>
</contact:Person>
</rdf:RDF>
```

- Could be used to provide semantic markup for XHTML documents
  - extension of the HTML META tag

Dublin Core

- Meta-data standard for describing networked resources
  - established by international, cross-disciplinary group of professionals from librarianship, computer science, text encoding, the museum community, and other related fields
  - major goal: help improve resource discovery
  - also used in closed environments, for other purposes(e.g., meta-data exchange)
- Meta-data description
  - uses a set of common meta data elements (nouns) and qualifiers (adjectives)
    - can be embedded in the resource (e.g., as HTML meta tags)
    - can be contained in a separate record/description of a resource (e.g., in a meta-data catalog file or database)
- Dublin Core defines
  - a vocabulary for meta data
    - simple to use, based on commonly used semantics, international, extensible,
  - best practices of how to use the language in various formats
    - HTML, XML, RDF

Dublin Core Elements

- Elements can be broadly grouped into three categories
  - Content
    - **Title**: A name given to the resource.
    - **Subject**: The topic of the content of the resource.
    - **Description**: An account of the content of the resource.
    - **Type**: The nature or genre of the content of the resource.
    - **Source**: A reference to a resource from which the present resource is derived.
    - **Relation**: A reference to a related resource.
    - **Coverage**: The extent or scope of the content of the resource.
  - Intellectual Property
    - **Creator**: An entity primarily responsible for making the content of the resource.
    - **Contributor**: An entity responsible for making contributions to the resource content.
    - **Publisher**: An entity responsible for making the resource available
    - **Rights**: Information about rights held in and over the resource.
  - Instantiation
    - **Date**: A date associated with an event in the life cycle of the resource.
    - **Format**: The physical or digital manifestation of the resource.
    - **Identifier**: An unambiguous reference to the resource within a given context.
    - **Language**: A language of the intellectual content of the resource.

DC Elements (cont.)

- Each element is optional and repeatable
- There is no defined order of elements
- Definition of controlled vocabularies possible (i.e., permitted values for elements)
  - uses concept of qualifiers

Qualifiers

- Two broad classes
  - Element Refinement: make the meaning of an element narrower or more specific
  - Encoding Scheme: identify schemes that aid in the interpretation of an element value
- Example: Element **Date**
  - Refinements
    - Created, Valid, Available, Issued, Modified, Date Copyrighted, Date Submitted
  - Encoding Schemes
    - DCMI Period, W3C-DTF
- Example: Element **Relation**
  - Refinements
    - Is Version Of, Has Version, Is Replaced By, Replaces, Is Required By, Requires, Is Part Of, Has Part, Is Referenced By, References, Is Format Of, Has Format, Conforms To
  - Encoding Scheme
    - URI

DC XML Implementation Guidelines

- Each DC element is represented as a separate XML element
  - refinements become elements of their own
  - encoding schemes are represented using xsi:type
- Example

```
<metadata xmlns=…>
  <dc:title> UKOLN </dc:title>
  <dc:subject> national centre, network information support</ dc:subject>
  <dc:identifier xsi:type="dcterms:URI"> http://www.ukoln.ac.uk/ </dc:identifier>
  <dcterms:modified xsi:type="dcterms:W3CDTF">
          2001-07-18
  </dcterms:modified>
  …
</metadata>
```

DC RDF Implementation Guidelines

- Each resource is described in an RDF description element
  - most appropriate URI to be used for rdf:about attribute
- Example

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc ="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://dublincore.org/">
    <dc:title>Dublin Core Metadata Initiative - Home Page</dc:title>
    <dc:description>
        The Dublin Core Metadata Initiative Web site.
    </dc:description>
    <dc:date>2001-01-16</dc:date>
    <dc:format>text/html</dc:format>
    <dc:language>en</dc:language>
    <dc:contributor>The Dublin Core Metadata Initiative</dc:contributor>
  </rdf:Description>
</rdf:RDF>
```