Metadaten

Rolf Assfalg

Duale Hochschule Baden-Württemberg Heidenheim
Studiengang Informatik
assfalg@dhbw-heidenheim.de

1 Einführung

Betrachtet man Datensätze in relationalen Datenbanksystemen, Ausprägungen gängiger XML-Anwendungen oder Referenzdatenbestände im Bereich Information und Dokumentation (IuD), fällt eine wichtige Gemeinsamkeit auf: Diese Bestände benötigen eine Beschreibung ihrer inneren Struktur. Bei diesen Strukturbeschreibungen handelt es sich also sozusagen um „Daten über Daten“ und diese können kurz gefasst auch als *Metadaten* bezeichnet werden. Hierzu gehören Syntaxelemente und ggf. eine Spezifikation, wie diese Syntaxelemente angewendet werden.

Der Anwendungszweck bzw. die Perspektive des Betrachters ist jedoch ausschlaggebend, ob es sich jeweils um Daten oder um Metadaten handelt. Abb. 1 illustriert dies: Ein klassisches Referenzretrievalsystem enthält Verweise auf Literatur, die vielleicht in gedruckter Form vorliegt und in einer traditionellen Bibliothek aufgestellt sein mag. Für die Benutzer, die als Rechercheure nach Literatur suchen, sind diese Biliotheksbestände dann das Datenmaterial. Die Datensätze des *Online-Public-Access*-Systems (OPAC) sind für Bibliotheksbenutzer dann also Metadaten. Aus der Sicht der Entwickler- oder der Server-Betreiber des Online-Public-Access-Systems sind die in ihrer Datenbank gespeicherten Datensätze keine Metadaten, sondern aus deren Sicht befinden wir uns auf Datenebene. Die hierbei häufig eingesetzten relationalen Datenbanksysteme strukturieren diese Daten in Form von Tabellen, deren Spalten in einer bestimmten Abfolge und unter Beachtung bestimmter Domänen (Zeichenketten, Zahlen, Datum, Uhrzeit, usf.) gegliedert sind. Diese Strukturbeschreibung wird für Relationale Datenbanksysteme gemeinhin als Schema bezeichnet. Aus Informationssystemsicht sind Festlegungen des Datenbankschemas die Metadaten für die in der Datenbank gespeicherten Datensätze.

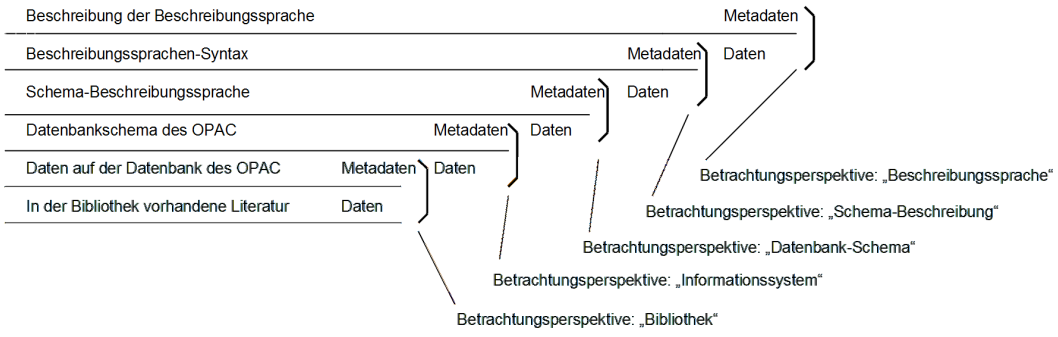

Abbildung. 1: Beispiel unterschiedliche Betrachtungsperspektiven ergeben jeweils spezifische Unterscheidungen zwischen Daten und Metadaten.

Für relationale Datenmodelle gibt es eine Vielzahl von Beschreibungsmöglichkeiten. Traditionell können Datenbankschemata innerhalb des Definitionsteils der *Sequential Query Language* (SQL) ausgedrückt werden. Aus Sicht des Entwicklers eines SQL-Interpreters handelt es sich bei diesem schemabeschreibenden SQL-Code um ein Artefakt der Datenebene. Die Regeln über den Aufbau und die Kombinationsmöglichkeiten der SQL-Anweisungen können in einer Syntaxbeschreibung dargestellt werden. Die gängigste Form ist hierfür die Backus-Naur-Form (BNF), auf die wir im folgenden Abschnitt näher eingehen werden. In unserem Beispielszenario bildet eine BNF-Beschreibung der SQL-Syntax die Metaebene der SQL-Datenstrukturbeschreibungen und sozusagen die Meta-Meta-Ebene der in der Datenbank gespeicherten Daten.

Zusammenfassend entscheidet aus informationswissenschaftlicher Perspektive die Sicht auf die pragmatische Ebene, ob es sich bei Artefakten um Daten oder um Metadaten handelt.

2 Beispiele für Ausprägungen von Metadaten in Informationssystemen

2.1 Die Backus-Naur-Form (BNF) – Eine Metabeschreibungssprache

Die BNF dient als Metasprache zur Darstellung kontextfreier Grammatiken. Sie wurde im Zusammenhang mit der formalen Beschreibung der Programmiersprache IAL von J.W. Backus (Lit. 1, S. 12ff) eingesetzt, deren Weiterentwicklung heute als ALGOL 60 bekannt ist. Schon damals hatte Backus erkannt, dass für eine Programmiersprache zum einen eine präzise Beschreibung der Symbole und ihrer Sequenzierung notwendig ist, damit die Übersetzbarkeit eines Programms in Maschinencode eindeutig ist und diese nicht von irgendwelchen zufälligen Eigenschaften einer konkreten Implementation eines Übersetzerprogramms (Compiler) abhängig sein darf. Zum anderen dürfen die Anweisungen eines Programms und deren Kombination keine semantischen Interpretationsspielräume bieten. Ein Stück Quellcode muss nach Übersetzung bei der Ausführung stets dasselbe Verhalten aufweisen, auch wenn der ausgeführte Maschinencode mit unterschiedlichen Übersetzungsprogrammen erzeugt worden sein sollte. (Lit. 1, S. 15)

Aus dieser Betrachtung lassen sich bereits zwei wesentliche Grundeigenschaften von Metadaten ablesen:

- Daten werden durch eine Metadatenbeschreibung portabel.
- Die Interpretation der Daten ist nicht abhängig von Eigenschaften des Rezipienten.

Ausdrücke in Backus-Naur-Form werden in textueller oder in graphischer Form niedergeschrieben. Wenden wir uns zunächst der textuellen Aufschreibungsvariante zu, die wir in informaler und intuitiver Form einführen wollen. Zunächst sei als Beispiel ein einfaches Signatursystem einer kleineren Bibliothek gegeben, welches wir uns anhand einiger weniger Beispielsignaturen klarmachen wollen:

INF/43/a	Informatik->Programmiersprachen->Java
W/2a/a	BWL->Grundlagen (Nachschlagewerke) ->Recht
MB/3/m	Maschinenbau->Mechanik->Theoretische Mechanik
MB/3	Maschinenbau->Mechanik (allgemein)

Die Definition dieses Signatursystems würde dann in Backus-Naur-Form (BNF) wie folgt aussehen:

```
<Großbuchstabe>     ::=   A | B | C | … | X | Y | Z
<Kleinbuchstabe>    ::=   a | b | b | … | x | y | z
<Zahl>              ::=   1 | 2 | 3 | …
<Signatur>          ::=   <Hauptgruppe>"/"<Gruppe>["/"<Untergruppe>]
<Hauptgruppe>       ::=   <Großbuchstabe>[< Großbuchstabe >][< Großbuchstabe >]
<Gruppe>            ::=   <Zahl>[<Kleinbuchstabe>]
<Untergruppe>       ::=   <Kleinbuschstabe>
```

Eine Backus-Naur-Definition ist also eine Ansammlung metalinguistischer Formeln, wobei der Operator „::=“ als *ist definiert durch* oder *ergibt sich aus* zu lesen ist. Die Ausdrücke in spitzen Klammern sind metalinguistische Variablen bzw. Kategorienamen, die hier nicht mit XML-Tags verwechselt werden dürfen und die – einmal definiert – in weiteren nachfolgenden BNF-Ausdrücken wiederverwendet werden können. Der senkrechte Strich „|“ ist als „oder“ zu lesen. Zeichen, die - wie der Schrägstrich in unserem Beispiel - als Literal direkt übernommen werden sollen, werden in Hochkommas gesetzt. Teile mit optionaler Verwendung werden in eckige Klammern „[…]“ gestellt. Von Vorteil ist auch der Umstand, dass BNFs die Bezeichner der grammatikalischen Bestandteile wie *Hauptgruppe*, *Gruppe* und *Untergruppe* gleich mitdefinieren.

Aus obiger Backus-Naur-Definition ergeben sich nun folgende Eigenschaften unseres fiktiven Signatursystems: Die Hauptgruppen werden mit ein-, zwei- oder drei aufeinanderfolgenden Großbuchstaben bezeichnet. Eine Gruppe ist mit einer Zahl bezeichnet, die ggf. mit einem angehängten Kleinbuchstaben erweitert werden kann. Die Untergruppe kann weggelassen werden, wohingegen Hauptgruppe und Gruppe im Rahmen der Sacherschließung jeder Dokumentationseinheit zwingend festgelegt werden müssten. Untergruppen werden immer mit einem einzelnen Kleinbuchstaben bezeichnet.

Im Übrigen lassen sich diese BNF Definitionen auch grafisch darstellen. Hierzu ein Beispiel in Abbildung 3. Die Pfeile symbolisieren den Lese- bzw. den Verarbeitungsfluss. Gabelt sich ein Pfeil, steht das für eine Weg-Alternative. Der zweite- und der dritte Großbuchstabe sind also jeweils optional, da sie sprichwörtlich *umgangen* werden können.

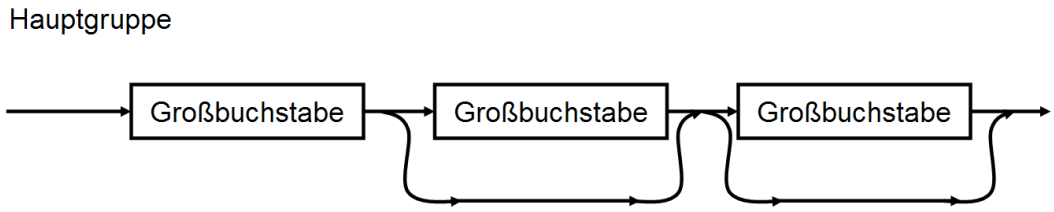

Abbildung 3: BNF-Beispieldefinition in grafischer Notation

BNF-Definitionen können in einer Weise erweitert werden, dass bereits bestehende Daten trotzdem gültig bleiben. Soll beispielsweise die Untergruppe mit zusätzlichen Buchstaben erweitert werden, um Signaturen wie „MB/3/ma“, „MB/3/mb“, „MB/3/mca“ usf. bilden zu können, genügt folgende Änderung:

```
<Untergruppe>       ::=   <Kleinbuchstabe> | <Untergruppe><Kleinbuchstabe>
```

Diese Definition gestattet mit ihrer rekursiven Struktur im Prinzip beliebig lange Aneinanderreihungen von Kleinbuchstaben als Untergruppen-Identifikatoren, wobei bestehende Signaturen, wo nur ein Buchstabe vorgesehen ist und die noch der vorhergehenden Definition folgen, nach wie vor gültig bleiben.

Im Laufe der Zeit setzten sich bestimmte Modifikationen und Erweiterungen der BNF durch. So wurde z.B. das Syntaxelement „{… }“ eingeführt, mit dem sich beliebig häufige Wiederholungen ausdrücken lassen. Die letztgenannte Definition von <Untergruppe> lässt sich hiermit eleganter, kürzer und verständlicher formulieren:

```
<Untergruppe>       ::=    {<Kleinbuchstabe>}
```

Diese formal dargestellte Beliebigkeit der Anzahl der Wiederholungen ist allerdings für Metadaten-Definitionen in praktischer Hinsicht ein Problem. So gesehen ist auf Datenebene nicht alles praktikabel was laut einer BNF-Definition der Metaebene gültig wäre. Es können eben lediglich nur praktikable Datenbestände auf Gültigkeit überprüft werden.

2.2 Taxonomien als Metadaten

Nachdem wir uns im Zusammenhang mit der Beschreibung von Syntaxstrukturen über die Konstruktion eines prototypischen Signatursystems den Taxonomien genähert haben, wollen wir hier noch etwas detaillierter darauf eingehen.

Historisch betrachtet, etablierten sich erste Metabeschreibungen für Dokumente in Bibliotheken und an der Schwelle zur Neuzeit stand hier zunächst die Erfassung der formalen Aspekte wie Autor, Titel und Jahr im Vordergrund. Mit dem Aufstellungsort konnte allenfalls entweder die Dokumentart oder ein einzelner inhaltlicher Aspekt verbunden werden. Nach der Erfindung des Buchdrucks ergab sich in Bibliotheken jedoch eine erhebliche Mengenzunahme und somit entstand das Bedürfnis, den Benutzern auch Instrumente zur thematischen Suche an die Hand zu geben. G.W. Leibniz entwickelte am Ende des 17. Jahrhunderts in seiner Rolle als kurfürstlicher Hofhistograph für die Herzog-August-Bibliothek in Wolfenbüttel einen alphabetischen Katalog und vor allem eine *„Dezimalklassifikation“*, die als eine universelle Systematik zur Einteilung und Beschreibung von Wissen dient. Auf oberster Ebene gibt es zehn Grundkategorien, die jeweils in zehn weitere darunterliegende verfeinert werden. Auch diese Unterkategorien können wieder in bis zu zehn weitere Unter-Unterkategorien verzweigt werden. Nach Schwed (Lit. 15, S112) handelt es sich um eine analytische Klassifikation, die vom Allgemeinen zum Speziellen verzweigt. Sie präkoordiniert das Wissen monohierarchisch, d.h. es gibt jeweils nur einen Oberbegriff. Dieses Klassifikationsschema ist, im Gegensatz zu später aufkommenden Konzepten, nicht genormt und kann somit frei an die jeweilige Systemsprache angepasst werden. Die Notationen bestehen überwiegend aus Ziffern und Zeichen. Die Grundform der Dezimalklassifikation wurde Jahre 1876 von Dewey erweitert und ist vor allem im angloamerikanischen Sprachraum unter der Bezeichnung *„Dewey-Dezimalklassifikation“* (DDC) bekannt. Sie ist immer noch in vielen Bibliotheken gebräuchlich und wird von OCLC laufend weiterentwickelt. (vgl. Lit. 15, S.112f) Hierzu ein Beispiel: Die Grundkategorie „5“ steht in der DDC für „Naturwissenschaften“. Die Unterkategorie „52“ steht für „Astronomie“ und „527“ steht für „Astronavigation“.

Die Internationale Patentklassifikation (IPC) ist ein ebenfalls dokumentbezogener Standard, um Patente thematisch zu verorten und sie dient sowohl der inhaltlichen Abgrenzung von Schutzinteressen als auch dem Auffinden von Patenten im Rahmen einer Patentrecherche. Ein Vorteil besteht darin, dass die IPC-Kürzel in vielen Sprachen sorgfältig übersetzt zur Verfügung stehen. Damit umreißt die IPC faktisch auch den Wortschatz für Übersetzungen im Bereich der gewerblichen Schutzrechte.

Auch Nichtdokumente können mit Klassifikationssystemen kategorisiert werden. Die International Classification of Deseases (ICD) unterteilt Diagnosen, Erkrankungen und Todesursachen. Die ICD-Kürzel finden sich auf vielen Artefakten, die im Rahmen der medizinischen Dokumentation und –Abrechnung anfallen. Die Krankengeschichte eines Patienten kann als chronologische Abfolge von ICD-Kürzeln aufgefasst werden. Auch die ICD ist in viele Sprachen übersetzt.

Taxonomien haben allerdings folgende Eigenschaften:

- **Begriffliche Beharrung:** Einmal eingeführte Begriffe wandeln ihre Bedeutung oder die betroffenen Themengebiete werden mit der Zeit anders bezeichnet. Ein Klassifikationssystem kann aber nicht ohne weiteres umgebaut werden.
- **Neue Begriffe fehlen zunächst:** Bei neu aufkommenden Kategorien vergeht meist eine gewisse Zeit, bis das Klassifikationssystem aktualisiert wird. Zwischenzeitlich vorgenommene Klassierungen verwenden dann noch nicht die neu eingeführten Begriffe und sind somit suboptimal.
- **Unausgewogene Hierarchietiefe:** Nach längerem Gebrauch können sich starke Ungleichgewichte in der Anzahl neu eingerichteter Kategorien und deren Verzweigungstiefe ergeben.
- **Singulärer Kontext:** Die Einordnung in eine Kategorie verengt den Klassierungsgegenstand auf einen einzigen Kontext. Sind mehrere Kontexte zu berücksichtigen, kann dies nur durch dessen Mehrfacheinbettung bewältigt werden. Für Signatursysteme in Freihandbibliotheken könnte dies heißen, dass ein Buch mehrfach beschafft werden müsste, um es in mehreren Signaturbereichen gleichzeitig auffinden zu können.

Aus diesen Gründen ist zur inhaltlichen Erschließung von Dokumentationseinheiten die Einführung eines Klassifikationssystem alleine nicht ausreichend. Neben den Angaben aus der Formalerschließung empfehlen sich kontrollierte Vokabularien (Thesauri) und frei vergebene Schlagworte. Dies wird als auch als *„Hybriderschließung“* bezeichnet.

Taxonomien können im übrigen auch automatisch erzeugt werden. Bereits Salton (siehe Lit. 16, S. 228ff) hat ein Verfahren beschrieben, wie aus Dokumentansammlungen *„Cluster“* gebildet werden können. Der Algorithmus fasst in mehreren Iterationen nacheinander kleinere Cluster zu immer größeren zusammen, so dass eine hierarchische Struktur entsteht. Voraussetzung sind Ähnlichkeitsmetriken zwischen Dokumenten, wie sie z.B. im Salton’schen Vektormodell ohnehin vorhanden sind.

2.3 Metadaten für bibliografische Inhalte

Vor der Einführung der Datenverarbeitung wurden Metadatenbestände in Bibliotheken in Form von Zetteln oder Karteikarten geführt. Deren inhaltliche Gestaltung unterlag schon im 19. Jahrhundert gewissen Regelwerken, wie z.B. der *„Preussischen Instructionen“*, deren Weiterentwicklung und spätere Computerisierung in standardisierte Beschreibungsverfahren

wie z.B. der *International Standard Book Description (ISBD)* mündete. Die technische Herausforderung im Rahmen der Informatisierung der Bibliotheken bestand dann vor allem darin, die Daten per OCR-Software zu dekodieren und diese auf Rechnersysteme zu transferieren. Die konsequente Einhaltung einer Syntaxstruktur ist für die korrekte Aufschlüsselung der Daten hier unerlässlich.

Mit der Einführung klassischer Online-Referenzdatenbanken hat sich losgelöst von Bibliotheksstandards ein Konsens einer Datensatzstruktur entwickelt, die üblicherweise folgende Felder umfasst: Autor (AU), Titel (TI), Quelle (SO), Publikationsjahr (PJ), Publikationssprache (LA), Abstract (AB), Klassifikationscode (CC), Thesaurusbegriffe (CT), Zusätzliche Schlagworte (ST).

Der hier besprochenen Formate orientieren sich aber an traditionellen Textualitätsbegriffen und stoßen bei hypertextualisierten Ressourcen wie sie z.B. im Zusammenhang mit der Metabeschreibung von Web-Dokumenten notwendig werden, an ihre Grenzen. Semantic Web verfolgt die Idee, nicht nur eine Abfolge gleichartig strukturierter Werke zu darzustellen. Mit Semantic Web sollen auch Hyperlinks, Personen, Organisationen, usf. und deren Beziehungen untereinander beschrieben werden. (vgl. B12 Semantic Web in diesem Band). Allerdings macht die Textualität von Hypertexten die Inhalte für Benutzer auf verschiedene Weise erfahrbar, je nachdem über welchen Weg ein Benutzer auf einen bestimmten Knoten navigiert. Genau hier stoßen Metabeschreibungen für bibliografische Zwecke an ihre natürliche Grenze.

2.4 Metadaten per Textauszeichnung - Von SGML zu TEI

In den 70er und 80er Jahren ergab sich durch die zunehmende Informatisierung des Drucklegungsprozesses das Bedürfnis nach besseren Schnittstellen zwischen Autoren und Verlagen. Den Autoren sollten Möglichkeiten an die Hand gegeben werden, inmitten ihrer Texte makrotypografische Metainformation zu hinterlegen. Überschriftengrade, Absatzgliederung, aber auch referenzielle Verknüpfungen zur Literatur oder vom Register in den Text sollten möglich sein. Die *Auszeichnung* des Texts sollte mit lesbaren Zeichen- und Zeichenkombinationen geschehen, die in Texten normalerweise so gut wie gar nicht vorkommen, aber mit jeder beliebigen Tastatur realisierbar sein müssen. So kam mit der *Standard Generalized Markup* Language (SGML) eine Lösung zustande, die Metainformation und Information in einem vom Autor verfassten Text vereint. Die Textauszeichnung mit sogenannten Tags (wie z.B. „“ bzw. „“) ist sozusagen der Metadaten-Anteil eines Texts. Hierzu ein Beispiel:

```
<h1>Kapitelüberschrift</h1>
<p>Ich bin ein Absatz in einem Kapitel.
Manche Wörter können <b>hervorgehoben</b> werden. </p>
<img src="/bild.jpg" alt="mein aktuelles Bild"/>
<p>Und nach dem Bild beginnt ein neuer Absatz… </p>
```

(Hierbei steht „`<h1>`“ für *Heading ersten Grades*, „`<p>`“ steht für *Paragraph* und „`<b>`“ für *bold*.) Nutzte man in der Anfangszeit noch Beispiele und Spezifikationstexte zur Beschreibung dieser Metaanteile, kam mit der maschinellen Verarbeitung der Textauszeichnung das Bedürfnis auf, diese formal zu definieren, um die Validität des autorenseitig gelieferten- und ausgezeichneten Texts im Rechner zu verifizieren. Hierfür wurde die *Document Type Definition* entwickelt, die Metabeschreibungen für die Textauszeichnung in einer eigenständigen und recht einfachen Syntax ermöglicht. Hier ein

Beispiel zur Definition des obigen „<img…“-Tags auf Metaebene (gekürzt):

```
<!ELEMENT IMG - O EMPTY                -- Embedded image -->
<!ATTLIST IMG
  %attrs;                              -- %coreattrs, %i18n, %events --
  src         %URI;          #REQUIRED -- URI of image to embed --
  alt         %Text;         #REQUIRED -- short description --
  …usf…
>
```

Eine Anwendung kommt also zustande, wenn Anwendungsdateien und die einmal erstellte und wiederverwendbare DTD zusammen verarbeitet werden.

Die *Text-Encoding-Inititiative* (TEI) ist eine Weiterentwicklung der SGML-Ideen mit dem Ziel so viele strukturelle Elemente unterschiedlicher Textsorten wie möglich in DTDs oder XML-*Namespaces* zu repräsentieren. Es stehen vor allem geisteswissenschaftliche Textkodierungsprojekte im Vordergrund (Lit. 14).

2.5 HTML

Die Idee einer unabhängigen Metabeschreibungssprache kommt spezifischen fachlichen Bedürfnissen entgegen. So können Anwender eigene DTDs entwickeln und damit Tag-Sprachen erfinden, die Textauszeichnungen für ihre speziellen Bedürfnisse ermöglichen. Die Entwickler des *World-Wide-Web* folgten genau diesem Weg und entwarfen 1989 eine DTD zur Definition der *Hypertext-Markup-Language* (HTML). Leider hat sich die Vereinigung von Daten und Metadaten in jeweils einer einzigen Datei nicht bewährt. Vor allem Layout-Informationen, Design-Festlegungen und referenzielle Verknüpfungen können nicht als eigenständige und typisierte Objekte verwaltet werden. Nur mit aufwändigen *Web-Content-Management*-Systemen, lassen sich heutzutage Inhaltsdaten, also der *Content* mit seinen Hypertext-Verknüpfungen, und die Metadaten in Form von Templates für Web-Anwendungen sinnvoll voneinander trennen. Diese Systeme können dabei zusätzlich die Plausibilität der Linkstruktur gewährleisten. HTML-Dokumente werden sozusagen dynamisch und erst im Moment des Abrufs erzeugt. Es bleibt zu bedauern, dass die seinerzeit schon verfügbaren Erkenntnisse der Hypertextforschung im Wold-Wide-Web-Konzept keinerlei Berücksichtigung fanden.

2.6 XML

XML ist eine Nachfolgeentwicklung von SGML. Die Idee der Textauszeichnung hatte mit dem Aufkommen des Internets neue Anwendungshorizonte vor allem als Datenübertragungs- oder Speicherformat erschlossen. Neue Bedürfnisse waren eine verbindliche Festlegung, wie eine Anwendungsdatei auf ihre zuständige Meta-Beschreibungsdatei (DTD) Bezug nimmt,oder wie XML-Daten-Dateien auf mehrere Metabeschreibungen gleichzeitig Bezug nehmen können. Ein einzelnes XML-Tag ist grundsätzlich einem *Namespace* zugeordnet. Hierzu ein Beispiel: Gegeben sei folgendes *Tag*: <xs:attribute name="src" type="xs:string"/> Der Tagname „xs:attribute“ ist durch den Doppelpunkt in den *Namespace-Identifier*-Präfix und den Tag-Namen innerhalb dieses mit „xs“ benannten Namespace unterteilt. Auf diese Art können selbst zufällig namensgleiche *Tags* verschiedener *Namespaces* eindeutig ihren zuständigen Metabeschreibungen zugeordnet werden. Im Dokument-Kopf wird mit „xmlns:xs="…"“ für „xs“ die Metabeschreibung angeschlossen. Damit ältere Dokumente aus der SGML-Zeit (z.B. HTML-Seiten) kompatibel zu XML bleiben, kann für XML-Dokumente ein *Default-Namespace* benannt werden, für die dann gewissermaßen eine *Default-DTD* herangezogen ist. In diesem Fall kann der *Namespace*-Präfix weggelassen werden und gängige HTML-Tags wie z.B.: „<html:body>“ können wie schon 1989 immer noch als „<body>“ niedergeschrieben werden.

Die einfache DTD-Syntax ist für die Textauszeichnung geeignet. Jedoch sind nach Harold und Means (Lit. 7, S.5) Definitionen wie: *Dieses Element enthält eine Zahl* oder *diese Zeichenkette repräsentiert ein Datum zwischen 1974 und 2032* leider nicht möglich. Die W3C XML-Schema-Language erlaubt es als Nachfolger der DTD solche Einschränkungen zu spezifizieren (Lit. 7, S.5). Ein XML-Schema folgt im Unterschied zur DTD keiner eigenständigen Syntax, sondern ist selbst als XML-Dokument niedergeschrieben. Folgender gekürzter Auszug aus einem XML-Schema-Dokument, welches das oben vorkommende „<img…“-Tag spezifiziert, soll einen Einblick geben:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
...
 <xs:element name="img">
 <xs:complexType>
  <xs:complexContent>
   <xs:restriction base="xs:anyType">
    <xs:attribute name="src" type="xs:string"/>
    <xs:attribute name="alt" type="xs:string"/>
   </xs:restriction>
  </xs:complexContent>
 </xs:complexType>
 </xs:element>
...usf...
</xs:schema>
```

Ein XML-Schema-Dokument ist also ein XML-Dokument, das eine Metabeschreibung für andere XML-Dokumente vorhält. Interessant ist hier der Umstand, dass die Metabeschreibung von XML-Schema-Definitionen wiederum auch als XML-Schema vorliegt. So ist die Datei http://www.w3.org/2001/XMLSchema.xsd eine XML-Schema-Beschreibung der XML-Schema-Beschreibungssprache. Man könnte hier also sozusagen von Meta-Meta-Daten-Definition sprechen.

Metadaten begegnen uns in der XML-Welt noch in anderer Hinsicht, und so können uns XML-Dokumente auch in der Rolle einer Metadaten-Beschreibung für Dokumentationseinheiten begegnen. *Semantic-Web*-Dokumente beschreiben die Inhalte von regulären Dokumenten wie z.B. gedruckte Werke, Web-Dokumente oder XML-Dokumente. *Semantic-Web*-Dokumente sind das Ergebnis einer speziellen Form der inhaltlichen Erschließung, wo ontologisch aufbereitetes Wissen repräsentiert wird (vgl. B12 Semantic Web in diesem Band).

Was ist nun XML? – Im Grunde ist nur wenig vorgegeben: Der Kopfbereich mit seinen *Namespace*- und Syntaxdefinitionsdateiverweisen regeln das prinzipielle Aussehen von Tags und wie Tags korrekt geschachtelt werden. Bei Einhaltung dieser allgemeinen XML-Eigenschaften spricht man von *Wohlgeformtheit*. Alles andere wird von Anwendern in den Metabeschreibungssprachen DTD oder XML-Schema definiert. Folgt ein wohlgeformtes XML-Dokument diesen Festlegungen auf Metadaten-Ebene, spricht man von einem *gültigen* XML-Dokument.

2.7 Die Dublin Core Metadata Initiative

Aus Sicht der Information und Dokumentation kommt den Metadaten vor allem die Rolle der inhaltlichen Erschließung zu. Hier ist der Standard der *Dublin-Core-Metadata-Initiative* (DCMI) eine nähere Betrachtung wert, der zur inhaltlichen Erschließung informationeller Einheiten dient. Die Idee zu Dublin Core kam im Rahmen einer informellen Diskussion auf

einem OCLC/NCSA Workshop Jahr 1995 in Dublin (DC 1), auf. Dort wurden zunächst 13 Eigenschaftselemente festgelegt : DC.Title, DC.Autor, DC.Subject, DC.Publisher, DC.OtherAgent, DC.Date, DC.Object, DC.Type, DC.Form, DC.Identifier, DC.Relation, DC.Source, DC.Coverage, DC.RightsManagement. Diese bilden den ursprünglichen *Dublin Core Metadata Element Set* (Lit. 9). Auf HTML-Seiten können im Kopfbereich – also konkret im Bereich zwischen „<Head>“ und „</Head>“ - Meta-Tags untergebracht werden, die beispielsweise folgendermaßen aussehen können:

```
<meta name="DC.Publisher" content="Rolf Assfalg" >
```

Später wurde ein XML-*Namespace* definiert und auf 15 Elemente erweitert: Contributor, Coverage, Creator, Date, Description, Format, Identifier, Language, Publisher, Relation, Rights, Source, Subject, Title, Type (Lit. 6). Die *Dublin Core Metadata Initiative* hat hierzu (in Lit. 12) *Levels of interoperability* benannt. Die Ebenen bauen aufeinander auf und sind natürlich auch der Rückwärtskompatibilität bereits vorhandener Anwendungen geschuldet, die auf früheren Versionen des DC-Standards basieren. Mit jeweils höheren Levels sollen Dublin-Core-Meta-Auszeichnungen in jeweils elaborierterer Form maschinell verarbeitbar sein. Hier die Levels (entlang Lit. 12) im Einzelnen:

- **Level 1: *Shared term definitions*:** Im Grunde müssen hier nur die oben erwähnten Eigenschaftselemente verwendet werden. Eine URI-Festlegung für die Dokumente ist nicht vorgesehen. In jedem Fall ist dieses *Interoperability Level* auch für den menschlichen Leser geeignet.

- **Level 2: *Formal Semantic interoperability*:** Den Hintergrund bildet ein auf RDF basierendes Semantic-Web, wobei die DCMI Metadaten-Terme innerhalb von <rdf:Description> formuliert werden und als separate RDF-Dateien abgelegt sind. Folgendes Beispiel aus Johnston (Lit. 8 zit. in 10) sei hierzu wiedergegeben, bei der eine RDF-Datei eine Anwendungsdatei gleichen Namens inhaltlich auszeichnet und die nur dazu dient, den Dokumenttitel auszuweisen:

  ```
  <?xml version="1.0" encoding="UTF-8" ?>
  <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="">
     <dc:title>Services to Government</dc:title>
  </rdf:Description>
  </rdf:RDF>
  ```

- **Level 3: *Description Set syntactic interoperability*:** Auf diesem Level gilt das Dublin Core Metadaten-Modell, das die innere Struktur von Metadaten-Elementen explizit vorgibt. Im Gegensatz zum vorhergehenden Level 2, wo unterhalb von „<rdf:description>“ nur die 15 Metadatenelemente kommen können, sind hier ontologische Strukturen möglich. Dazu gibt es noch eine textuelle Dublin Core Repräsentation, die als „DC-Text“ bezeichnet wird und die eine eigene Syntax hat. Hierzu ein Beispiel aus (Lit. 10):

  ```
  @prefix dcterms: <http://purl.org/dc/terms/> .
  @prefix ex: <http://example.org/taxonomy/> .
  DescriptionSet (
   Description (
    ResourceURI ( <http://example.org/123> )
    Statement (
     PropertyURI ( dcterms:subject )
     VocabularyEncodingSchemeURI ( ex:MyVocab )
     ValueString ( "Ornitology" )
  ```

)
)
)

- **Level 4*: Description Set Profile interoperability*:** Es gibt strukturelle *Constraints* für die Anwendung der DC-Text-Syntax im Rahmen sogenannter *Description Set Profiles* wie sie von Nilsson (Lit. 11) spezifiziert wurden. Die Entwicklung befindet sich aber derzeit noch in der Konzeptionsphase.

2.8 Metadaten in Relationalen Datenbanken

Anwendungs- und Informationssysteme sind in der Regel in mehrschichtigen Architekturen aufgebaut. Die oberste Schicht ist die Präsentationsschicht, in der Mitte befindet sich die *Geschäftslogik* und auf unterster Ebene befindet sich das Datenbanksystem. Als Metadaten für das Datenbanksystem ergeben sich zweierlei Betrachtungen: Zum einen das Schema, das die Tabellenstruktur und die Beziehungen zwischen den über diese Tabellen verteilten Entitäten entlang ihrer Kardinalitäten beschreibt. Zum anderen gewissermaßen die Detailsicht der Daten, die im Rahmen eines *Data-Dictionary* genauere Festlegungen zu syntaktischen und semantischen Zusammenhängen innerhalb der in den Tabellen bereit gestellten Feldern ermöglicht.

Im ersten Abschnitt wurde bereits darauf hingewiesen, dass das Datenbank-Schema in SQL repräsentiert werden kann. Allerdings ist diese Darstellungsform für Entwurfszwecke nicht sehr geeignet. Betrachten wir zunächst das folgende Diagramm in Abbildung 2. Hierbei handelt es sich um einen Schemaentwurf in Form eines *Entity-Relationship*-Diagramms, hier in der ursprünglichen Darstellungsform nach Peter Chen (Lit. 3), wo Entitätsmengen als Rechtecke- und Beziehungen (*relations*) als Rauten dargestellt sind. Das Beispiel impliziert insgesamt drei Datenbanktabellen und für die *lieferbar von*-Beziehung noch eine Beziehungstabelle, die aus der n:m-Kardinalität der Beziehung resultiert. Die anderen Beziehungen sind als Fremdschlüsselverweise in den Datensätzen darstellbar. Anzahl und Ausprägung der konkret in diesen Tabellen gespeicherten Datensätze sind beliebig, was den Metacharakter des Diagramms untermauert.

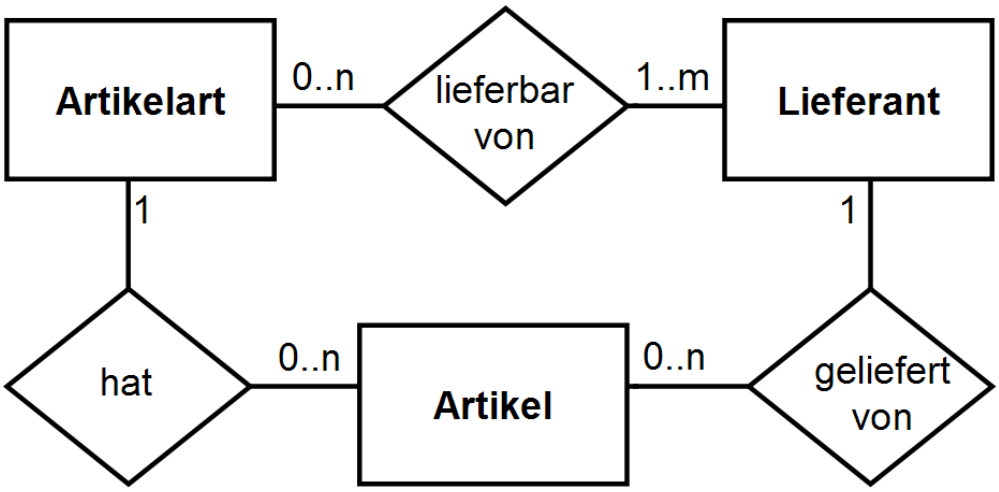

Abbildung 2: Beispiel eines Datenbankschemas

Das Diagramm ist wie folgt zu lesen:

- Eine Artikelart kann von mindestens einem Lieferant geliefert werden und ein Lieferant kann keinen, einen oder mehrere Artikel liefern.

- Ein Artikel ist immer einer Artikelart zugeordnet und eine Artikelart kann keine, einen oder mehrere (oder viele) Artikel-Ausprägungen haben. (Im Versandhandel wäre Artikelart das, was ein Kunde mit einer Bestellnummer angibt.)

- Ein Artikel wurde konkret von einem bestimmten Lieferant geliefert und ein Lieferant, hat entweder noch keinen, oder einen oder bereits viele Artikel geliefert. (Hierdurch können z.B. Garantiefälle an den Lieferanten weitergegeben werden.)

Viele Vorgehensmodelle der Software-Entwicklung gehen davon aus, dass für Anwendungssysteme zu einem sehr frühen Zeitpunkt Datenmodelle dieser Art entworfen werden. Laufen auf Präsentations- und Geschäftslogik-Schicht bereits Programme, die auf dem so definierten Schema aufsetzen, und sind bereits viele Test-Datensätze auf der Datenbank gespeichert oder ist das System bereits gar im Produktiveinsatz, sind größere Änderungen der Software aufwändig und teuer. Aus diesem Grund kommt der entwurfsmotivierten Meta-Betrachtung im Bereich der Software-Entwicklung ein vergleichsweise hoher Stellenwert zu. Sie zu beherrschen, gehört heutzutage zu den Kernkompetenzen eines Software-Entwicklers.

Ein Data-Dictionary, welches eine Detailsicht der Daten auf Metaebene beschreibt, wird dann relevant, wenn ein Softwaresystem dokumentiert wird, bevor Daten ausgetauscht werden, oder wenn die Datenbank-Inhalte genau spezifiziert werden, um sie mit höheren Schichten in Einklang zu bringen, was auch für das automatische Testen relevant werden kann. Ein *Data-Dictionary* kann als separates Artefakt im Software-Entwicklungsprozess – also im einfachsten Fall als Dokument – realisiert sein. Es kann aber auch als Systembaustein eines Datenbanksystems oder eines Entwicklungsframeworks existieren und funktioniert dann mindestens halbautomatisch.

Die in Datenbanktabellenfeldern gespeicherten Daten sind in nur sehr grobe Domänenbereiche wie *ganze Zahlen* (INTEGER), *Zeichenketten* (VARCHAR), *Datum* (DATE), usf. eingeteilt. Beispielsweise sei aus Sicht eines Datenbanksystems der *Familienstand* als Zeichenkette (VARCHAR) deklariert. Im Data-Dictionary lassen sich darauf aufbauend im einfachsten Fall diskrete Zeichenketten exklusiv festlegen, wie das folgende Beispiel von Balzert (vgl. Lit. 2) zeigt.

```
<Familienstand>    ::=    [ ledig | verheiratet | geschieden | verwitwet ]
```

Die BNF ist also auch im Data-Dictionary als Implementations- und Spezifikations-Metasprache gebräuchlich. Allerdings wurden im SQL2-Standard über sogenannte *Constraints* Möglichkeiten geschaffen, um diese Art von Definition oder Wertbereichsgrenzen gleich direkt in der Tabellendefinition festlegen zu können. Darüber hinausgehende Definitionen sind allerdings in SQL2 nicht möglich. In BNF können aber dann optionale Elemente und Zusammensetzungen von Datenelementen leicht spezifiziert werden. Ein einfaches und selbsterklärendes Beispiel verdeutlicht dies:

```
<Hausnummer> ::=    <Zahl>[<Kleinbuchstabe>]
```

2.9 Objektorientierte- und Objektrelationale Systeme

Das Konzept der Klassen und Instanzen bietet eine intuitive und leicht zu beherrschende Unterscheidung zwischen der Datenebene und der Metaebene. Die Instanzen befinden sich auf der Datenebene, und die Klassen bilden hierbei die Metaebene, auf der die Befüllung der Attribut-Werte auf Datenebene über Schnittstellen-Methoden geregelt werden kann. Hier ein Beispiel für eine Klasse zur Darstellung von Brüchen in der Programmiersprache Java:

```
class Bruch {
       private int zaehler;
       private int nenner;

       public void setNenner(int nennerUebergabe) {
              if (nenneruebergabe == 0)
                     throw new MeinEingabefehler("Nenner darf nicht Null sein");
              else
                     nenner = nennerUebergabe;
…usf…} …usf…
}
```

Zum Beschreiben der Instanz-Attribute sind sogenannte *Setter-Methoden* üblich. Das Attribut *Nenner* ist geschützt und kann nur über eine Nachricht gesetzt werden, welche die Ausführung der Methode *setNenner()* impliziert. Die dort vorhandene Programmierung ermöglicht das Setzen des Nenners für alle Werte außer der 0 und dies gilt gleichzeitig für alle Bruch-Objekte, die es in diesem so programmierten System geben kann. Weil die Ausdrucksmächtigkeit einer elaborierten Programmiersprache zur Verfügung steht, sind damit die kompliziertesten generischen Meta-Definitionen formulierbar. Durch Vererbungsmechanismen lassen sich zudem die Eigenschaften auf Metaebene gut organisieren.

Die Weiterentwicklung der *Sequential Query Language* hat um die Jahrtausendwende durch den SQL3-Standard zur Einführung objektorientierter Ideen in die Welt relationaler Datenbanken geführt. Eine sogenannte objektrelationale Datenbank gestattet nun – wie das folgende SQL3-Codebeispiel zeigt – die Einführung von Substrukturen wie sie bislang nur in BNF-basierten Data-Dictionaries möglich waren

```
create type PersonTyp (
              Name Row (Vorname varchar(15), Nachname varchar(20));
              Spitzname varchar(30),
              Ehepartner REF(Person),
              Kinder REF(Person)ARRAY[10],
              Function AnzahlKinder(PersonTyp) returns Integer
);
```

Aber auch Referenzen lassen sich nun typisieren, und Funktionsdefinitionen ermöglichen Schnittstellendefinitionen gewissermaßen auf Metaebene, ähnlich dem obigen Beispiel und wie sie in objektorientierten Systemen üblich sind. Da SQL3 rückwärtskompatibel angelegt ist, gab es nie den Zwang zur Anwendung dieser Konstrukte. Doch sind alle Altdaten bei der Einführung eines solchen objektrelationalen Datenbanksystems noch kompatibel. Dadurch führte allerdings die Etablierung objektrelationaler Systeme zu einem Absterben der zunächst hoffnungsvollen Epoche der rein objektorientierten Datenbanken, deren Neunutzung ggf. gravierende Altlastenproblem nach sich ziehen kann.

3 Heterogenität von Forschungsdaten und deren Metadatenebene

Um zunächst Anwendungsspektrum und Themenverteilung von Forschungsdaten kennenzulernen, sei hier eine kleine Zusammenstellung von Willis et al. (Lit. 13) wiedergegeben, die für eine Studie einer vergleichenden Betrachtung von Metadatenformaten zusammengetragen wurde und die einen repräsentativen Querschnitt bietet: CIF: Experimentaldaten zur Kristallographie (Kristallstrukturen); DarwinCore: Empirische Studien und Probensammlungen; DDI: Experimentalstudien, Empirische Studien und Statistische Studien im Sozialwesen; EML: Experimentalstudien und Empirische Studien in der Ökologie; MAGE, mmCIF und MINiML: Experimentelle Studien in Molekularbiologie (z.B. Molekülhäufigkeiten); NEML: Experimentelle Studien in Phylogenese (Phylogenetische Baumstrukturen); ThermoML: Experimentelle Studien zur Thermodynamik (Thermodynamische Eigenschaften). Wie man sieht, reicht die Palette also von tabellarisch darstellbaren Daten zur statistischen Analyse bis hin zu Strukturdaten, die als Graphen repräsentiert sind.

Eine sachgerechte Nutzung von Forschungsdaten setzt voraus, dass sie in standardisierter Form bereitgestellt, dokumentiert und ausreichend mit Metadaten versehen werden. Standards, Metadatenkataloge und Register sind unter Berücksichtigung fachspezifischer Anforderungen so zu entwickeln, dass auch deren interdisziplinäre Nutzung möglich ist. Auch in infrastruktureller Hinsicht bestehen Herausforderungen: Die Anforderungen müssen in Zusammenarbeit von Wissenschaftlerinnen und Wissenschaftlern mit Informationsspezialistinnen und Informationsspezialisten definiert werden. Infrastrukturen sind gemäß diesen Anforderungen zu entwickeln und möglichst von Beginn an in internationale und interdisziplinäre Netzwerke interoperabel einzubinden (Lit. 5). Willis et al. (Lit. 13, S.11) unterscheiden hierfür elf fundamentale Anforderungen für Metadatenbeschreibungen bei Forschungsdaten:

- **Schema Abstraktion:** Ein gut definiertes Metadaten-Schema sollte längere Zeit stabil bleiben. Abstraktion eröffnet Möglichkeiten zur laufenden Erweiterung, ohne Altbestände ungültig zu machen.

- **Schema Erweiterbarkeit, Flexibilität und Modularität:** Dies sind wesentliche Anforderungen für Informationssysteme, einschließlich der Metadaten und stellt die Langlebigkeit des Schemas sicher und gewährleistet, dass ständige Änderungen auch kurzfristig möglich sind.

- **Reichhaltigkeit und Angemessenheit:** Der Autor eines Metadaten-Schemas sollte sich bemühen, eine Menge von Metadatenelementen (oder Vokabeln) so zu definieren, dass diese einerseits umfassend, andererseits eine minimale Menge wesentlicher Elemente umfasst, die für die Dokumentation eines Domänenbereichs notwendig sind.

- **Einfachheit:** Der Autor eines Metadaten-Schemas sollte die technischen Kompetenz der Anwender im Blick behalten und seine Lösung sollte sowohl Anwendern mit wenig Werkzeug- und Ressourcenunterstützung wie auch Anwender, die diesbezüglich gut ausgestattet sind, gleichermaßen nützen.

- **Datenaustausch:** Eine wesentliche Funktion eines Metadaten-Schemas für wissenschaftliche Daten ist die Fähigkeit, Daten unter *Community*-Mitgliedern auszutauschen, oder gemeinsam zu nutzen.

- **Retrieval:** Eine weitere wesentliche Funktion eines Metadaten-Schemas für wissenschaftliche Daten ist die Fähigkeit, Daten unter Berücksichtigung

fachspezifischer Zugriffspfade zu finden und rezipierbar zu machen.

- **Datenarchivierung:** Eine wesentliche Funktion eines *Repository* ist die Fähigkeit zur Langzeitarchivierung von Daten. Schema-Entwickler sollten ihre Design-Entscheidungen mit der Gruppe der Nutzer digitaler Langzeitarchivierung koordinieren.

- **Datenpublikation:** Forschungsdaten sind ein wichtiger Bestandteil des Prozesses der wissenschaftlichen Kommunikation und -Dokumentation. Schema-Entwickler sollten die Verknüpfung von Daten mit den veröffentlichten Forschungsergebnissen, analog zu Zitaten einrichten, wie sie in wissenschaftlichen Publikationen üblich sind.

Nimmt man die obigen Anforderungen unter Berücksichtigung der Heterogenität der Eigenschaften von Forschungsdaten und deren Anordnung, stößt man auf die klassischen Fragestellungen, der sich insbesondere die Informationswissenschaft von Anfang an mit ihrer pragmatischen Betrachtungsweise auf den Informationsbegriff verschrieben hat. Dort sind in den Bereichen Information Retrieval und Datenarchivierung Methoden entstanden, bevor der Begriff Metadaten überhaupt existierte. Als Ausblick erscheint das synergetische Zusammenwirken des bewährten informationswissenschaftlichen Methodenspektrums unter Berücksichtigung einer sorgfältigen Absicherung auf Metadatenebene vielversprechend. Neuere Ansätze – wie etwa ontologiebasierte Informationssysteme – gehen diesen Weg und sind als elaborierte Variante von Taxonomien bzw. Thesauri unter Zugabe der bewährten objektorientierten Prinzipien zu verstehen.

Literatur

1 Backus, J. W.: The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference. Proceedings of the International Conference on Information Processing, pp.125-132, UNESCO, 1959

2 Balzert, Helmut: Lehrbuch der Software-Technik. Band 1. 2. Auflage. Elsevier-Verlag, 2001

3 Chen, Peter: The entity-relationship model - toward a unified view of data. ACM Transactions on Database Systems (TODS) - Special issue: papers from the international conference on very large data bases: September 22–24, 1975, March 1976, pp. 9-36, DOI: 10.1145/320434.320440

4 Connors, Christine: Metadata: Practical, painless, profitable. Bulletin of the American Society for Information Science and Technology. Vol. 32, No. 6, pp.13-14, Wiley Subscription Services, Inc., A Wiley Company, DOI: 10.1002/bult.2006.1720320606, 2006

5 CODATA-Germany: Grundsätze zum Umgang mit Forschungsdaten, CODATA-Germany, URL: http://www.codata-germany.org, 2010

6 DCMI: Dublin Core Metadata Element Set, Version 1.1, Dublin Core Metadata Initiative – Recommendation, 2012, Identifier: http://dublincore.org/documents/2012/06/14/dces/

7 Harold, Elliotte Rusty; Means, Scott: XML in a Nutshell – A Desktop Quick Reference; 3. Auflage, O'Reilly, 2004

8 Johnston, Pete; Powell, Andy: Expressing Dublin Core metadata using HTML/XHTML meta and link elements, Dublin Core Metadata Initiative – Recommendation, 2008, Identifier: http://dublincore.org/documents/2008/08/04/dc-html/

9 Kunze, J.; Baker, T.: The Dublin Core Metadata Element Set, Request for Comment (RFC) 5013, Network Working Group, 2007, URL: http://www.ietf.org/rfc/rfc5013.txt

10 Nilsson, Mikael; Powell, Andy, Johnson, Pete; Naeve, Ambjörn: Expressing Dublin Core metadata using the Resource Description Framework (RDF), Dublin Core Metadata Initiative – Recommendation, 2008, Identifier: http://dublincore.org/documents/2008/01/14/dc-rdf/

11 Nilsson, Mikael: Description Set Profiles: A constraint language for Dublin Core Application Profiles, Dublin Core Metadata Initiative – Working Draft, 2008, Identifier: http://dublincore.org/documents/2008/03/31/dc-dsp/

12 Nilsson, Mikael; Baker, Thomas; Johnson, Pete: Interoperability Levels for Dublin Core Metadata, Dublin Core Metadata Initiative – Recommended Resource, 2009, Identifier: http://dublincore.org/documents/2009/05/01/interoperability-levels/

13 Willis, Craig / Greenberg, Jane / White, Hollie: Analysis and synthesis of metadata goals for scientific data. Journal of the American Society for Information Science and Technology, Vol. 63 No. 8, pp.1505-20,, DOI: 10.1002/asi.22683, 2012

14 Bruvik, Tone Merete: Yesterday's Information Tomorrow – The Text Encoding Initiative; Österreichisches Literaturarchiv der Österreichischen Nationalbibliothek; 2002 PURL: http://purl.org/sichtungen/bruvik-tm-1a.htm

15 Schwed, Gernot: Konzeption und integrierte Formalisierung eines Referenzmodells für informationslogistische Agentensysteme – Ontologie; Intuition des Thesaurus?, Dissertation, Philosophische Fakultät der Universität des Saarlandes, GRIN-Verlag, 2007

16 Salton, Gerard, McGill, Michael J.: Information Retrieval – Grundlegendes für Informationswissenschaftler, McGraw-Hill, 1987