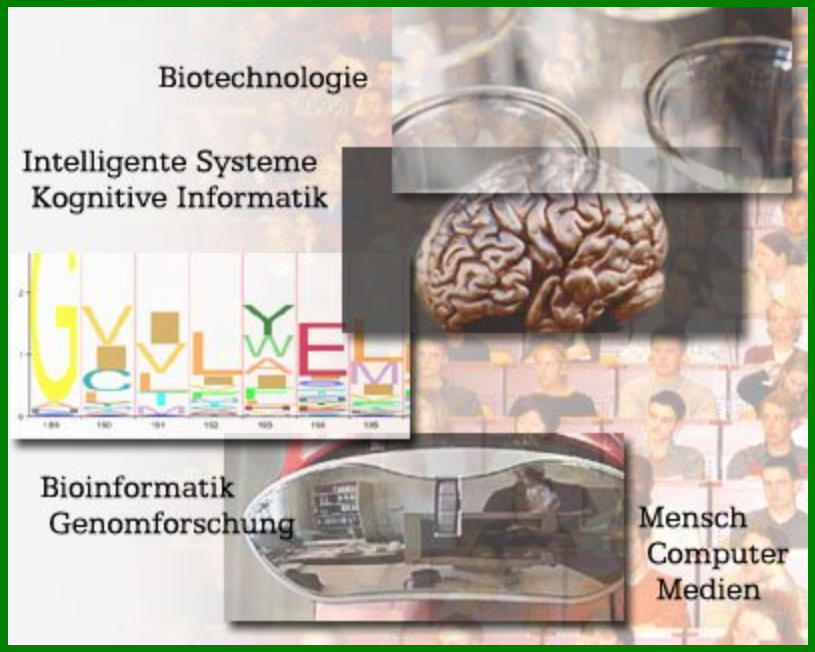

Biotechnologie
Intelligente Systeme
Kognitive Informatik
Bioinformatik
Genomforschung
Mensch
Computer
Medien

XML
Jenseits der spitzen Klammer
Konzepte und Anwendungsmethoden

Dr. Martin Fröhlich
Bielefeld
2007-06-19

Index

01 Was ist XML?

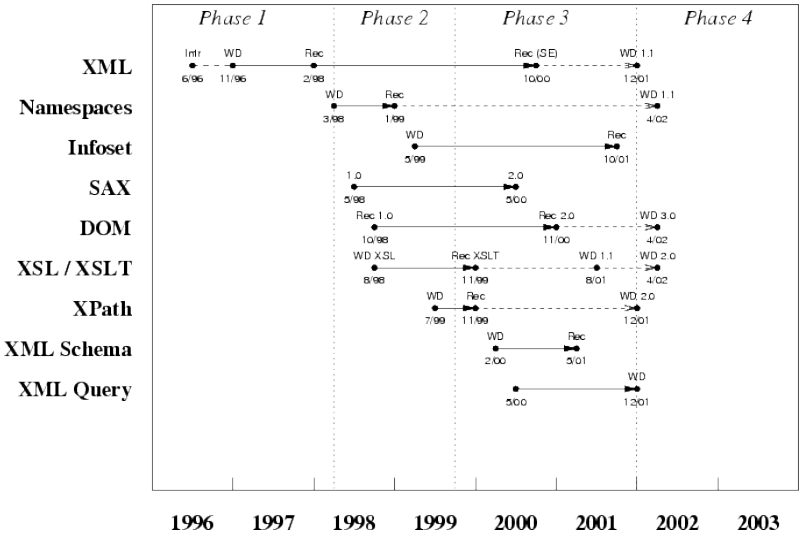
```
<orderRequestMessage
  xmlns="http://www.Telefonica.DE/markup/llu/aol/1.0/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  si:schemaLocation="http://www.Telefonica.DE/markup/llu/aol/1.0/ llu-aol.xsd"
  communicationDate="2005-02-11T17:10:00" schemaVersion="0.4" >
  <head>
    <answerChannel>
      <brokerName>localhost</brokerName>
      <brokerPort>80</brokerPort>
      <queue-topicName>test</queue-topicName>
    </answerChannel>
  </head>
  <orderID>
    <orderIdReseller>5000</orderIdReseller>
    <orderIdTD>10002</orderIdTD>
  </orderID>
  <naturalPerson>
    <salutation>Herr</salutation>
    <heraldic>Fuerst</heraldic>
    <honorific>Dr.</honorific>
    <givenname>Detlef</givenname>
    <surname-prefix>von</surname-prefix>
    <surname>Musterberg</surname>
  </naturalPerson>
  <billingAddress>
    <street>Musterweg</street>
    <houseNumber>42</houseNumber>
    <postalCode>12345</postalCode>
    <city>Musterhausen</city>
    <gemeindeKennziffer>110</gemeindeKennziffer>
  </billingAddress>
```

DAS!

```
  <bandwidth>
    <downStream>0</downStream>
  </bandwidth>
  <deliveryDate>
    2005-02-11T00:00:00
  </deliveryDate>
  <IPTNPhoneNumber>
    <cc>49</cc>
    <ndc>1234</ndc>
    <sn>29293</sn>
  </IPTNPhoneNumber>
  <IPTNPortingPhoneNumber>
    <cc>49</cc>
    <ndc>23112</ndc>
    <sn>3229929</sn>
  </IPTNPortingPhoneNumber>
  <phoneBookEntry>true</phoneBookEntry>
  <lineType>shared</lineType>
  <cpe>none</cpe>
  <services>
    <voIP>true</voIP>
    <data>false</data>
    <fastPath>false</fastPath>
  </services>
  <userName>toto</userName>
  <userId>100101</userId>
</orderRequestMessage>
```

Was ist XML?

- XML ist eine SGML-Applikation
  - Historisch hat XML seine Wurzeln in SGML (ISO 8879)
    - Daher gibt es DTDs (Document Type Descriptions) für XML-Dokumente
    - Deshalb wird XML auch für Text-Centric-Markup benutzt (DocBook, XHTML), also zur Annotierung von Textdokumenten bez. Struktur und Metadaten
    - Daher hat XML seine charakteristischen <>-Klammern



Was ist XML?

- XML ist eine externe Datenrepräsentation
  - Es können komplexe Datenstrukturen extern, z.B. auf Papier oder Festplatte, mit XML dargestellt werden
- XML ist ein universeller Datenstruktur Simulator
  - Es können Datenstrukturdefinitionen mit XML erstellt werden
  - Es kann überprüft werden, ob ein Dokument einer solchen Definition entspricht
- Hierzu kommt gleich noch mehr!

Was ist XML?

- Gemäß der theoretischen Informatik (formale Sprachen):
  - XML ist eine Sprache, deren Akzeptor heißt XML-Parser.
  - XML ist alles, was ein korrekter XML-Parser akzeptiert! (per def.)
  - XML ist eine Sprache in der Grammatiken (also Def. anderer Sprachen) geschrieben werden können (Uniformität)
    - Bekannt: XML-Schema, Relax-NG, Schematron ...
    - Es kann mit einem sog. validierendem XML-Parser überprüft werden, ob ein Dokument einer solchen Sprache entspricht (valide ist)

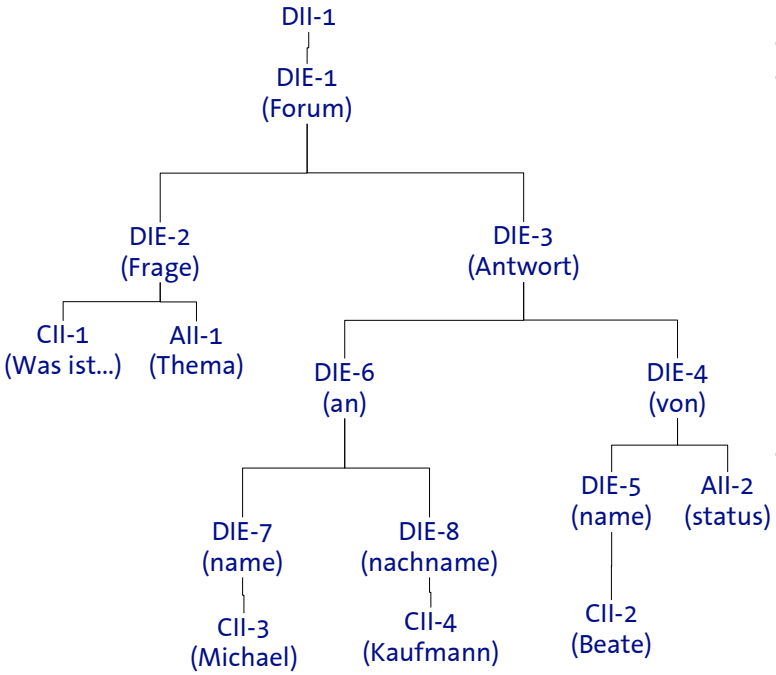
- Pragmatisch: XML ist, was dem XML-Standard entspricht
  - korrekt, aber minimal hilfreich...

Erfolgsfaktoren
Valide Dokumente

- wesentliche Erfolgsfaktoren für XML
  - Datenstrukturen formal zu definieren und deren Einhaltung zu validieren
  - Daten solcher formaler Strukturen zu serialisieren und damit extern zu repräsentieren
  - Serialisierte Daten wieder strukturiert (intern) einlesen zu können
  - Verfügbarkeit moderner Modellierungsmethoden (z.B. in XML-Schema)
    - Typisierung
    - Aggregation
    - Verfeinerung (Einschränkung)
    - Ableitung
    - Modularisierung

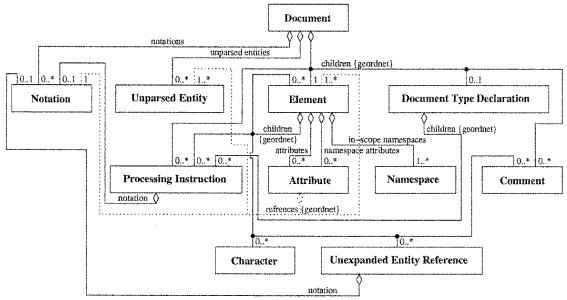
02 XML Datenmodell
Information Set (Infoset)

- Die Sprache XML hat ein eigenes Datenmodell, annotiert man die Elemente eines Dokuments gemäß dem Datenmodell, erhält man das sog. Infoset (nicht mit der XML Syntax verwechseln!)
- Obwohl im Kern baumförmig aufgebaut, beinhaltet es ausgedehnte Referenzmechanismen z.B. ID / IDREF




```
<?xml version="1.0" encoding="ISO-8859-15" standalone="yes"?>
<Forum>
  <Frage thema="XML">Was ist ein XML InfoSet?</Frage>
  <Antwort>
    <von status="keine Lehrerin">
      <name>Beate</name>
    </von>
    <an>
      <name>Michael</name>
      <nachname>Kaufmann</nachname>
    </an>
  </Antwort>
</Forum>
```

DIE = Document Information Element
DII = Document Information Item
CII = Character Information Item
AII = Atttibute Information Item



Datenmodell des Infosets

Infoset und DOM
Abgrenzung

- Das „Document Object Model“ (DOM) ist ein „Application Programmer's Interface“ (API), eine Programmierschnittstelle
- Das Infoset ist ein Datenmodell
- Das Datenmodell hinter dem DOM-API ist dem Infoset zwar ähnlich, beide sind jedoch *nicht* gleich:
  - DOM: DokumentFragment — IS: none
  - DOM: none — IS: Namespace (PSVI, kommt noch)
  - DOM: Entity — IS: none
  - DOM: Text — IS: Character
- Es gibt andere Mechanismen (APIs) mit denen man einen strukturierten Zugriff auf ein XML-Dokument erhält. Diese haben zum Teil eigene Datenmodelle, die teilweise fundamental vom Infoset abweichen.

Das 'reine' XML
Zusammenfassung

- XML fußt auf älteren, eingeführten Modellen wie SGML und wurde schrittweise erweitert und ausgebaut.
- Das bisher hier betrachtete umfasst 'reines' XML, d.h. ohne auf seine Möglichkeit einzugehen, andere Datenmodelle zu simulieren.
- Das das Datenmodell des Infoset ist das „hauseigene“ Datenmodell von XML

- Unsere Betrachtungen fokussieren sich im weiteren weniger auf die Annotation von Texten, als auf die Repräsentation von Daten und Datenstrukturen, auch wenn die Konzepte natürlich im ersteren Fall anwendbar bleiben.

03 Schema(s/ta)

Struktur in der Struktur

- Die Methode, um in XML eine neue Sprache/Datenstruktur einzuführen heißt „Schema“.
- Es gibt viele Möglichkeiten ein Schema zu beschreiben! Es muß nur formal validierbar sein, d.h. es muß eine formale, maschienenlesbare Form geben.
  - Die Anforderungen machen es sinnvoll Schemas ebenfalls in XML zu verfassen. Es bedarf dazu zwingend noch einer Beschreibung der Semantik (Bedeutung), die meißt natürlichsprachig (Spezifikationsdokument) erfolgt.
  - Die Theorie des Compilerbaus macht es erforderlich zur Konstruktion eines beweisbar korrekten Compilers (Parsers) eine Schema-Beschreibung immer in sich selbst zu beschreiben! (wie C-Compiler auch in C geschrieben sind)
- Bekannte Schema-Beschreibungen: XML-Schema, Relax-NG, Schematron

XML Schema vs. XML-Schema
Vermeidung von Mißverständnissen

- Es gibt eine Sprache/Schema zur Beschreibung von Schemas, genannt „XML-Schema“, genauer W3C XML-Schema.
- XML-Schema ist eine Möglichkeit XML Schemas zu beschreiben
- Es gibt ein XML Schema für XML-Schema, geschrieben in XML-Schema
- Es gibt auch eine **DTD** die XML-Schema negativ abgrenzt:
  - DTD akzeptiert, dann vielleicht XML-Schema (aber nicht sicher!)
  - DTD lehnt ab, dann definitiv nicht XML-Schema

- Alles klar? Auf das '-' kommt es an!
- XML-Schema wird im weitern als XML Schema Beschreibung eingeführt und genutzt.

Post-schema-validation Infoset (PSVI)

- Bei dem parsing-Vorgang eines XML-Dokuments gegen ein XML(-) Schema, können die Elemente des Infosets mit weiter Information annotiert werden. Z.B.:
  - Validität des Dokuments (z.B. am DII)
  - Typen von Elementen und Attributen
    - Randbedingungen
    - Schlüsselfelder
  - Referenzen zu dem Schema

- Nach dem parsen (validieren) weiß man mehr!

Der Datenstruktur Simulator
Begriffe

- Mit XML-Schema können Schema beschrieben werden.
  - Schema ≙ Datenstrukturspezifikation
  - Dokument ≙ Datensatz
  - XML-Schema ≙ Sprache zur Datenstrukturdefinition
  - Einige XML-Fragmente können ebenfalls Datensätze darstellen (s.u.)
- Damit Datensätze verschiedener Strukturen <u>zusammen verwendet</u> und dennoch <u>unterschieden</u> werden können, gibt es *Namespaces*.
  - Namespace identifiziert Zugehörigkeit zu Schema
  - Schema wird ein-eindeutig identifiziert durch URI (URL/URN)
  - Namespace wird ein-eindeutig bezeichnet durch URI

Achtung: UR**I** (⇒ Identifier), *keine* Download-Adresse, auch nicht bei URL

Datenstrukturen in XML
Eigenschaften

- Die Repräsentation einer Datenstruktur (Form des Datensatzes) ist nicht eindeutig!
  - Zwei verschiedene Formen können den gleichen Datensatz darstellen!
  - Korrektes Einlesen daher nur unter Einhaltung aller Regeln möglich
- Alle Regeln sind machinenlesbar (XML-Schema)!
  - Also lasst die Interpretation ein Programm erledigen (XML-Parser)
  - Warum: Folgende Dokumente sind identisch im Inhalt aber verschieden in der Form (äquivalent):

```
<?xml version="1.0"?>
<Book xmlns:lib="http://www.library.com">
   <lib:Title>Sherlock Holmes</lib:Title>
   <lib:Author>Arthur Conan Doyle</lib:Author>
</Book>
```

=

```
<?xml version="1.0"?>
<Book xmlns="http://www.library.com">
   <Title>Sherlock Holmes</Title>
   <Author>Arthur Conan Doyle</Author>
</Book>
```

Modellierung in XML-Schema

XML als Data Definition Language

- Elemente und Attribute werden OO-artig aggregiert, spezialisiert (Ableitung, constraint facet).
- Basistypen
  - boolean („true“/„false“), base64Binary, hexBinary, anyURI, language, normalizedString, string, token, byte, decimal, double, float, int, integer, long, negativeInteger, nonNegativeInteger, short, ..., date, dateTime, duration, time, ..., Name, NCName, NOTATION, Qname, ..., ID, IDREF, IDREFS, NMTOKEN, NMTOKENS
- Basisstrukturen
  - Choice, Sequence, Group

Modellierung in XML-Schema
strenge Typisierung

- complexeType bietet Aggregation
- simpleType bietet Spezialisierung durch Einschränkung

```
<xsd:complexType name="SubscriberLocationType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Describes a location with contact person/company description
      where the line is installed, with the name of the local
      contact person or company.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="city" type="CityType" />
    <xsd:element name="postalCode"
      type="PostalCodeType" />
    <xsd:element name="street" type="xsd:string" />
    <xsd:element name="houseNumber"
      type="HouseNumberType" minOccurs="0" />
    <xsd:element name="salutation"
      type="SalutationType" minOccurs="0" />
    <xsd:element name="givenName"
      type="GivenNameType" minOccurs="0" />
    <xsd:element name="surname"
      type="SurnameType" />
  </xsd:sequence>
</xsd:complexType>
```

```
<xsd:simpleType name="VersionType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Modeling protocol version according
      the std. interface/protocol
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d+(\.\d+){0,2}"/>
  </xsd:restriction>
</xsd:simpleType>
```

Modellierung mit XML Schemas
komplex aber mächtig

- XML-Dokumente können mehrere Schemas gleichzeitig benutzen!
  - Namespaces sind das wichtigste Modellierungswerkzeug über Schema-Grenzen hinweg!
  - Die Schemas müssen das erlauben!

```
<?xml version="1.0"?>
<c:camera xmlns:c="http://www.camera.org"
  xmlns:nikon="http://www.nikon.com"
  xmlns:olympus="http://www.olympus.com"
  xmlns:pentax="http://www.pentax.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.camera.org Camera.xsd">
  <c:body>
    <nikon:description>Ergonomically designed casing for easy handling</nikon:description>
  </c:body>
  <c:lens>
    <olympus:zoom>300mm</olympus:zoom>
    <olympus:f-stop>1.2</olympus:f-stop>
  </c:lens>
  <c:manual_adapter>
    <pentax:speed>1/10,000 sec to 100 sec</pentax:speed>
  </c:manual_adapter>
</c:camera>
```

Schema Design
Questionnaire

- Aus Instanzdokument Sicht
  - Sollen das Instanzedokument das Hinzufügen von Daten erlauben?
  - Welche Applikationen sollen das Dokument verarbeiten können?
  - Sollen Bestandteile das Dokuments variant (plug&play) sein?
  - Sollen Namespaces in dem Dokument versteckt sein?
- Aus Schema Sicht
  - Soll das Schema „standalone“, (unabhängig von anderen Sch's) sein?
  - Soll das Schema für häufige Modifikationen entworfen werden?
  - Soll das Schema „sortenrein“ sein
    - wird also nur eine der Schema-Sparchen wie XML-Schema, Relax-NG, Schematron Anwendung finden, oder mehrere?

Methoden der Modellierung
caveat architecton

- Die Sprachmerkmale vom XML, XML Schema wie auch XML-Schema sind sehr mächtig
- Wenn man sie geschickt auswählt, kann man verschiedene Ziele für Schema-Dokument wie für die Instanzdokumente verfolgen
- Vorsicht vor Kochrezepten!
  - Jedes Sprachmittel hat seinen Anwendungsfall
  - „Es immer so und so zu machen“ ist keine Lösung!

- Näheres:
  - Handouts
  - http://www.xfront.com/BestPracticesHomepage.html

„Best Practices“

http://www.xfront.com/BestPracticesHomepage.html

- Sehr hilfreiche Zusammenstellung von Methoden und deren jeweiligen Vor- und Nachteilen
  - Liegt zum Teil im Handout vor
- Da einige der Methoden komplex sind, empfiehlt sich folgendes Herangehen:
  - Jedes Kapitel erst einmal nach den verwendeten Sprachmitteln (Abstract Type, Constraint Facets, etc.) durchgehen, und ihre Syntax und Sematik nachschlagen
  - Dann erst die Methode mit ihren Vorbedingungen und Nebenwirkungen analysieren
- Referenzen:
  - http://www.w3.org/XML/Core/#Publications
  - XML in a Nutshell, O'Reilly, ISBN: 0-596-00292-0

04 Andere Datenrepräsentationen

„the whole enchilada“

- Das relationale Datenmodell (Codd, C-ACM 13(6),377-387 (1970))
  - Datentabellen, deren Zeilen durch Referenzen (Schlüssel) in Relation gesetzt werden (Math.: Relation m:n, Funktion 1:1)
  - Das relationale Modell erfüllt die Bedingungen einer math. Algebra, d.h. man kann darin „rechnen“ und Korrektheit von Algorithmen beweisen
  - Wichtigste Operationen: *projection* (Auswahl Spalte), *selection* (Auswahl Zeile), *union* (Zsmmnfssng Zeile), *join* (Zsmmnfssng Spalte)
- Objektorientiertes Modell
  - Hierarchische Datentypen
  - Dedizierte Methoden auf den Datentypen
- Funktionales Modell
  - aus der Theorie: Abstrakter Datentyp: Grundmenge/Axiome

Zugriff auf Daten im XML Instanzdokument
nicht konvertieren sondern transformieren

- Durch Parsen DOM/SAX in interne Datenstruktur überführen, Such- und Selektionsmechnismen der verwendeten Sprache (Java/C/C++/Perl/...) verwenden
  - Führt das Problem auf ein bekanntes zurück (+)
  - Generell ineffizient und kompliziert (-)
- Verwendung von XML-Werkzeugen (XSLT/XPath/XQuery)
  - Standardisierter Zugriff (+)
  - neue Technik (o)
  - XPath: Basis von XSLT und XQuery
    - Pfadausdruck zum wandern im Elementbaum:
      `//movie/title[contains(.,"007")]`
  - XSLT: schablonenhafte Ersetzung: XML→XML, XML→nicht-XML
  - XQuery: FLWOR-Abfrage (SQL-ähnlich): XML→XML, nicht-XML→XML

Zugriff auf Daten in Datenbank
z.B. Oracle RDBMS 10g/9i

- Externe Programmbibliotheken Oracle-XSU (Java/C++) (im XDK)
  - eigentliches SQL ändert sich nicht
  - XML-Wandlung findet client-side statt
  - herstellerspezifisch
- SQL/XML: Bestandteil des SQL-Standards
  - ISO/IEC 9075-14:2005 (E): Information technology – Database languages – SQL – Part 14: XML-Related Specifications (SQL/XML)
  - Enthält XML-Datentyp
  - Enthält XQuery
  - „scherdding“ (verschiedene Spalten) und „storing“ (VARCHAR/CLOB)
  - Schemas können DB-seitig validiert werden

XML speichern

- Da man auch ein ER-Schema in einem XML Schema simulieren kann, kann man auch XML-Daten in RDBMS-Tabellen speichern
  - Das XML-Dokument wird zerrissen, sein Inhalt bleibt erhalten (schredding)
- Ein XML-Instanzdokument ist Text und kann als solcher Behandelt werden (CLOB)
- Es können in SQL die XML-Datentypen und Methoden benutzt werden
  - Mischungen aus „schredding“ und „storage“ möglich
  - Typsicherheit / Schemavalidierung
- Die Frage ist:
  - Was liegt schon vor? Was kann man gerade designen?

XML speichern
Oracle

- Das RDBMS bietet die Möglichkeit in dem XML-Schema Instanzdokument, das das XML Schema darlegt, durch das einbinden eines eigenen Schemas die „Schredding-Abbildung“ zu spezifizieren.
  - Das Beste aus beiden Welten
- XML-DB-Features erlauben XML-Werte in SQL (SQL*plus)
  - SQL:

```
SELECT XMLQuery(
   'for $c in ora:view("My_very_Table")
    return $c' RETURNING CONTENT)
FROM dual;
```

  - Hinweis: Bei *Abfrage* nur Std.-Funktionen verwenden, vielleicht mit Ausnahme von ora:view.
- Nachteil der Std.-Funktionen: *Update* noch nicht standardisiert!
  - Oracle hat aber den lead in dem Standardisierungsprozeß, daher wahrscheinlich unkritisch

SQL/XML
XML und SQL aus einer Hand

- SQL:2003:Die Funktionen sind:
  - XMLELEMENT, XMLATTRIBUTS, XMLAGG, XMLFOREST, XMLCONCAT
  - Alle zur Wandlung von SQL-Tabellen in XML-Strukturen
- SQL:2006: XML Type:
  - XML(DOCUMENT), XML(CONTENT), XML(SEQUENCE)
- SQL:2006: XQuery:
  - XMLQUERY: XQuery für SQL-SELECT-clause ⇒ XML
  - XMLEXISTS: XQuery für SQL-FROM-clause ⇒ gemäß SELECT-clause
  - XMLTABLE: XML nach SQL
- Oracle: (10g unterstützt SQL:2006)
  - ora:view: SQL-Funktion zur kanonischen Wandlung ganzer Tabellen in XML, arbeitet wie die XSU, nur database-seitig; UPDATEXML

05 „Der Witz dabei“

- XML nicht andauernd wandeln, sondern in XML transformieren und abfragen
- Aus DB direkt in XML, minimale „Adapter“
- Zugriff über XML-Prozessor (Oracle, Saxon, DataDirect etc.)
  - Anfragen mittels XPath/XQuery
  - Updates über (noch properitäre Erweiterungen zu) XPath/XQuery
  - Umstrukturierung durch XSLT/XQuery
- Versenden direkt durch JMS oder REST
- ⇒ Datenmodellierung in einer „Welt“ (XML)
  - Leichter portierbar und migrierbar
  - geschlossene Darstellung, weniger Medienbrüche

XQuery
Beispiele

- Anfrage

```
<antwort>
  for $m at $i in movies/movie
  where $i > 1
  order by $i/yearReleased ascending
  return data($m/@myStars)
</antwort>
```

- Antwort

```
<antwort>3 4</antwort>
```

```
for $i in doc("items.xml")//item_tuple
let $b := doc("bids.xml")//bidTuple[ino = $i/ino]
where contains($i/description, "Bicycle")
order by $i/ino
return
  <item_tuple>
    { $i/ino }
    { $i/description }
    <high_bid>
      { max($b/bid) }
    </high_bid>
  </item_tuple>
```

- Hinweis:
  - XQuery „let“-Operator zur Definition von Variablen ist wie OO-„new“:
    let $a = <e>42</e>, $b = <e>42</e>
    $a is $b ⇒ false, da gleicher Wert, aber verschiedene Knoten!

SQL/XML und XQuery
Was, Wo und Wann?

- Es gibt neben den Standards, die beschreiben was XQuery ist, verschieden Implementationen (engines) in verschiedenen Anwendungen
  - Java, client-side: Oracle XSU, vergleichbar mit JDBC
  - Java: XQJ: XML/XQuery für JDBC aufkommender Standard
  - SQL, database-side: SQL/XML: liefert Umsetzung von XML ⇔ SQL
  - CLI, client-side: SQL*plus hat eigenes 'xquery' commando
- Links
  - Oracle: http://www.oradev.com/xml_functions.jsp
  - DataDirekt: http://www.datadirect.com/developer/xquery/index.ssp
  - Berkeley Labs: Nux, http://dsd.lbl.gov/nux/
  - Oracle: Oracle® XML DB Developer's Guide
    http://download-west.oracle.com/docs/cd/B19306_01/appdev.102/b14259/toc.htm