XML : Grundstein zum grenzenlosen Datenaustausch

Autor(en): **Jung, Frank**

Objekttyp: **Article**

Zeitschrift: **Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association Suisse des Electriciens, de l'Association des Entreprises électriques suisses**

Band (Jahr): **91 (2000)**

Heft 9

PDF erstellt am: **24.05.2024**

Persistenter Link: https://doi.org/10.5169/seals-855542

Nutzungsbedingungen

Die ETH-Bibliothek ist Anbieterin der digitalisierten Zeitschriften. Sie besitzt keine Urheberrechte an den Inhalten der Zeitschriften. Die Rechte liegen in der Regel bei den Herausgebern.
Die auf der Plattform e-periodica veröffentlichten Dokumente stehen für nicht-kommerzielle Zwecke in Lehre und Forschung sowie für die private Nutzung frei zur Verfügung. Einzelne Dateien oder Ausdrucke aus diesem Angebot können zusammen mit diesen Nutzungsbedingungen und den korrekten Herkunftsbezeichnungen weitergegeben werden.
Das Veröffentlichen von Bildern in Print- und Online-Publikationen ist nur mit vorheriger Genehmigung der Rechteinhaber erlaubt. Die systematische Speicherung von Teilen des elektronischen Angebots auf anderen Servern bedarf ebenfalls des schriftlichen Einverständnisses der Rechteinhaber.

Haftungsausschluss

Alle Angaben erfolgen ohne Gewähr für Vollständigkeit oder Richtigkeit. Es wird keine Haftung übernommen für Schäden durch die Verwendung von Informationen aus diesem Online-Angebot oder durch das Fehlen von Informationen. Dies gilt auch für Inhalte Dritter, die über dieses Angebot zugänglich sind.

Ein Dienst der *ETH-Bibliothek*
ETH Zürich, Rämistrasse 101, 8092 Zürich, Schweiz, www.library.ethz.ch

http://www.e-periodica.ch

XML – Grundstein zum grenzenlosen Datenaustausch

Hintergründe zu Technologie und Einsatzbereich

Web-Experten muss man eigentlich nicht mehr erklären, was XML ist. Schliesslich ist die Empfehlung des W3C für die Extensible Markup Language bereits über zwei Jahre alt, und das ist in der Web-Welt eine ziemlich lange Zeit. Anders dagegen sieht es in der traditionellen IT-Welt aus, in der Welt grosser Datenbanken und noch grösserer Maschinen. Hier beginnt man erst allmählich Notiz von XML zu nehmen, obwohl diese Bereiche inskünftig noch mehr als die Web-Welt von XML profitieren dürften. Dieser Beitrag gibt einen Überblick über die grundlegenden Eigenschaften, Fähigkeiten und Einsatzgebiete von XML sowie der zugehörigen Substandards wie XSL, XQL usw.

XML ist eine Metasprache (Metasprache, da sie nur die Syntax, d.h. die Grammatik, und nicht die Sprachelemente selbst definiert), mit der sich anwendungsbezogen inhaltliche Strukturen von Dokumenten und Daten unterschiedlichster Art beschreiben lassen. Durch diesen universellen, flexiblen und erweiterbaren Ansatz eröffnet sich für XML von der Textverarbeitung über Electronic Business bis zur Datenspeicherung ein fast unbegrenztes Einsatzgebiet. Die Bedeutung von XML ist vielleicht am ehesten mit SQL zu vergleichen. Auch dieser Standard bildete über Jahrzehnte eine feste Grundlage, auf der sich Hersteller über Plattformen und Systemgrenzen hinweg immer wieder getroffen haben. SQL war einer der Angelpunkte, um den sich die IT-Welt bei ihrem gigantischen Aufschwung drehen konnte. XML wird in den nächsten Jahren eine ähnliche Rolle spielen.

Adresse des Autors
Frank Jung, Software AG, Uhlandstrasse 12
D-64297 Darmstadt
E-Mail frank.jung@softwareag.com

Von SGML über HTML zu XML

Grenzen von HTML

Die Entstehungsgeschichte von XML ist eng mit der Entwicklung des World Wide Web verknüpft. Eine der Grundfesten des World Wide Web ist HTML (Hypertext Markup Language), mit der der Aufbau von Web-Seiten beschrieben wird. Die Anwendung von HTML ist nicht kompliziert: Der Web-Designer legt mit einer Reihe von Codezeichen (Tags) fest, wie das Dokument im Browser erscheinen soll, welche Links zu anderen Web-Seiten bestehen, ob eventuell Applets eingebunden sind usw. HTML ist eine recht einfache Seitenbeschreibungssprache. Der grosse Vorteil von HTML ist, dass die Web-Seiten überall weitgehend gleich aussehen, sich allerdings auch automatisch an unterschiedliche Bildschirmgrössen anpassen. HTML-Seiten können heute ohne weiteres mit einem grafischen Editor erstellt werden, das früher übliche manuelle Eingeben der Tags entfällt weitgehend.

Mit der schnell fortschreitenden Entwicklung des Web ist HTML allerdings an Grenzen gestossen:

- HTML erlaubt keine semantische Kennzeichnung einzelner Elemente. Damit ist unter anderem die Wiedergabe von Spezifikationen der Datenstrukturen, wie sie etwa bei Datenbanken oder Objekt-Hierarchien benötigt werden, nicht möglich.
- HTML beschreibt nur das Erscheinungsbild von Dokumenten und kann keine inhaltlichen Aspekte erfassen. Damit kann HTML nicht für gezielte Abfragen verwendet werden.
- HTML ist nicht erweiterbar; es ist daher nicht möglich, eigene Tags für spezifische Anforderungen zu definieren, ebenso wenig können Datenformate innerhalb von Dokumenten verwendet werden.
- Weiter gehende Anforderungen an Web-Seiten müssen aufwendig durch Programmierung abgedeckt werden, sei es durch Applets oder durch spezielle Applikationen. Die Beschränkungen von HTML haben nicht zuletzt zu einer Reihe proprietärer Erweiterungen geführt, was wiederum die universelle Verwendbarkeit der in HTML formatierten Informationen und damit die des Web insgesamt einschränkt.

SGML – flexibel, aber kompliziert

Die wenigsten Anwender von HTML wissen, dass HTML auf SGML (Standard Generalized Markup Language) basiert. SGML ist eine sehr komplexe Metasprache, welche erlaubt, Regeln für den Umgang mit Dokumenten unterschiedlicher Art zu definieren. Zahlreiche Unternehmen und Organisationen haben eigene Standards auf der Basis von SGML entwickelt. Die Bandbreite reicht dabei von sumerischen Inschriften bis zu technischen Dokumentationen in der Elektrotechnik (s. z.B. *Bulletin SEV/VSE* 17/99: «Die Publikation der Niederspannungs-Installations-Norm auf der Basis von SGML») oder auch im Flugzeugbau und umfasst auch Spezialitäten wie Musiknoten.

Eine der Anwendungen von SGML ist HTML. Sie wurde 1989 von Tim Berners-Lee speziell für die Verteilung von Dokumenten im Internet «erfunden». Berners-Lee beschränkte sich ganz auf die formalen Aspekte der darzustellenden Dokumente und klammerte alles aus, was – aus damaliger Sicht – für das neue Medium des Web nicht erforderlich war. Bei allen Einwänden gegen HTML sollte man nicht vergessen, dass die Verbreitung und Akzeptanz des Web ohne die Einfachheit und Beschränkung von HTML gar nicht möglich gewesen wäre. Niemand hätte zum Beispiel vor sieben oder acht Jahren den Sinn inhaltlicher Dokumentklassifizierungen im Internet eingesehen.

Da SGML die «Muttersprache» von HTML ist, liegt der Gedanke nahe, die Beschränkungen von HTML durch den Einsatz von SGML zu überwinden. Tatsächlich hat SGML einige entscheidende Vorteile gegenüber anderen Seitenbeschreibungssprachen. Erstens handelt es sich um einen allgemeinen Standard, der von einer Vielzahl von Softwareherstellern unterstützt wird. Deswegen ist der Fortbestand einer Dokumentendatenbank im SGML-Standard viel weniger von der Kurzlebigkeit herstellerspezifischer Standards abhängig. Zweitens können die Dokumente von jedem gelesen und gleichzeitig von Programmen syntaktisch zerlegt und analysiert werden. Drittens beschreiben SGML-Dokumente die Daten selbst, nicht nur die Art ihrer Darstellung. Das heisst, die Tag-Namen (in spitze Klammern gesetzte Auszeichnungsnamen) sollten sich in der Regel thematisch auf die damit markierten Daten beziehen, um diese eindeutig nach inhaltlichen Gesichtspunkten zu klassifizieren, das heisst, deren Typ (Bedeutung) im Klartext festzulegen.

Aber SGML hat auch Schwächen, die einer allgemeinen Verbreitung im Wege stehen. SGML stammt aus einer Zeit, in der Dokumente überwiegend in der Stapelverarbeitung verarbeitet wurden. So ist SGML sehr allgemein und komplex, die Spezifikation umfasst rund 500 Seiten, das meiste davon ist für den praktischen Einsatz im Web nicht relevant. Da SGML für jeden Aspekt so viele Optionen zur Verfügung stellt, ist die praktische Interoperabilität zwischen Unternehmen trotz Standardisierung gering.

Von SGML zu XML

Während für die aktuellen Anforderungen SGML zu kompliziert ist, ist die SGML-Anwendung HTML zu einfach. Eine Erweiterung von HTML aber wäre ihrerseits wieder begrenzt und müsste laufend angepasst werden, wobei bei jeder Anpassung erneut die Gefahr einer Zersplitterung bestünde. Notwendig war also nicht eine breitere HTML, sondern eine einfachere SGML. Also eine neue Metasprache, die es den Anwendern erlaubt, eigene Anwendungen auf Basis eines einfachen Standards zu realisieren: das ist XML.

XML, die Extensible Markup Language, ist – anders als HTML – eine Untermenge von SGML. XML ist eigentlich gar keine eigenständige Markup Language, sondern eine Metasprache, ein Definitionsapparat, mit dem die Anwender eigene Markup Languages bauen können. XML stellt lediglich ein Sprachgerüst zur Verfügung, mit dem sich die auf seiner Basis ganz unterschiedlich gebauten Anwendungen untereinander verstehen können. Gegenüber SGML ist XML in erster Linie wesentlich einfacher: seine Spezifikation durch das World Wide Web Consortium (W3C) umfasst nur noch überschaubare 26 Seiten.

XML wurde aus Kompatibilitätsgründen so gebaut, dass HTML in den neuen Rahmen hineinpasst. So trägt der Nachfolger von HTML 4 den Namen XHTML und ist nichts anderes als eine spezielle XML-Anwendung. Freilich ist nicht jede real existierende HTML-Seite auch eine gültige XML-Seite: Im Laufe der Zeit wurden Webbrowser so ausgebaut, dass sie auch syntaktisch fehlerhafte HTML-Seiten verstehen – und die sind im XML-Standard ausdrücklich verboten.

XML – die einfache Metasprache

Der grosse Unterschied zwischen HTML und XML besteht darin, dass man bei XML eigene Tags definieren kann. Damit ist es möglich, Daten nicht nur nach formalen Kriterien (wie Überschrift, Textkörper usw.) zu strukturieren, sondern auch nach inhaltlichen Gesichtspunkten. Deren Festlegung geschieht jedoch nicht in XML, sondern auf Basis von XML, denn XML ist ja eine Metasprache. XML erlaubt eine inhaltliche Strukturierung, da sie ja auf einer anderen Ebene als HTML agiert, genauso gut würde sie eine über den Umfang von HTML hinausgehende Layout-Beschreibung zulassen.

XML lebt daher durch das, was die Anwender auf Basis von XML schaffen. Jeder Anwender kann mit XML neue, eigene Tags für seine jeweiligen Anforderungen definieren, beispielsweise <Geburtsdatum>, <Schuhgrösse> oder <Kochzeit> – sofern man über XML Kochrezepte austauscht. XML definiert nicht selbst diese Tags, sondern schreibt in erster Linie vor, wie Tags definiert werden müssen; darüber hinaus gibt es auch noch XML-Tag-ähnliche Anweisungen, die für alle Dokumente gelten (z.B. XML-Prozessanweisungen, Elementtyp-Deklarationen usw., die speziell bei der Hinterlegung von Definitionen der Dokumentenstruktur (DTD) von Bedeutung sind).

XML stellt damit eine Grammatik für die Beschreibung des Inhalts von Dokumenten bereit, also zum Beispiel

<Bezeichner> Inhalt </Bezeichner>,

die der Anwender mit Inhalt füllen muss; beispielsweise so:

<Geburtsdatum> 25.10.78 </Geburtsdatum>.

Genauso wäre es beispielsweise denkbar, dass Meteorologen für den Austausch von Wetterdaten eigene Tags wie etwa <Temperatur>, <Luftdruck>, <Windstärke> usw. festlegen und diese Definitionen in entsprechenden Schemas ablegen. XML-fähige Anwendungen könnten dann derartige Web-Seiten direkt verarbeiten, also beispielsweise Wetterdaten automatisch per Web auswerten.

Dennoch ist auch XML nicht in der Lage, den Inhalt eines Dokuments zu verstehen. Was sich hinter den Tags verbirgt, ist Sache des Anwenders. Anstelle der «sprechenden» Bezeichnung <Geburtsdatum> könnte natürlich auch <XYZ56789ABC> verwendet werden, ohne dass ein XML-Parser[1] dies beanstanden würde. Damit ginge jedoch ein wichtiger Vorzug von XML verloren, da sie durch die Möglichkeit, selbst definierte Tags einzusetzen, gerade die Verwendung «verstehbarer» Tags erlaubt.

Auch die verstehbaren, anwendungsspezifischen Tags ergeben nur dann einen Sinn, wenn alle in Frage kommenden Nutzer sie kennen. Das ist dann der Fall, wenn sich Benutzergruppen auf bestimmte Dokumenttypen einigen. Dazu dienen DTD (s. Kap. Document Type Definition – DTD), Dokumentenschemas, die die zulässigen Tags und die zulässige Struktur einer Dokumentenklasse festlegen. Trotz vorgegebener Struktur können XML-Dokumente unbekannte (nicht in der DTD definierte) Tags verkraften, solange das Dokument syntaktisch fehlerfrei (wohlgeformt) ist. Die XML-Tags werden als solche erkannt und als «ungültig» deklariert. Der Empfänger eines derart «undefinierten» XML-Dokuments kann somit eine Falschinterpretation vermeiden. Die XML-Architektur ist also sehr flexibel und bereits darauf ausgerichtet, dass beim Dokumentenaustausch auch Anwender bedient werden können, die Informationen benötigen, welche

<?xml version="1.0"?>	XML-Kopfteil
<Produkt>	Anfangs-Bezeichner (Start-Tag)
....	
<Preis gültig-ab="01/01/99">	Anfangs-Bezeichner (Start-Tag) und Attribut
49,95	Inhalt
</Preis>	Ende-Bezeichner (End-Tag)
</Produkt>	Ende-Bezeichner (End-Tag)

Bild 1 Die grundlegende Syntax von XML

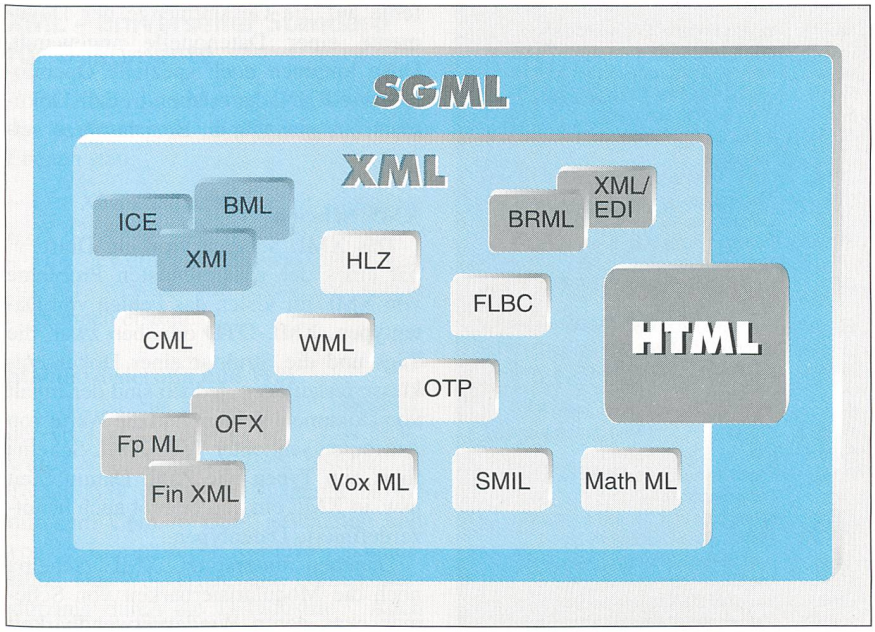

Bild 2 Die Struktur von SGML und XML

XML ist (nach einigen Anpassungen von SGML) eine Untermenge von SGML, HTML eine Anwendung von SGML (und XHTML eine Untermenge von XML).

über die von DTD geforderten Inhalte hinausgehen. Da die Definitionen zudem in Klartext erfolgen und nicht in kryptischen Steuerzeichen, bleiben XML-Dokumente lesbar. Unabhängig von der jeweiligen DTD des Erstellers können alle XML-kodierten Dokumente bearbeitet, gespeichert und verteilt werden (Bild 1).

XML kann auf diese Weise sehr flexibel an alle denkbaren Einsatzgebiete angepasst werden. Der Phantasie sind keine Grenzen gesetzt; alle Informationen, die in Textform vorliegen, können mit XML erfasst werden. Man kann Preise, Verfassernamen, Zeit- oder Datumsangaben, Schlagworte, Aktienkurse usw. definieren. Für solche Inhalte erweist es sich als sinnvoll, dass bezogen auf bestimmte Anwendungsbereiche (beispielsweise Immobilienmakler, Börsendienste, Verlage) jeweils spezielle Dokumententypen mit Hilfe einer DTD festgelegt werden. In der Tat haben schon viele Organisationen, Vereinigungen und Industriegruppen sich auf entsprechende XML-Dokumententypen geeinigt. Auf dieser Basis ist es dann zum Beispiel möglich, gezielte Abfragen nach den definierten inhaltlichen Kriterien durchzuführen. Die Ergebnisse der Auswertung von Dokumenten oder Web-Seiten, die einem XML-Standard entsprechen, können auch direkt in Anwendungsprogrammen weiterverarbeitet werden. So könnte etwa eine Applikation aus Web-Seiten selbstständig Preisangaben oder Aktienkurse herauslesen und diese verwenden. Und ebenso lassen sich Metadaten für CAD-Daten oder für Röntgenbilder erstellen.

XML findet vielseitige Anwendung im Datenaustausch zwischen unterschiedlichen Systemen. XML ist weitaus flexibler als das starre Feld-Konzept relationaler Daten und erweitert die Leistungsfähigkeit von Schnittstellenstandards wie Corba und DCOM. Auch wenn der Einsatz von XML-Technologie heute hauptsächlich in Verbindung mit dem Web diskutiert wird, geht ihr Anwendungsbereich weit darüber hinaus; er liegt überall dort, wo komplexe Daten gespeichert oder übertragen werden.

Hat man sich einmal auf ein XML-Format, beispielsweise zur Darstellung von Molekülstrukturen, verständigt, kann man nicht nur im Web gezielt nach bestimmten chemischen Verbindungen suchen. Es ist ebenso möglich, solche Informationen auf ähnliche Weise in einer Datenbank zu speichern und abzurufen. Solche Strukturen in den Feldern einer relationalen Datenbank abzubilden, so dass nach einzelnen Komponenten oder Verbindungen gezielt gesucht werden kann, ist nahezu unmöglich. Hier scheitert der relationale Ansatz an der Komplexität der Information. Es bleibt daher meist die unstrukturierte Speicherung als Text oder Grafik, wobei die Informationen aber nur dargestellt werden, sich jedoch nicht auswerten lassen. Da Informationen überwiegend nur in unstrukturierter Form verfügbar sind – der Rest sind Dokumente, Bilder, Grafiken, Spread-Sheets usw. –, lässt sich das Potenzial einer Sprache ermessen, die diese Daten erfassen kann (Bild 3).

Beispiel Shakespeare

Bild 5 zeigt die grundlegende Logik von XML im Vergleich mit einem «normalen», also nicht IT-gerechten Text und mit HTML. Basis ist eine Anwendung von XML für die Darstellung von Shakespeare-Dramen[2]. Haben sich einmal alle Shakespeare-User auf die einheitliche Verwendung ihrer XML-Tags (hier: <ACT>, <TITLE> usw.) verständigt, also auch auf die Bedeutung und die Anwendungen von Schlüsselwörtern wie ACT, SCENE oder SPEECH, so kann ohne Schwierigkeiten eine entsprechende Applikation programmiert werden, die die Dramen ausdruckt oder möglicherweise einem Sprachausgabe-Algorithmus übergibt. Erweiterungen lassen sich nahtlos anschliessen.

Umsetzung von XML

Die XML-Welt setzt sich aus einer Reihe von separaten Substandards zusammen, die verschiedene Aspekte der Darstellung und Wiedergabe der Dokumente beschreiben.

Document Type Definition – DTD

DTD sind Schemas zur Definition von Dokumenttypen mittels Vorgabe von Dokumentstrukturen, inklusive der darin erlaubten Elemente. Damit kann beispielsweise festgelegt werden, dass die Tags in einer bestimmten Struktur in den betreffenden Dokumenten enthalten sein müssen[3]. DTD werden bei der Entwicklung von XML-Anwendungen aufgestellt. XML-Dokumente können auch ohne DTD verarbeitet werden, verlieren dann aber die dort hinterlegten Strukturinformationen. Im Wesentlichen werden DTD nur verwendet, um XML-Werkzeuge zu steuern und XML-Dokumente auf strukturelle Gültigkeit zu prüfen – für das Verständnis eines XML-Dokuments sind sie nicht erforderlich.

```
<!ELEMENT Produkt (Name, Kategorie?, Farbe?,
Beschreibung?, Produkt-Nummer, Verfügbarkeit?,
Preis?)>
<!ELEMENT Name (#PCDATA)*>
<!ELEMENT Beschreibung (#PCDATA)*>
<!ELEMENT Produkt-Nummer (#PCDATA)*>
<!ELEMENT Verfügbarkeit (#PCDATA)*>
<!ELEMENT Preis (#PCDATA)*>
<!ATTLIST Produkt Preis gültig-ab #required>[4,5]
```

Extensible Style Language – XSL: Formatierungssprache

Das Erscheinungsbild eines XML-Dokuments ist weder in ihm selbst noch in seiner DTD definiert. Es gehört zu den Grundgedanken von XML, Inhalt und

Darstellung völlig zu trennen. Wie ein Dokument dargestellt wird, wird in einem Style Sheet festgelegt, das auf dem XSL- oder CSS-Standard beruht. Es kann auch mehrere XSL-Sheets für dasselbe Dokument geben, die das gleiche Dokument verschieden darzustellen vermögen. Dabei unterstützt XSL auch verschiedene Ausgabemedien, wie Bildschirm, Drucker u.a. Mit Hilfe von XSL können XML-Dokumente auch in HTML-Dokumente umgewandelt werden. Geschieht dies auf einem Server, so können auch Endgeräte unterstützt werden, die zwar HTML, aber kein XML verstehen.

```
<xsl:template match="Produkt-Katalog">
 <ul>
  <xsl:apply-templates select="Produkt">
   <xsl:sort select="Name"/>
  </xsl:apply-templates>
 </ul>
</xsl:template>

<xsl:template match="Produkt">
 <li>
  <xsl:value-of select="Name"/>
  <xsl:text> </xsl:text>
  <xsl:value-of select="Produkt-Nummer"/>
  <xsl:text> </xsl:text>
  <xsl:value-of select="Preis"/>
 </li>
</xsl:template>
```

In obigem Beispiel sortiert ein Style Sheet (Formblatt) aus einem Produktkatalog alle Produkte nach <Name> und gibt die Liste in einem HTML-Dokument aus. Wie man sieht, sind XSL-Style-Sheets selbst wieder XML-Dokumente. Sie lassen sich deshalb auch mit den gleichen Werkzeugen wie XML bearbeiten.

XSL hat gegenwärtig den Status eines Working Draft[6,7] (Arbeitsgruppenvorschlag).

XPointer und XLink

Die Konventionen für das Verbinden von Objekten (Linking) in XML-Dokumenten sind mit XLink[8] und XPointer[9] festgelegt. Neben klassischen HTML-Links werden ausserdem unterstützt:

- bidirektionale Links
- erweiterte 1:n-Links (beispielsweise Links zu mehreren Versionen eines Dokuments)
- indirekte Links
- Adressierung (Links können sich auf bestimmte Teile von Dokumenten richten)

XML Query Language – XQL

XQL ist derzeit noch ein Vorschlag für eine Abfragesprache von XML-Dokumenten[10,11]. Ähnlich wie SQL bei relationalen Datenbanken dient XQL dazu, XML-Dokumente oder Teile von Dokumenten von einer XML-Datenquelle, z.B. einer Datenbank, abzurufen. Dabei ist der Befehlsvorrat von XQL sehr der XSL-Pattern-Language angenähert, die es erlaubt, aus einem Dokument bestimmte Elemente herauszufischen. Im Gegensatz zu XSL werden jedoch bei XQL die Befehle auf die Gesamtmenge der Dokumente einer Datenquelle angewandt. Dazu kommen noch spezielle Operatoren, wie JOIN, der erlaubt, fremde Dokumente miteinander in Beziehung zu setzen.

aecXML	for Architecture, Engineering and Construction
AIML	Artificial Intelligence Markup Language
BioML	Biopolymer Markup Language
BML	Bean Markup Language
BSML	Bioinformatic Sequence Markup Language
CPML	Call Policy Markup Language
CPL	Call Processing Language
CFML	Cold Fusion Markup Language
BRML	Common Rules and Business Rules Markup Language
DTD	for patent documents - ST32 US Patent Grant
ECMdata	Electronic Component Manufacturer Data Sheet Library Specification
XFRML	Extensible Financial Reporting Markup Language
EAD	Encoded Archival Description
FpML	Financial Product Markup Language
FinXML	XML for Capital Markets
FLBC	Formal Language for Business Communication
ICE	Information and Content Exchange
IOTP	Internet Open Trading Protocol
JSML	Java Speech Markup Language
KBML	Koala Bean Markup Language
LitML	Liturgical Markup Language
MathML	Mathematical Markup Language [20]
MoDL	Molecular Dynamics Language
NAA	Standard for Classified Advertising Data
NVML	Navigation Markup Language
OTP	Open Trading Protocol
PMML	Predictive Model Markup Language
PML	Procedural Markup Language
PDX	Product Definition Exchange
RDF	Resource Description Framework [21]
SVG	Scalable Vector Graphics
SMIL	Synchronized Multimedia Integration Language[22]
TMX	Translation Memory Exchange
TML	Tutorial Markup Language
VoxML	Voice Recognition ML
WIDL	Web Interface Definition Language
WML	Wireless Markup Language for WAP [23]
XBEL	XML Bookmark Exchange Language
XMI	XML Metadata Interchange

Bild 3 Weitere XML-Aktivitäten

XML Schema

Das XML Schema Working Draft[12,13] hat eines der gravierendsten Probleme von XML im Visier, das Fehlen von Datentypen. XML-DTD erlauben zwar, die Tags und die Struktur einer Dokumentklasse festzulegen, jedoch sind der Inhalt von Dokumentelement und die Werte von Attributen schlichter Text. XML Schema führt nun Typen wie Zahl, Datum, Zeit usw. in XML ein und erlaubt auch benutzerdefinierte Datentypen.

Daneben unterstützt XML Schema noch die Modularisierbarkeit von Schemas, was deren Wiederverwendbarkeit verbessert. XML Schemas werden wiederum in XML abgefasst und können so mit XML-Werkzeugen bearbeitet werden[14].

Document Object Model – DOM

Das Document Object Model[15] ist ein API (Application Programming Interface) für HTML- und XML-Dokumente. Es erlaubt Anwendungsprogrammen, in der Struktur von Dokumenten zu navigieren, einzelne Elemente oder Attribute zu lesen, hinzuzufügen, zu verändern oder zu löschen. DOM ist ein plattform- und programmiersprachenneutrales Interface, das Programmen und Skripten den dynamischen Zugriff auf Dokumente erlaubt, um diese bezüglich Inhalten, Struktur und Form zu verändern. Das Modell wird beispielsweise von Microsoft Office 2000 und anderen Anbietern unterstützt.

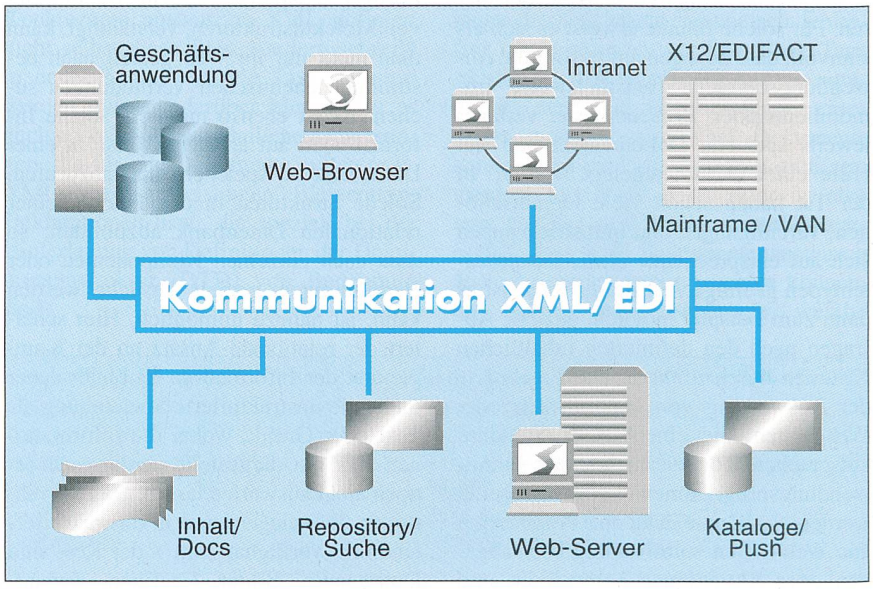

Bild 4 Firmen-zu-Firmen-Kommunikation mit XML/EDI

XML – universeller Standard für Dokumente

Da XML eine Metasprache ist, findet die Festlegung der Bedeutung auf zwei Ebenen statt:

- Der XML-Standard[16] selbst wird vom W3C betreut und weiterentwickelt.
- Die jeweiligen Anwendungen von XML werden von freien Benutzergruppen erstellt.

Standardisierung und W3C

XML ist ein unabhängiger Standard, der vom W3C gepflegt wird. Das W3C wurde vom Massachusetts Institute of Technology (MIT) in Zusammenarbeit mit dem Cern in Genf geschaffen und von der Europäischen Kommission unterstützt. Das W3C erfährt eine breite Unterstützung durch die IT-Industrie[17]. Auch die Software AG ist Mitglied im W3C.

XML-Anwendungen

In der Industrie gibt es bereits zahlreiche Initiativen zur Einführung von XML als Standard für den Datenaustausch. In allen Branchen haben sich Unternehmen und Organisationen zusammengefunden, um XML-Anwendungen zu entwickeln:

- HL7, ein Zusammenschluss von Gesundheitsorganisationen, die Standards für den elektronischen Austausch von Klinik-, Finanz- und Verwaltungsdaten zwischen unabhängigen Computersystemen des Gesundheitswesens[18] entwickeln
- Chemical Markup Language (CML), entwickelt in Grossbritannien, um Chemikern den Austausch von Beschreibungen von Molekülen, Formeln und anderen chemischen Daten zu ermöglichen[19]
- Open Financial Exchange (OFX), das von Intuit Quicken und Microsoft Money benutzte Format für die Kommunikation mit Banken
- Open Software Distribution (OSD) von Marimba und Microsoft

Einige weitere Aktivitäten sind in Bild 5 angegeben. Ausführliche Informationen finden sich unter www.xml.com.

EDI

Zu den wichtigsten Anwendungen von XML zählt EDI (Electronic Data Interchange). Schon seit mehreren Jahren

Tamino – eine Datenbank für Internetanwendungen

Das XML-Datenbankmanagementsystem Tamino der Software AG ist eine vollständig neue integrierte Datenbank-Lösung, die folgende Elemente umfasst:

- X-Machine: eine hochleistungsfähige, vom Typ her vollkommen neue, auf XML basierende Methode zur Datenspeicherung, die benutzerdefinierte Server-Erweiterungen und Dienste zur Dokumententransformation umfasst
- X-Node: eine Komponente zur Datenabbildung, die eine integrierte, einheitliche Sicht auf verteilte, heterogene Datenquellen ermöglicht
- X-Port: ein auf HTTP basierendes Web-Server-Interface, über welches mittels URL direkt vom Internet aus auf Tamino-Objekte zugegriffen werden kann
- SQL-Engine: ein Web-fähiger SQL-Speicher für strukturierte Daten
- Tamino SDK: ein Werkzeugpaket mit Funktionsbibliotheken für Softwareentwickler, die von XQL-, SQL- oder OO-Anwendungen (DOM) aus direkt auf Tamino zugreifen möchten
- Tamino Manager: eine auf Internet basierende Schnittstelle zur «ferngesteuerten» Verwaltung (Single Point of Control)

Tamino ist damit in der Lage, alle Arten von Internet-Objekten wie XML- oder HTML-Seiten in XML-Notation zu speichern (ohne diese vorgängig in andere Formate zu wandeln) sowie ein sicheres Transaktionskonzept für Geschäftstransaktionen über das Internet anzubieten. Ferner kann Tamino komplexe Informationsobjekte und strukturierte Daten effizient auf eine konsistente, benutzerdefinierte Art darstellen und Dokumente von Anwendungen aller Art wie Briefe, Faxe und Kalkulationstabellen speichern.

Eine Selbstverständlichkeit im Multimedia-Zeitalter ist Taminos Fähigkeit, neben herkömmlichen Informationen (Texte und Zahlen) alle Arten von komplexen Informationsobjekten wie Multimedia- oder Biometrie-Daten zu speichern. Über X-Node kann der Anwender auf externe Datenbanken wie z.B. Adabas oder andere RDBMS zugreifen.

Klartext	HTML	XML
Erster Aufzug ERSTE SZENE. Helsingör. Eine Terrasse vor dem Schlosse. Francisco auf dem Posten. Bernardo tritt auf. BERNARDO: Wer da? FRANCISCO: Nein, antwortet mir: steht und gebt Euch kund. BERNARDO: Lang lebe der König!	<H1>ERSTER AKT</H1> <P><I>ERSTE SZENE. Helsingör. Eine Terrasse vor dem Schlosse.</I></P> <P><I>FRANCISCO auf dem Posten. BERNARDO tritt auf.</I></P> <P><B>BERNARDO:</B>Wer da?</P> <P><B>FRANCISCO:</B>Nein, antwortet mir: steht, und gebt Euch kund.</P> <P><B>BERNARDO:</B>Lang lebe der König!</P>	<ACT><TITLE>ERSTER AKT</TITLE> <SCENE><TITLE>ERSTE SZENE. Helsingör. Eine Terrasse vor dem Schlosse.</TITLE> <STAGEDIR>FRANCISCO auf dem Posten. BERNARDO tritt auf.</STAGEDIR> <SPEECH> <SPEAKER>BERNARDO</SPEAKER> <LINE>Wer da?</LINE> </SPEECH> <SPEECH> <SPEAKER>FRANCISCO</SPEAKER> <LINE>Nein, antwortet mir: steht, und gebt Euch kund.</LINE> </SPEECH> <SPEECH> <SPEAKER>BERNARDO</SPEAKER> <LINE>Lang lebe der König!</LINE> </SPEECH>
Um diesen Auszug aus Shakespeares Hamlet lesen zu können, benötigen Menschen keine Textcodierungen oder Tags. Sie sind auf Grund weit zurückliegender Erfahrung der Schulzeit in der Lage, die Bedeutung einzelner Textstücke zu erkennen oder abzuleiten.	Derselbe Text in HTML-Version. Die HTML-Tags legen nur fest, wie der Text in einem Browser dargestellt wird, sie fügen dem Text keinen semantischen Wert hinzu und können daher auch nicht nach inhaltlichen Fragestellungen ausgewertet werden (zum Beispiel «Wie viel Text hat Bernardo zu sprechen?»).	Dem Text wurden nun XML-Tags hinzugefügt. Computer können die Bedeutung nicht aus einem Textzusammenhang erschliessen, sondern nur Zeichenketten anhand definierter Regeln analysieren. Sie benötigen daher Textcodierungen, um Informationen über den Inhalt zu erhalten. Der XML-Text könnte nun zum Beispiel in einen Sprachcomputer geladen oder für die Aufstellung einer Besetzungsliste ausgewertet werden.

Bild 5 XML im Schloss von Helsingör

```
<?xml version="1.0"?>
<!DOCTYPE bioml SYSTEM="bioml.dtd">
<bioml label="Insulin, Gen- und Proteinstruktur">
        &paragraph;
        Diese BIOML-Datei enthält eine einfache Informationssammlung über das Protein Insulin. Das
        Beispiel ist nicht als extensive Studie über Insulin gedacht, sondern nur als Demonstration des
        Aufbaus einer BIOML Datei. Um zum nächsten Thema zu gelangen clicken Sie es bitte mit der
        Mouse an oder drücken Sie die "WEITER" Taste.
        <organism label="Homo sapiens (Mensch)">
        <chromosome label="Chromosom 11" number="11">
        &paragraph;
        Der Eintrag über Chromosome zeigt, dass der Abschnitt der das Insulin-Gen beinhaltet, bei
        Menschen im Chromosom Nummer 11 zu finden ist. Alle Einträge die mit diesem Eintrag verbunden
        sind, befinden sich auf diesem Chromosom.
        <locus label="HUMINS Abschnitt">
        &paragraph;
        Der HUMINS Abschnitt enthält die komplette Sequenzinformation, die zur Codierung von Insulin
        benötigt wird. Der Abschnitt kann entweder insgesamt, oder nur in Fragmenten bekannt sein. In vie-
        len BIOML-Dateien enthält der Abschnitt ein gesuchtes Gen (oder mehrere).
        ...
          <gene label="Insulin Gen">
            <dna label="Komplette HUMINS Sequenz" Start="1" Ende="4992">
                1 ctcgaggggc ctagacattg ccctccagag agagcaccca acaccctcca ggcttgaccg
                61 gccagggtgt ccccttccta ccttggagag agcagcccca gggcatcctg caggggggtgc
                121 tgggacacca gctggccttc aaggtctctg cctccctcca gccaccccac tacacgctgc
                181 tgggatcctg gatctcagct ccctggccga caacactggc aaactcctac tcatccacga
                ...
                4861 ggccagggct gggcaggcgg gtggacggcc ggacactggc cccggaagag gagggaggcg
                4921 gtggctggga tcggcagcag ccgtccatgg gaacacccag ccggccccac tcgcacgggt
                4981 agagacaggc gc
            </dna>
          </gene>
        </locus>
        </chromosome>
        </organism>
        ...
      <copyright label="1998 ProteoMetrics, LLC">
        &cr;Copyright &copyright; 1998 ProteoMetrics, LLC. Alle Rechte vorbehalten.
      </copyright>
</bioml>
```

Bild 6 BIOML – Polymere Sequenzen, Insulin (stark verkürzt)

BIOML wurde mit XML zur Darstellung biopolymerer Strukturen entwickelt. Ziel war es, einen Erfahrungsaustausch von Wissenschaftern über das Web zu ermöglichen. Im Beispiel wird ein Insulin-Protein beschrieben[24]. Bild 6 wurde in Teilen reproduziert und übersetzt, mit freundlicher Genehmigung von Proteo Metrics, LLC, New York.

wird versucht, einen EDI-Standard für den Austausch von Informationen zwischen Unternehmen zu etablieren. Der Grundgedanke ist dabei ebenso einfach wie überzeugend: Gelänge es, Daten, wie sie beispielsweise in Lieferscheinen, Aufträgen oder Rechnungen enthalten sind, mit einem kompatiblen Format auszutauschen, so könnte man sich die aufwendige und fehleranfällige Erfassung solcher Daten sparen. In der Praxis scheiterte dieser Ansatz allerdings an der unüberschaubaren Fülle von Daten, Datenarten und Formaten. In den USA nutzen gerade einmal 2 Prozent der Unternehmen EDI.

Mit XML lassen sich diese Schwierigkeiten bewältigen, weil nun ein gemeinsames Dach für alle EDI-Anwendungen verfügbar ist, das dennoch für alle Anforderungen genügend Freiraum lässt. Das von der XML/EDI-Gruppe konzipierte XML/EDI stellt eine neue Grundlage für den Business-to-Business-Datenaustausch dar und damit auch für den wichtigen Bereich des Electronic Business. Auf Basis von XML werden dabei nicht nur Dokumente ausgetauscht, sondern Transaktionen wie beispielsweise Bestellvorgänge durchgeführt. Um solch leistungsfähige Anwendungen zu erstellen, ist also auch eine performante XML-Infrastruktur von grösster Bedeutung[5] (Bild 4)

XML und Datenbanken

Die grosse Flexibilität von XML erschliesst diesem Standard ein Einsatzgebiet, das weit über das von HTML hinausreicht. XML kann nicht nur komplexe hierarchische Informationen verarbeiten, sondern lässt sich auch für kommerzielle Transaktionen einsetzen. Aus Anwendungssicht ist XML das, was TCP/IP aus Sicht der Kommunikation ist. Beide Standards zusammen bilden eine Grundlage für die Nutzung des Web für Electronic Business. Vor diesem Hintergrund müssen zwei Forderungen an eine XML-Infrastruktur gestellt werden:

1. XML-Informationen müssen – gegebenenfalls in grossem Umfang und auch für Transaktionen – schnell und sicher zur Verfügung gestellt werden.
2. XML-Informationen müssen in den vorhandenen Unternehmensdatenbestand integriert werden.

Lassen sich HTML-Seiten noch in einem Dateisystem verwalten, so sind für die komplexen XML-Dokumente leistungsfähigere Systeme nötig. Dafür bieten sich nach dem derzeitigen Stand der Technik Datenbanken an, deren Funktionalität – Konsistenz, Wiederanlauffähigkeit, Datensicherung und -wiederherstellung usw. – die XML-Daten nutzen können.

Der einfachste Weg, XML-Objekte in einer Datenbank unterzubringen, ist die Speicherung als «Character Large Objects». Dabei würden die Tags in diesen Objekten lediglich als Fliesstext behandelt werden, die mit den Methoden der Volltextsuche erschlossen werden können. Darüber hinaus kann eine Datenbank zusätzliche Möglichkeiten der Indexierung von Informationsobjekten bieten, was flexiblere Zugriffspfade eröffnet. Damit wäre der Zugriff auf diese Objekte sowohl über die Strukturen der Objekte (Structure-based Retrieval) als auch über ihre Inhalte (Content-based Retrieval) möglich. Allerdings müssten auf diesem Weg sämtliche relevanten Informationen in separaten Feldern oder Tabellen verwaltet werden. Die hierarchischen Strukturen von XML-Dokumenten könnten auf diese Weise jedoch nicht abgebildet werden. Eine derartige Lösung liesse sich mit den gegenwärtig verfügbaren Mitteln zwar relativ einfach umsetzen, eine performante Bewältigung auch grösserer Datenvolumina oder gar für XML-Transaktionsdaten liesse sich so keinesfalls erreichen.

XML und SQL

Der naheliegende Weg einer Integration von XML in herkömmliche Datenbanktechnologien ist die Implementierung von XML-Strukturen entsprechend heute weithin üblichen Datenmodellen, wie zum Beispiel dem relationalen Modell. XML-Daten könnten dafür über einen Konversionsprozess in relationale Datenbankstrukturen eingepasst werden. Umgekehrt sollten auch relational gespeicherte Daten abfragebedingt in einen XML-Datenstrom zurückgewandelt und als XML-Dokument wiedergegeben werden können.

```
<?xml version="1.0"?>
<!DOCTYPE anzmeta PUBLIC "-//ANZLIC//DTD ANZMETA 1.1//EN"
"http://www.environment.gov.au/net/anzmeta/anzmeta-1.1.dtd">
<anzmeta>
        <citeinfo>
                <uniqueid>
                        ANZCW0301000001
                </uniqueid>
                <title>
        AVHRR NDVI 14-tägige Versuchsreihe, die Kontinental-Australien in voller Auflösung wiedergibt.
                </title>
                <origin>
                        <custod>Meteorologiebüro</custod>
                        <jurisdic><keyword>Australien</keyword></jurisdic>
                </origin>
        </citeinfo>
        <descript>
                <abstract>
                        <p>
                            AVHRR NDVI (Normalisierter Differenzen-Vegetationsindex). 14-tägige
                            Versuchsreihe, die Kontinental-Australien in "1 Kilometer-Auflösung"
                            wiedergibt. NDVI ist ein Mass für die Absorption von rotem Licht
                            durch pflanzliches Chlorophyll und die Reflexion von Infrarot Wärme-
                            verluste durch wassergefüllte Blattzellen. Seine Werte erlauben eine
                            grobe Dichtemessung des aktiven Blattwerks.
                        </p>
                </abstract>
                <theme>
                        <keyword qualifier="biodiversity">
                                LANDWIRTSCHAFT
                        </keyword>
                        <keyword qualifier="Management">
                                ATMOSPHÄRE
                        </keyword>
                        <keyword qualifier="Monitoring">
                                ATMOSPHÄRE Druck
                        </keyword>
                        <keyword qualifier="Umsetzung">
                                KLIMA UND WETTER
                        </keyword>
                        <keyword>
                                GEFAHREN Seuchen
                        </keyword>
                </theme>
                <spdom>
                        <place>
                                <dsgpolyo>
                                        <long>112.5</long>
                                        <lat>-10</lat>
                                        <long>154</long>
                                        <lat>-10</lat>
                                        <long>154</long>
                                        <lat>-44</lat>
                                        <long>112.5</long>
                                        <lat>-44</lat>
                                        <long>112.5</long>
                                        <lat>-10</lat>
                                </dsgpolyo>
                        </place>
                        <bounding>
                                <northbc>-10</northbc>
                                <southbc>-44</southbc>
                                <eastbc>154</eastbc>
                                <westbc>112.5</westbc>
                        </bounding>
                </spdom>
        </descript>
        <timeperd>
                <begdate>
                        <date>
                                1991-04-01
                        </date>
                                .
                                .
                                .
                                .
```

Bild 7 Ermittlung von Vegetationsdifferenzen in Australien

Die Umwandlung von relationalen Daten in XML-Objekte ist grundsätzlich immer möglich. Dabei stellt zurzeit das Fehlen von Datentypen in XML noch ein Problem dar, wobei jedoch mit der Entwicklung von XML Schema (siehe oben) Abhilfe in Sicht ist. Da bei relationalen Daten die Komposition komplexer Datenobjekte aus «flachen» Tabellen in der Anwendungslogik erfolgt, sind hierfür entweder zusätzliche Definitionen in Form von «Data Maps» erforderlich oder der Zusammenbau verschiedener Datensätze erfolgt im Rahmen einer XQL-Abfrage mittels JOIN.

Umgekehrt stellt die Implementierung beliebiger Baumstrukturen, wie sie von XML unterstützt werden, in einem relationalen Datenmodell zwar ein lösbares, aber in der Praxis kaum überschaubares Problem dar. XML erlaubt eine hierarchische Staffelung von Informationen in beliebiger Tiefe. Um dies in relationalen Datenbanken wiederzugeben, müssten umfangreiche Tabellenverknüpfungen nicht nur eingerichtet, sondern auch verwaltet werden. In der Praxis ergibt sich deshalb hier schon nach relativ wenigen Ebenen ein massives Performanceproblem. In diesem Modell könnten also wieder nur ganz einfache XML-Dokumente verwaltet werden.

Ein anderes, gravierendes Problem stellen die Sperrmechanismen der RDBMS (Relational Database Management Systems) dar, welche Sperren auf Dokumentebene nicht unterstützen – Dokumente spielen in der Methodologie der RDBMS keine Rolle. Für das Ändern eines in einem RDBMS abgebildeten XML-Dokuments müssten zahlreiche Sperren in vielen Tabellen kontrolliert werden – ein weiterer Grund für Performanceverluste und das Entstehen eines enormen internen Verwaltungsaufwands.

Ausserdem entzieht sich die typische Dokumentenstruktur vieler mit Hilfe von XML modellierter Informationsobjekte, wie zum Beispiel längere Textelemente, Bilder und komplexe Querverbindungen, einer unmittelbaren Darstellung mit relationalen Mitteln, ganz abgesehen von der Möglichkeit, dass man in XML auch nicht vordefinierte Tags verwenden kann. Es ist ja gerade der grosse Vorteil von XML, dass diese für Objekte dieser Art eine bessere Abbildung als «reines» SQL liefert.

Diese Problematik ist im übrigen davon unabhängig, an welcher Stelle der Datenbank die Konvertierung von XML ins relationale Modell erfolgt, ob als externe Schnittstelle oder Kernel-nah.

XML und ODBMS

Die Welt der hierarchisch strukturierten XML-Objekte legt auf den ersten Blick eine direkte Verbindung mit objektorientierten Datenbanken (ODBMS) nahe. Diese bieten ja für XML eine fast generische Möglichkeit, die Strukturen persistenter Objekte in einen Server zu implementieren. Damit könnten persistente Objekte an einer transparenten Schnittstelle mit XML bereitgestellt und von einer Vielzahl von Benutzern lokal oder über das Web genutzt werden. Allerdings werden die bekannten Nachteile der ODBMS nicht aufgehoben, wenn die Daten über eine XML-Schnittstelle laufen. ODBMS sind in der Praxis für hohen Durchsatz und grosse Datenmengen nicht

brauchbar; sie eignen sich vor allem wenig für Transaktionsaufgaben, wie sie gerade im Electronic Business gefordert werden. Zudem sind ODBMS aufwendig zu implementieren und fallen in ihrer Leistungsfähigkeit erst recht zurück, wenn es darum geht, Altdaten aus RDBMS oder gar aus Dateisystemen zu integrieren. ODBMS konnten sich daher bisher nur in Spezialaufgaben bewähren – und eine derartige Spezialaufgabe ist XML eindeutig nicht.

Die ideale Grundlage für die Speicherung und den Austausch verschiedenartiger XML-Objekte bietet ein XML-Server. Vor allem durch die Möglichkeit der impliziten Datenstrukturierung ergibt sich eine enorme Flexibilität, die gerade für Electronic-Business-Anwendungen gebraucht wird. Datenbankabfragen und -befehle werden dabei nicht als eine Abfolge von SQL-Queries formuliert, sondern als URL an den Server geschickt. So lassen sich in einer einzigen Abfrage nicht nur die auf dem XML-Server befindlichen Dokumente abfragen, sondern auch andere Datenquellen wie entfernte XML-Server, relationale Datenbanken oder Multimediaelemente von speziellen Multimediaservern. Für den Anwender erscheinen dabei die Daten von einem einzigen Server zu kommen – XML ist der Leim, der die verschiedenen Elemente zusammenhält.

So läge beispielsweise in einem Krankenhaussystem der Patientenbericht als XML-Dokument vor, die Patientenstammdaten wären in einer relationalen Datenbank gespeichert, und die Computertomographie läge als Volume-Bilddatei vor. All das kann von XML und einem XML-Server zu einem einzigen Dokument zusammengeführt werden. Dabei bewahrt der Einsatz von XML ein derartiges Konzept vor den Problemen proprietärer Ansätze. Die Software AG hat dieses Konzept mit seiner XML-Datenbank *Tamino* in die Realität umgesetzt (siehe Kasten S. 25).

Weitere Beispiele für XML-Anwendungen

Die Bilder 5–7 zeigen nicht die üblichen Vereinfachungen zur Erläuterung der XML-Grundlagen, sondern echte XML-Anwendungen. Tamino ist für die Verwaltung von Informationen dieser Art konzipiert.

Anmerkungen

[1] Der Parser untersucht die XML-Dokumente auf ihre syntaktische Korrektheit.

[2] Quelle: http://metalab.unc.edu/pub/sun-info/standards/xml/eg/shakespeare.1.10.xml.zip; Shakespeares Gesamtwerk wurde von hier heruntergeladen.

[3] XML unterscheidet Dokumente, die «well formed» (syntaktisch richtig) und «valid» (strukturkonform bezüglich einer DTD) sind. Ein gültiges Dokument ist immer wohlgeformt, ein wohlgeformtes Dokument aber nicht immer gültig.

[4] http://www.w3.org/XML/1998/06/xmlspec-19980910.dtd

[5] Weitere Informationen: XML/EDI Group Home Page URL: http://www.xmledi.net/

[6] http://www.w3.org/TR/WD-xsl

[7] http://www.w3.org/TR/xslt

[8] http://www.w3.org/TR/WD-xlink

[9] http://www.w3.org/TR/WD-xptr

[10] http://www.w3.org/TandS/QL/QL98/pp/xql.html

[11] http://www.xml.com/pub/Guide/Query_Technologies

[12] http://www.w3.org/TR/NOTE-xml-schema-req

[13] http://www.xml.com/pub/Guide/Schema

[14] http://www.xml.com/pub/1999/07/schemas/syntax.html

[15] http://www.w3.org/DOM/

[16] http://www.w3.org/XML/

[17] http://www.xml.com/pub/stdlist

[18] http://www.oasis-open.org/cover/KonaProp970718.html

[19] http://www.xml.com/pub/w3j/s3.rustintro.html

[20] http://www.w3.org/TR/MathML2

[21] http://www.w3.org/TR/PR-rdf-schema

[22] http://www.w3.org/TR/REC-smil/

[23] http://www.wapforum.org/what/technical.htm

[24] http://204.112.55.140/BIOML/Examples/insulin3.html

XML – première pierre d'un échange de données sans frontières

Dans les coulisses de la technologie et des applications

Plus besoin d'expliquer aux experts du Web ce qu'est XML. La recommandation de W3C pour l'Extensible Markup Language date de plus de deux ans, ce qui est – dans l'univers du Web – relativement long. Il en va cependant autrement dans le monde traditionnel de l'informatique, dans le monde des grosses bases de données et des machines encore plus grosses. Ici, on commence à peine à remarquer XML, bien que ces secteurs profiteront encore plus de XML que l'univers du Web. Cet article brosse un tableau des caractéristiques, aptitudes et champs d'application de XML ainsi que des sous-standards associés tels que XSL, XQL, etc.