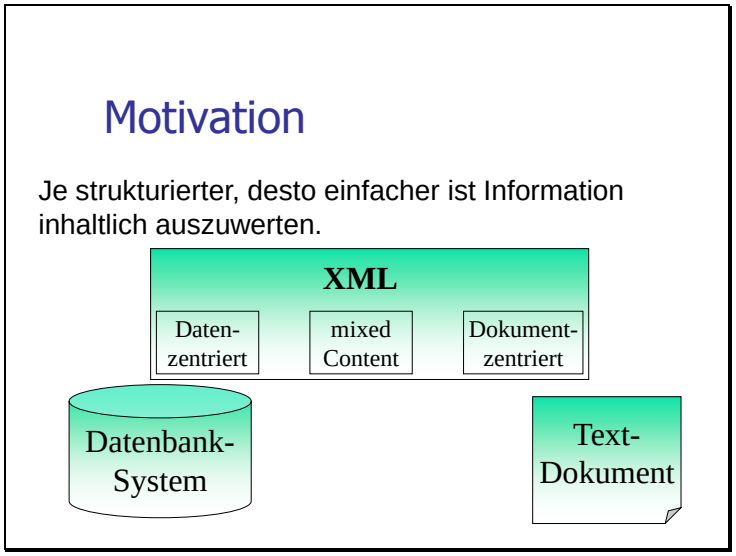
XML Einführung

1 Motivation



XQuery, XLink, Namespace, DTD, Schema, Xsl, XHtml, … Wenn Xml für Sie neu ist, wissen Sie womöglich kaum, wo sie anfangen sollen. Die vorliegende Zusammenfassung versucht eine Orientierung zu bieten.

Ziel

- Umgang mit Xml lernen
- Xml primär als Informationsmodell begreifen
- Xml als Grundlage für viele Anwendungen sehen

Neue Entwicklung

- XML ist eine der wichtigsten Entwicklungen der letzten Jahre in der Informatik.
- XML ist ein Universalkonzept für **Datenmodellierung**, **Datenspeicherung** und **Datenaustausch**.
- Keine andere Technologie hat in so kurzer Zeit einen solchen Rush von neuen Tools ausgelöst.
- XML gehört heute zu den Grundfertigkeiten, die ein Informatiker im Umgang mit Daten beherrschen sollte.

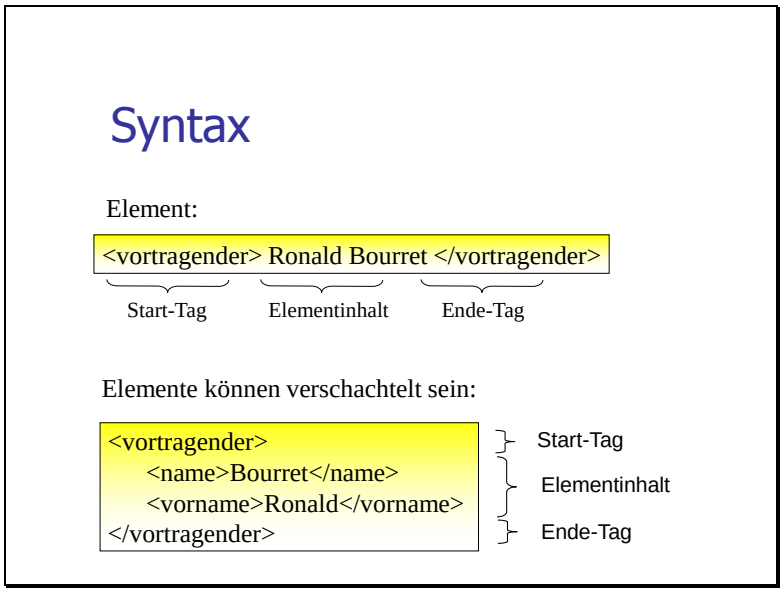
Anwendungen

Um die Vielseitigkeit von XML aufzuzeigen, sind hier ein paar mögliche Anwendungsbeispiele aufgelistet:

- Arbeits-Skript eines Elektrotechnikers
- Konstruktionszeichnungen
- Musikalische Kompositionen
- Theaterstücke
- Biochemische Prozesse
- Web-Anwendungen

Alles, was benenn- und beschreibbare Strukturen aufweist, lässt sich mit XML erfassen. Wie man diese Daten visualisieren, abspielen oder anderweitig verarbeiten kann, ist damit nicht festgelegt. Es geht zunächst nur mal darum, Daten sinnvoll zu strukturieren und vollständig zu beschreiben.

2 Syntax



Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CONTACTLIST SYSTEM "contacts.dtd">

<CONTACTLIST>
        <CONTACT>
                <NAME>Kurt Hurter</NAME>
                <PHONE type="mobile">079 800 43 56</PHONE>
                <PHONE type="private">044 837 43 00</PHONE>
        </CONTACT>
        <CONTACT>
                <NAME>Angelo Casanova</NAME>
                <ADDRESS>In der Mühle 71, 8330 Pfäffikon</ADDRESS>
                <PHONE type="fax">079 800 43 56</PHONE>
        </CONTACT>
</CONTACTLIST>
```

Interpretation

- Die erste Zeile beinhaltet **Versionsangabe und Zeichensatz**. Diese Zeile ist obligatorisch.
- Die zweite Zeile ist die Definition eines **Dokumenttyps zur Validierung**. Diese Zeile ist fakultativ.
- **Elemente** sind die grundlegenden Informations-Bausteine von Xml.
- Elemente sind in **Tags** eingegrenzt.
- Ein Start-Tag <xxx> muss immer mit einem End-Tag </xxx> abgeschlossen sein.
- Ein Tag kann mit **Attributen** versehen sein. Im Beispiel besitzt das Tag PHONE ein Attribut mit dem Namen type und kann die Werte mobile, private und fax annehmen.
- Attribute stellen einem Element Zusatzinformationen zur Verfügung. Attribute stehen immer im Start-Tag.
- Xml Dokumente sind **hierarchisch** (baumartig ist ein anderer Begriff).
- Ein übergeordnetes Element steht mit einem untergeordneten Element in einer **1 : 0,N** Beziehung.

3 Xml Parser

Parser

SAX

- einfacher Zugriff
- für einfach strukturierte Dokumente
- auch geeignet für sehr große XML-Dokumente
- geeignet, wenn Zugriff nur auf einzelne Elemente

DOM

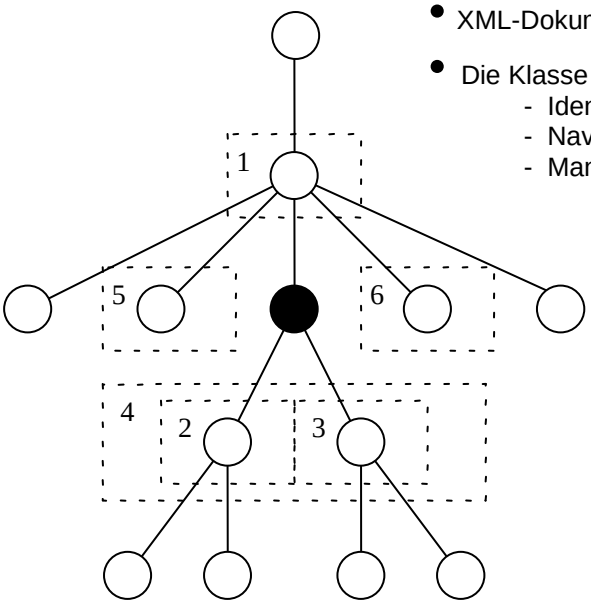
- Navigation durch Struktur
- auch für komplexe Strukturen geeignet
- geeignet für Manipulation der Struktur
- auch Speicherungstruktur
- Probematisch für grosse Dokumente

Idee

- Parser machen den Inhalt eines Xml-Dokuments für eine Applikation verfügbar.
- Standardisierte Interfaces für zahlreiche Programmiersprachen sind vordefiniert.
- Parser können validierend / nicht validierend in Bezug auf DTD oder Xml-Schema sein.
- Zwei konkurrierende Konzepte SAX und DOM stehen zur Auswahl.

SAX (Simple API for Xml)
SAX ist eine Programmschnittstelle für die Verarbeitung eines Xml-Dokuments. SAX liefert ein Xml-Element nach dem andern in einem Eingabestrom.

DOM (Document Object Model)
DOM liefert eine komplette Baumstruktur aller Objekte. Eine Applikation kann durch diese Struktur navigieren und sowohl Daten als auch Struktur manipulieren.



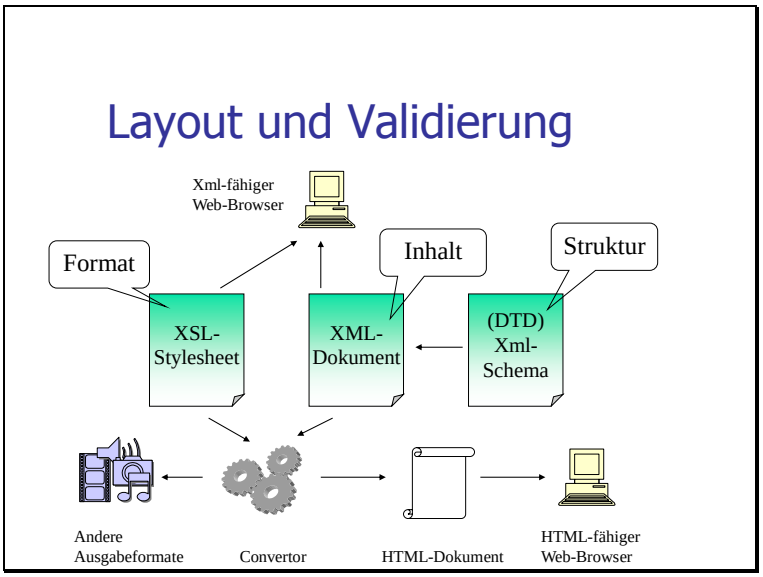
- XML-Dokument als DOM dargestellt
- Die Klasse Node enthält Methoden zur
  - Identifikation
  - Navigation
  - Manipulation

Navigation

Ausgehend vom diesem Knoten ● liefern folgende Methoden() die Knoten (-> nr) als Ergebnis:

```
getParentNode()      -> 1
getFirstChild()      -> 2
getLastChild()       -> 3
getChildren()        -> 4
getPreviousSibling()-> 5
getNextSibling()     -> 6
```

4 Format, Inhalt und Struktur



Layout
Um den Inhalt eines Xml-Dokuments in einem Web-Browser darzustellen, braucht es eine Vorschrift, wie die Daten darzustellen sind. Diese Vorschrift wird typischerweise mit einem XSL-Stylesheet verfasst.

Ein Browser findet innerhalb der Xml-Datei einen Verweis auf das zugehörige Stylesheet. z.B:
<?**xml-stylesheet** type = „text/xsl“ href = “contact.xsl”?>
Damit ist ein Web-Browser in der Lage die Informationen korrekt darzustellen.

Daneben gibt es diverse SW-Tools (Converter), die ein Xml-Dokument aufgrund einer XSL(T)-Vorschrift in ein anderes Format umwandeln können.

Validierung
Mit Hilfe einer Validierung wird die Struktur von Xml-Dokumenten zentral und verbindlich festgelegt. Alle Xml-Dokumente desselben Typs können anhand derselben Vorgabe überprüft werden. Das ist hilfreich, weil viele Xml-Editoren und andere Tools eine solche Prüfung auf Wunsch automatisch vornehmen können. Auf diese Weise werden Sie als Autor oder als Benutzer sofort auf allfällige Strukturverletzungen aufmerksam gemacht.

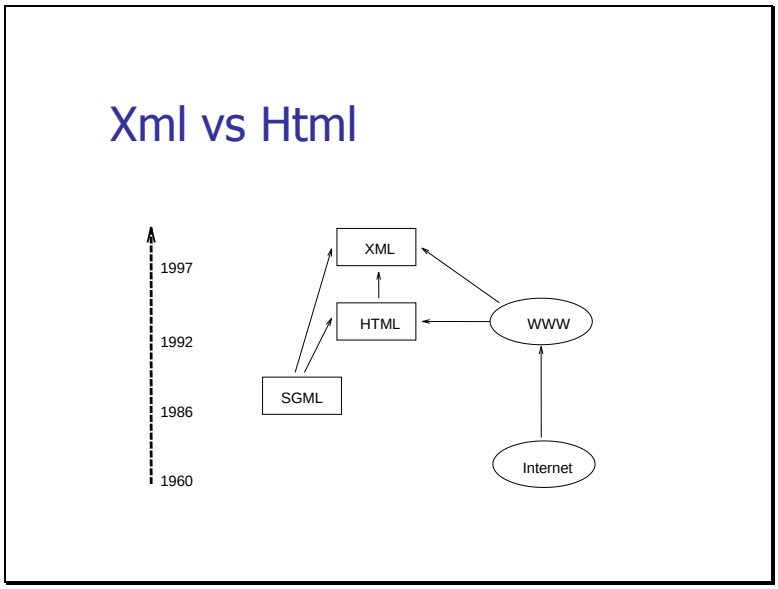
DTD (Document Type Definition) ist ein solcher einfacher Validierungsstandard.
Xml-Schema ist ein neuerer und mächtigerer Standard.
Xml-Schema und Xml-Dokumente stehen zueinander wie eine Klasse zu seinen Instanzen.

Diese Validierung ist sehr hilfreich beim Datenaustausch über mehrere Applikationen und vor allem bei **B2B Anwendungen**. In solchen Fällen legt ein Xml-Schema das gemeinsame Format eindeutig fest.

Die verarbeitenden Programme sind einfacher zu schreiben und können effizienter ablaufen, wenn sie sich darauf verlassen können, dass das Xml-Dokument syntaktisch korrekt ist.

Trotz der erwähnten Vorteile ist die Benutzung von Xml-Schemata **nicht obligatorisch**. Xml-Dokumente lassen sich auch ohne Validierung erstellen und verarbeiten.

5 Xml versus Html



Xml- und Html-Dokumente sehen sich sehr ähnlich. Beides sind Auszeichnungs-Sprachen, doch bestehen bedeutende Unterschiede:

Beschreibung vs Darstellung

- Xml wurde entwickelt um Daten zu beschreiben. Es existieren keine Tags mit fest vorgegebener Bedeutung. Xml benutzt Tags zur Abgrenzung von Daten und überlässt die Interpretation der Daten vollkommen der Anwendung, die sie verarbeitet.
- Html wurde entwickelt um Daten darzustellen. Jeder Web-Browser kennt alle Html-Tags und kann daher Html-Dokumente entsprechend darstellen.

Erweiterbarkeit

- Xml ist erweiterbar. Jedes einzelne Tag müssen Sie selber erfinden.
- Html ist nicht erweiterbar. Die Bedeutung aller Tags sind vorgegeben.

HTML Code:

```
<p>
    <b>Mrs McCoon</b>
    <br>
    1401 Main Street
    <br>
    Anytown, NC 34829
</p>
```

im Web-Browser dargestellt:

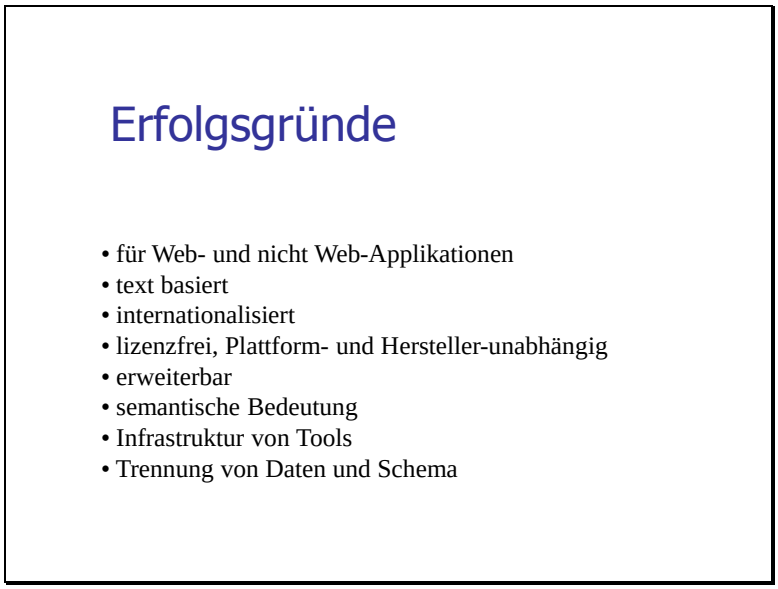
Mrs McCoon
1401 Main Street
Anytown, NC 34829

Das gleiche Beispiel in Xml Code:

```
<Address>
    <Name>Mr McCoon</Name>
    <Street>1401 Main Street</Street>
    <City>Anytown</City>
    <State>NC</State>
    <Zipcode>34829</Zipcode>
</Address>
```

So ist es für eine Applikation einfach zu verstehen, dass 34829 eine Postleitzahl bedeutet.

6 Xml Erfolgsgründe



Xml ist geeignet für Web- und nicht Web-Applikationen
Xml ist ein generelles Daten-Repräsentations-Format und kann in jeder Art von Applikation eingebunden sein.

Xml ist textbasiert
Dies war eine bewusste Entscheidung der Xml-Entwickler. Dadurch können Xml-Dokumente nicht nur mit jedem einfachen Text-Editor erstellt werden, sondern Fachleute können so einfacher debuggen und allenfalls Fehler korrigieren. Zudem ist ein Text-File firewall friendly.

Xml ist internationalisiert (Unicode)
Unicode bietet universellen Zeichensatz für Lateinisch, Griechisch, Arabisch, Hebräisch, Chinesisch, Japanisch, technisch und mathematisch-naturwissenschaftliche Symbole, etc.

Xml ist lizenzfrei, Hersteller-, HW- und SW-unabhängig
Xml ist eine Entwicklung vom World Wide Web Consortium. Das W3C besteht aus mehr als 400 Mitgliederorganisationen und kümmert sich um die Weiterentwicklung des WWW sowie deren zugehörigen Standards wie Html, Http, etc. Alle W3C-Entwicklungen sind gratis und an keine Hersteller gebunden.

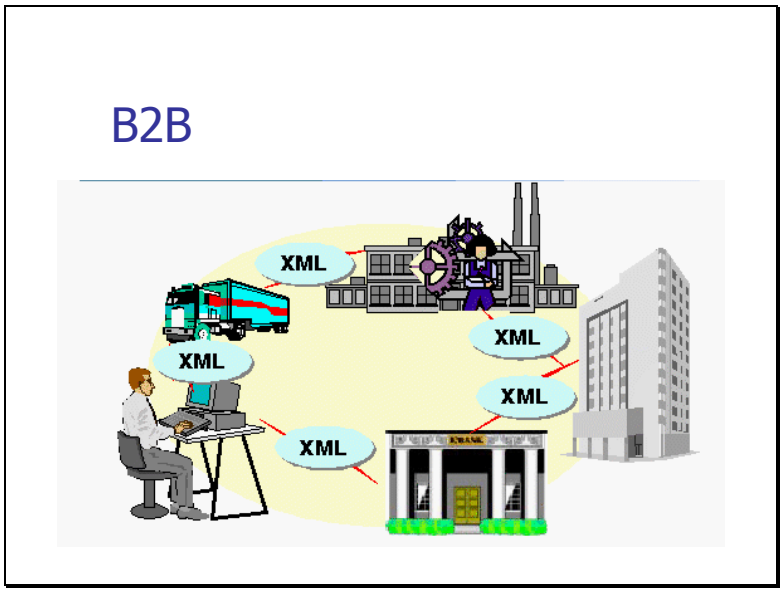
Xml ist erweiterbar
In Xml sind keine Tags vordefiniert. So müssen für jede Art von Information die korrekte Struktur und passende Tags selber definiert werden.

Semantische Bedeutung
Mit den Tags wird implizit die semantische Bedeutung der Daten mitgegeben.

Xml ist eine Infrastruktur von Tools
Xml ist nicht nur eine Syntax, sondern eine Familie von Technologien: Xml Information Set, Xml Schema, Xml Query, XLink, XPointer, Xml Forms, Xml Encryption, Xml Signature, …

Trennung von Daten und Schema
Philosophie Relationales Modell: schema first then data
Philosophie von Xml: Daten können existieren ohne Schema
Die Möglichkeit Daten zu definieren, ohne dass man schon genau wissen muss, was man will, führt zu schnelleren und iterativeren Entwicklungen. Man fängt mal an …

7 Business to Business Datenaustausch



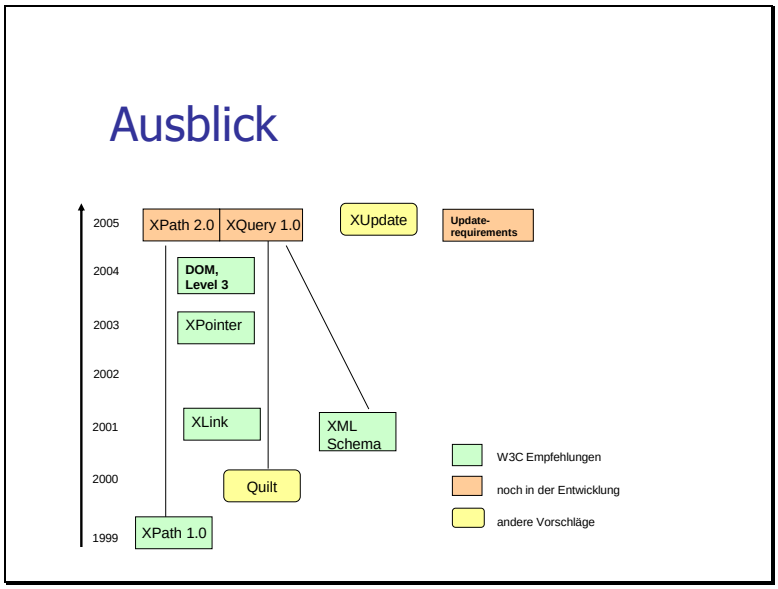
Weltweiter Datenaustausch
Das relationale Modell ist für den Datenaustausch zwischen Applikationen völlig ungeeignet. Hier liegt aber der **kommerziell grösste Erfolg von Xml**. Ein Hersteller möchte z.B. bei seinen Lieferanten Rohprodukte von definierter Anzahl und in geeigneter Qualität elektronisch bestellen. Verschiedene Firmen arbeiten aber selten mit den gleichen Anwendungen. Datenbanken und Applikationen speichern ihre Daten in inkompatiblen Formaten. Eine der zeitaufwändigsten Herausforderungen für Entwickler sind der Austausch solch inkompatibler Daten übers Internet. Die Kommunikation via Xml-Daten kann diese Komplexität stark reduzieren.
Viele solch interessanter Xml-B2B-Anwendungen sind weltweit in Entwicklung.

Xml und der Software-Markt
Für den klassischen Software-Markt stellt Xml eine gewaltige Provokation dar. Denn es ist ein Generalangriff auf alle proprietären Dateiformate, die an bestimmte, oft teure Programme gebunden sind.
Durch Xml entsteht aus Anwendersicht die Erwartung, dass die Daten, die zu erzeugen oder zu verarbeiten sind, in einem modernen, Xml-basierten Dateiformat abgespeichert sind. Der Wettbewerb zwischen konkurrierenden Software-Produkten verlagert sich dadurch: Die Software-Hersteller müssen dann durch überlegene Funktionalität und clevere Benutzerführung ihre Kunden an sich binden. Die Kundenbindung durch proprietäre Dateiformate fällt weg.

Trennung von Daten und Layout
Ein wichtiger Pluspunkt von Xml ist, dass Daten und Layout vollständig getrennt sind. Nicht einmal die Ausgabeform ist festgelegt. Aus dem gleichen Xml-Datenbestand lässt sich eine Webseite generieren, eine Druckvorlage oder eine akustische Ausgabe mit künstlichen Stimmen. Es gibt nur noch eine Datenquelle, und das Publizieren erfolgt mit Hilfe von Konvertern, die Xml-Daten in andere Daten (HTML, Postscript, etc.) übersetzen. Die klassischen, zuletzt immer monströser gewordenen Applikationen mit ihren hauseigenen Formaten werden zunehmend kleineren Software-Modulen weichen, deren Stärke das Zusammenspiel ist.

Das Konzept dieser Modularisierung ist zwar schon älter, aber bisher fehlten die Schnittstellen. Xml jedoch leistet eine solche **Schnittselle auf der Ebene der Datenformate**.

8 Ausblick



Eine ganze Reihe von Tools / Technologien / Spezifikationen sind in Entwicklung:

XLink
Hyperlink auf andere Xml-Dokumente

XPointer
Hyperlink, der nicht auf ganze Xml-Dokumente verweist, sondern nur auf einzelne Elemente.

XPath
XPath ist eine Sprache zur Adressierung von Dokumentbestandteilen.

XQuery
Abfragesprache für Xml-Dokumente, ähnlich wie SQL für RDBS.

XUpdate
Sprache um Daten und Struktur von Xml-Dokumenten zu verändern.

m : n Beziehungen
Xml ist grundsätzlich ein hierarchisches Modell, sprich es repräsentiert 1 : n Beziehungen. Analog zur Primärschlüssel - Fremdschlüssel - Beziehung im Relationalen Modell lassen sich aber mit ID und IDREF bei DTD (bzw. KEY und KEYREF bei Xml Schema) m : n Beziehungen darstellen.

.NET
Xml ist ein zentraler Bestandteil der .NET-Laufzeitumgebung. Dem Entwickler werden eine Reihe von Klassen bereitgestellt, um die Erzeugung und Manipulation von Xml-Daten zu vereinfachen.

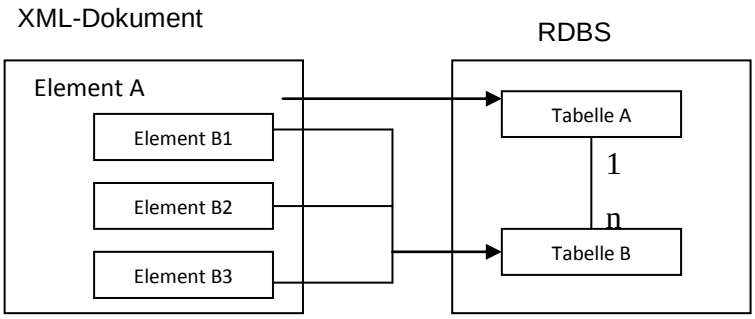
Epilog
Vieles ist noch in Entwicklung, Xml ist in aller Munde und wird oft noch als Hype (miss)verstanden. Derzeit wird noch behauptet, Xml sei everywhere, for everybody, for every application und demzufolge die ultimative Lösung für alle (bislang ungelösten) Probleme. Später wird Xml vielleicht wie die KI oder gar Expertensysteme verteufelt werden. Die Wahrheit wird wahrscheinlich irgendwo in der Mitte liegen.

9 XML Datenbanken

Mit etwas Goodwill kann man ein XML-Dokument schon als eine Datenbank betrachten. So ein Dokument enthält Daten in einer selbstbeschreibenden und portierbaren Form und hat, mit den entsprechenden Erweiterungen, weitere Eigenschaften von Relationalen Datenbanken wie Schemas (DTD, Xml Schema), Abfragesprachen (XQuery, XPath) und Programmierschnittstellen (DOM).
Aber der Datenzugriff ist aufgrund des Parsens des Dokuments langsam, Daten sind nicht indexierbar und es existieren keine Sicherheits- und Transaktionsmechanismen. Hier aber liegen die Stärken von relationalen DBS. Ein logischer Schritt ist daher die XML-Dokumente in die bestehenden relationalen DB-Systeme zu integrieren.

XML-Schnittstellen von RDBS

- Am einfachsten ist es, die XML-Dokumente als BLOB-Datentypen abzulegen. Damit können aber serverseitig keine Abfragen innerhalb des Xml-Blobs gemacht werden.
- Etliche DBS bieten Xml als eigenständigen Datentyp an. Mit entsprechenden Abfragesprachen können einzelne Xml-Elemente abgefragt werden.
- Bei der tabellenbasierten Abbildung von XML-Dokumenten (Shredding) werden die Daten jeder Elementebene in einer eigenen Tabelle abgelegt und die Tabellen untereinander mit Foreign Keys referenziert, siehe Abbildung.



Native XML-Datenbanken
Ein Beispiel für eine vollständig native XML-Datenbank ist Tamino von Software AG. Welche Prinzipien zur Datenverwaltung zu Grunde liegen, behält die Firma für sich. Man kann aber wohl davon ausgehen, dass ein erheblicher Teil aus dem Adabas-Know-how (hierarchisches DBS) in die Entwicklung eingeflossen ist. Abfragen können sowohl dokumentübergreifend als auch auf Dokumentebene durchgeführt werden.

10 Übungen

6.1.a Xml-Daten mit C# / .Net erzeugen

Diese Übung wurde mit Visual Studio 2003 erstellt.
Für das Erzeugen von Xml-Dateien bietet sich die Klasse **XmlTextWriter** an.

Aufgabe:
- Kreieren Sie ein C# Konsole-Projekt und übernehmen Sie die Methode WriteXmlFile();
- Der Namespace System.Xml muss bekannt sein.

```
using System.Xml;

static void Main(string[] args)
{
        WriteXmlFile();
}


static void WriteXmlFile()
{
        string fileName  = @"c:\Personen.xml";
        XmlTextWriter tw = new XmlTextWriter(fileName, null);
        tw.Formatting    = Formatting.Indented;

        tw.WriteStartDocument();
        tw.WriteComment("Dies ist Entenhausen");

        tw.WriteStartElement("Liste");
        WritePerson(tw, "Donald",   "Duck", "Herr");
        WritePerson(tw, "Daisy",    "Duck", "Frau");
        WritePerson(tw, "Dagobert", "Duck", "Herr");
        tw.WriteEndElement();

        tw.Flush();
        tw.Close();
}


static void WritePerson(XmlTextWriter tw, string vName, string nName, string anrede)
{
        tw.WriteStartElement("Person");
        tw.WriteElementString("Vorname",  vName);
        tw.WriteElementString("Nachname", nName);
        tw.WriteElementString("Anrede",   anrede);
        tw.WriteEndElement();
}
```

Kompilieren und starten Sie das kleine Programm.

Öffnen Sie das entstanden File 'c:\Personen.xml' mit dem Internet-Explorer:

```
<?xml version="1.0" ?>
<!-- Dies ist Entenhausen -->
<Liste>
        <Person>
                <Vorname>Donald</Vorname>
                <Nachname>Duck</Nachname>
                <Anrede>Herr</Anrede>
        </Person>
        <Person>
                <Vorname>Daisy</Vorname>
                <Nachname>Duck</Nachname>
                <Anrede>Frau</Anrede>
        </Person>
        <Person>
                <Vorname>Dagobert</Vorname>
                <Nachname>Duck</Nachname>
                <Anrede>Herr</Anrede>
        </Person>
</Liste>
```

6.1.b Xml-Daten mit C# / .Net lesen

Das SAX-Programmiermodell ist ein Push-Modell, d.h. eine Xml-Datei wird durchgescannt und ein Ergebnis in die Applikation gepushed, sobald ein Element erkannt wird.

Unter .Net werden die Klassen XmlReader und deren Ableitungen XmlTextReader, XmlNodeReader und XmlValidatingReader angeboten, die dem SAX-Programmiermodell sehr ähnlich sind, aber ein Pull-Modell darstellen. Bei diesem Pull-Modell können Daten selektiv in die Applikation geholt werden.

Aufgabe:
Erweitern Sie die vorhergehende Applikation um die Methode ReadXmlFile().

```
static void Main(string[] args)
        {
                WriteXmlFile();
                ReadXmlFile();
        }


        static void ReadXmlFile()
        {
                string fileName  = @"c:\Personen.xml";
                XmlTextReader tr = new XmlTextReader(fileName);

                while (tr.Read())
                {
                        if (( tr.NodeType == XmlNodeType.Element ) &&
                           ( tr.Name != "Liste") &&
                           ( tr.Name != "Person"))
                        {
                                Console.WriteLine(tr.ReadElementString());
                        }
                }
                System.Threading.Thread.Sleep(5000);
        }
```