ZGDV-Bericht 75/93

Tabellen in SGML

Dipl.-Inform. Hans Holger Rath

Zentrum für Graphische Datenverarbeitung e.V. (ZGDV)
Wilhelminenstraße 7, D 64283 Darmstadt
Tel.: 06151/155245, FAX: 06151/155480
EMail: rath@igd.fhg.de

Autor: Hans Holger Rath
Gesamtherstellung: Eigendruck (Computersatz)
Printed in Germany

Danksagung

Ich möchte mich an dieser Stelle bei den Personen bedanken, die erheblich dazu beigetragen haben, daß dieser Bericht in der vorliegenden Form entstehen konnte.

Erwähnt seien zunächst Kim Adamson-Sharpe, Mitarbeiter bei SoftQuad (West), Inc., Kanada, und Steve Pepper, Mitarbeiter bei Falch Hurtigtrykk, Norwegen. Sie haben mit dem SoftQuad Author/Editor bzw. ArborText Publisher die Beispieltabelle erstellt und mir die SGML-Instanz zukommen lassen. Ohne diese Zuarbeit wäre insbesondere eine Aussage über die ArborText-Tabellen unmöglich gewesen. Vielen Dank Kim und Steve. Hilfreich beim strukturellen Aufbau des DTD-Vergleichs waren auch die Arbeiten von R. Eric Thomas, Mitarbeiter bei Rutherford Appleton Laboratory, England, die er mir unaufgefordert zukommen ließ. Bedanken möchte ich mich auch bei Yuri Rubinsky, Präsident von SoftQuad, Inc.; er lieferte die Tabellen-DTDs von SoftQuad und ICADD. Nicht vergessen möchte ich Donald A. Gignac, Mitarbeiter bei Carderoc Division – Naval Surface Warfare Center, Bethesda (Maryland), U.S.A., der mich über CALS und die verscheidenen Standards aus diesem Umfeld informiert hat.

Bedanken möchte ich mich auch bei meinen Kollegen Frank Häfemeier, Erik Meißner und Hans-Peter Wiedling. Sie haben meine Ausarbeitungen äußerst kritisch beobachtet und kommentiert.

Darmstadt, im Oktober 1993

Hans Holger Rath

Inhaltsverzeichnis

1 Einleitung

Tabellen dienen in technischen Dokumenten zur übersichtlichen Präsentation von Daten und werden vielfälltig verwendet. Tabellen sind entsprechend ein Layoutmittel zur Aufbereitung von zusammenhängenden Datensätzen. Generell lassen sich Layoutangaben beliebig skalieren und sind nur schwer mit einem festen Satz von Eigenschaften zu beschreiben; dazu sind die Anforderungen zu vielfältig[1]. Dies gilt entsprechend für die Tabellen.

Um Tabellen mit SGML [17][6][11][4][46] zu verarbeiten, muß ihre „Struktur“ in einer DTD (Dokumenttypdefinition) abgebildet werden. Das bedeutet (bei einer ersten Betrachtung), Layout mit SGML zu beschrieben. Dies ist zwar nicht das eigentliche Anliegen von SGML[2], es läßt sich aber bei Tabellen nicht vermeiden, da Struktur und Layout in einem sehr engen Zusammenhang stehen, der nur schwer trennbar ist.[3]

Damit bei den weiteren Ausführungen zum Thema Tabellen einheitliche Begriffe und eine gemeinsame Vorstellung von der Komplexität der zu betrachtenden Tabellen vorhanden sind, wird in Kapitel 2 eine (mögliche) Tabellen-Definition vorgestellt. Diese ist ein pragmatischer Ansatz, der sich aus der Erfahrung mit verschiedenen Dokumentarten und Publishing Systemen ergeben hat. Er verlangt nicht den Anspruch der Vollständigkeit.

Nach der Einführung der Begriffe werden in Kapitel 3 unterschiedliche Techniken zur Abbildung von Tabellen in SGML vorgestellt und in Kapitel 4 die Tabellen in drei Klassen eingeteilt, zu denen Anforderungen und Beispiele angeführt werden. In Kapitel 5 werden existierende DTDs für freie Tabellen ausführlich untersucht und bewertet. Im Kapitel 6 wird einen neues Konzept zur strukturorientierten Darstellung von freien Tabellen vorgestellt: Tabellen als Koordinatensysteme. Abschließend wird in Kapitel 7 eine Zusammenfassung der Ergebnisse geliefert. Anhang A enthält eine umfangreiche Auflistung von verschiedenen Tabellen-DTDs.

---

1) Es wird bei der (manuellen) Satztechnik nicht umsonst von der „Schwarzen Kunst“ gesprochen.

2) Zur Festlegung des Layouts sind andere Formatbeschreibungssprachen entwickelt worden bzw. sind in der Entwicklung. Zu nennen wären hier FOSI (Formatted Output Specification Instance, [43]) als Teil der U.S.-CALS-Initative und DSSSL (Document Style Semantics and Specification Language, [24]) als Standard der ISO. Eine Studie, die die Formate untersucht und vergleicht, ist in [14] zu finden.

3) Im Unterabschnitt *Tabellen als Koordinatensysteme* auf Seite 47 wird ein DTD-Modell entwickelt, bei dem auf viele Layoutangaben verzichtet werden kann, da hier die logischen Beziehungen der Daten zueinander im Vordergrund stehen und diese Beziehungen implizit das Layout bestimmen.

2 Begriffe und allgemeine Anforderungen

In den nachfolgenden Definitionen 2–1 bis 2–10 werden die im weiteren Text verwendeten Begriffe eingeführt. Gleichzeitig ergeben sich aus den Definitionen auch die Anforderungen, die an eine Beschreibung von Tabellen gestellt werden.

Ist in den Definitionen ein Begriff *kursiv* gesetzt, so gibt es für diesen Begriff ebenfalls eine Definition.

Definition 2–1 Tabellenzelle (oder kurz Zelle)[4)]

Tabellenzellen (s. Bsp. 2–1) sind rechteckige Gebiete, die die Daten der *Tabelle* beinhalten. Die Daten (sog. Zelleninhalt) können Text, Graphik aber auch wieder eine *Tabelle* sein. Jede Zelle läßt sich eineindeutig einer *Reihe* und einer *Spalte* zuordnen.

Die Zellen sind durch *Tabellenlinien* optisch voneinander getrennt. Außer den *Linien* befindet sich kein Zwischenraum zwischen den Zellen.

Die Zelleninhalte können vertikal oben, in der Mitte oder unten in einer Zelle plaziert sein, ihre horizontale Ausrichtung kann links, zentriert, rechts, Blocksatz oder nach einem vorgegebenen Zeichen (z.B. Komma bei Zahlen) sein und sie können in 90°-Schritten in der Zelle gedreht sein. Der Hintergrund der Zelle kann mit einer Farbe und/oder einem Muster gefüllt sein. Weitere Eigenschaften der Zelle sind der obere, untere, linke und rechte Abstand der Daten zur Zellenbegrenzung.

Die Breite der Zelle ergibt sich aus der Breite der *Spalte*, in der sich die Zelle befindet.

Die Höhe der Zelle ist der Maximalwert aus der Höhe, die der Zelleninhalt beansprucht, und der Höhe der anderen Zellen in der gleichen *Reihe*.

Beispiel 2–1 Zellen

Zelle 1: Vertikale Ausrichtung: zentriert Horizontale Ausrichtung: linksbündig	Zelle 2: Linker Abstand: 5mm Vertikale Ausrichtung: oben Horizontale Ausrichtung: linksbündig
Zelle 3: Vertikale Ausrichtung: oben Horizontale Ausrichtung: zentriert	Zelle 4 mit Hintergrundfarbe: Vertikale Ausrichtung: unten
Zelle 5: Vertikale Ausrichtung: oben Horizontale Ausrichtung: rechtsbündig	Zelle 6: Vertikale Ausrichtung: oben Horizontale Ausrichtung: nach Komma 171,30 DM 2,50 DM

Definition 2–2 Zusammengefügte Tabellenzelle

Eine **zusammengefügte Tabellenzelle** (s. Bsp. 2–2) ist eine *Zelle*, die zwei oder mehrere *Zellen* umfaßt, die direkt nebeneinander oder übereinander liegen. Auch eine zusammengefügte Zelle stellt ein Rechteck dar, das sich aber über mehrere *Reihen* und/oder *Spalten* erstrecken kann.

---

4) Es ist auch der Begriff Tabellenfeld bzw. Feld gebräuchlich.

Die Ausrichtung der Zelleninhalte und die weiteren Zelleneigenschaften entsprechen denen der „normalen“ *Zelle*. Sie beziehen sich lediglich auf den gesamten Rechteckbereich der zusammengefügten Zelle.

Die Breite einer zusammengefügten Zelle ergibt sich aus der Summe aller Spaltenbreiten der *Spalten*, über die sich die zusammengefügte Tabelle erstreckt.

Die Höhe einer zusammengefügten Zelle ist der Maximalwert aus der Höhe, die der Zelleninhalt beansprucht, und der Gesamthöhe der anderen *Zellen* aus den *Reihen*, über die sich die zusammengefügte Zelle erstreckt.

Beispiel 2–2 Zusammengefügte Zellen

Zelle 1 horizontal zusammengefügt mit Zelle 2	
Zelle 3	Zelle 4 vertikal zusammengefügt mit Zelle 6
Zelle 5	

Definition 2–3 Tabellenlinie (oder kurz Linie)[5)]

Tabellenlinien (s. Bsp. 2–3) grenzen die *Tabellenzellen* optisch voneinander ab. Sie bilden aber auch die visuelle Abgrenzung der *Tabelle* zum übrigen Dokument.

Die Linie kann verschiedene Strichstärken annehmen, sie kann einfach oder doppelt, durchgehend oder gestrichelt und sichtbar oder unsichtbar[6)] sein.

Beispiel 2–3 Linien

Zelle 1	Zelle 2
Zelle 3	Zelle 4 mit unsichtbarer unterer Linie
Zelle 5	Zelle 6

Definition 2–4 Tabellenreihe (oder kurz Reihe)

Eine **Tabellenreihe** (s. Bsp. 2–4) umfaßt alle direkt nebeneinander liegenden *Zellen*.

Die Breite der Reihe entspricht der Breite der *Tabelle* bzw. der Gesamtbreite aller *Spalten*.

Die Höhe der Reihe ist gleich der Höhe der *Zelle* mit der größten vertikalen Ausdehnung in der Reihe.

Beispiel 2–4 Reihen

Reihe 1	
Reihe 2	
Reihe 3	

5) Es ist auch der Begriff Tabellenumrandung bzw. Umrandung gebräuchlich.

6) Unsichtbare Tabellenlinien sollten nicht mit *zusammengefügten Zellen* verwechselt werden.

Definition 2–5 Tabellenkopf

Der **Tabellenkopf** (s. Bsp. 2–5) ist eine oder sind mehrere spezielle *Reihen*, sog. **Kopfreihen**. Sollte die *Tabelle* aufgrund ihrer Größe über einen oder mehrere Seitenumbrüche verlaufen, so werden die Kopfreihen auf jeder weiteren Seite, auf der mindestens eine *Reihe* der *Tabelle* plaziert ist, vor den anderen „normalen“ *Reihen* in der Reihenfolge, wie sie das erste Mal auftreten, wiederholt.

Eine Kopfreihe kann auf allen Seiten, über die sich die *Tabelle* erstreckt, oder nur auf den Folgeseiten wiederholt werden. Damit kann ein Fortsetzungstext, wie „Fortsetzung von Tabelle XYZ“, der als erste *Reihe* auf jeder Seite außer der ersten Seite erscheint, realisiert werden (s. Bsp. 2–7).

Beispiel 2–5 Kopfreihe

Kopfreihe 1	
Kopfreihe 2	
„Normale“ Reihe 1	
„Normale“ Reihe 2	

... ... Seitenumbruch ... ...

Wiederholte Kopfreihe 1	
Wiederholte Kopfreihe 2	
„Normale“ Reihe 3	

Definition 2–6 Tabellenrumpf

Der **Tabellenrumpf** umfaßt alle *Reihen* außer den *Kopf*- und *Fußreihen*.

Definition 2–7 Tabellenfuß

Der **Tabellenfuß** (s. Bsp. 2–6) ist eine oder sind mehrere spezielle *Reihen*, sog. Fußreihen. Sollte die *Tabelle* aufgrund ihrer Größe über einen oder mehrere Seitenumbrüche verlaufen, so werden die Fußreihen auf jeder vorhergehenden Seite, auf der mindestens eine *Reihe* der *Tabelle* plaziert ist, nach den anderen „normalen“ *Reihen* in der Reihenfolge, wie sie auf der letzten Seite auftreten, wiederholt.

Eine Fußreihe kann auf allen Seiten, über die sich die *Tabelle* erstreckt, oder auf allen Seiten außer der letzten wiederholt werden. Damit kann ein Fortsetzungstext, wie „Tabelle XYZ wird fortgesetzt“, der als letzte *Reihe* auf jeder Seite außer der letzten Seite erscheint, realisiert werden (s. Bsp. 2–7).

Beispiel 2–6 Fußreihe

Kopfreihe 1	
Kopfreihe 2	
„Normale“ Reihe 1	
„Normale“ Reihe 2	
wiederholte Fußreihe 1	

... ... Seitenumbruch ... ...

wiederholte Kopfreihe 1	
wiederholte Kopfreihe 2	
„Normale“ Reihe 3	
Fußreihe 1	

Beispiel 2–7 Kopf-/Fußreihen mit Fortsetzungstext

Kopfreihe 1	
Kopfreihe 2	
„Normale“ Reihe 1	
„Normale“ Reihe 2	
wiederholte Fußreihe 1	

Tabelle XYZ wird fortgesetzt ... (Fußreihe)

... ... Seitenumbruch ... ...

Fortsetzung der Tabelle XYZ (Kopfreihe)

wiederholte Kopfreihe 1	
wiederholte Kopfreihe 2	
„Normale“ Reihe 3	
Fußreihe 1	

Definition 2–8 Tabellenspalte (oder kurz Spalte)

Eine **Tabellenspalte** (s. Bsp. 2–8) umfaßt alle direkt übereinander liegenden *Zellen*. Die Spalte verläuft über die gesamte *Tabelle*, also auch über Seitenumbrüche.

Die Breite der Spalte ist eine Eigenschaft von dieser. Man kann die Breite in absoluten oder relativen Werten (Prozent oder Proportionen bzgl. der Tabellenbreite) festlegen. Eine Mischung von absoluten und relativen Werten ist möglich[7)], wobei sich die Proportionen auf die Restbreite der *Tabelle* beziehen, die nach Abzug der absoluten und prozentualen Breiten von der vorge-

7) Als Beispiel eine *Tabelle* mit drei Spalten: in absoluten Werten 40mm, 60mm, 100mm; in Prozent 20%, 30%, 50%; in Proportionen 1, 1.5, 2.5 oder 2, 3, 5; gemischt 40mm, 30%, 5.

gebenen Tabellenbreite übrig bleibt. Bei der Verwendung von mindestens einem relativen Wert muß die *Tabelle* einen absoluten Wert für die Gesamtbreite besitzen. Bei der ausschließlichen Verwendung von absoluten Spaltenbreiten ergibt sich der Wert für die Gesamtbreite der *Tabelle* aus der Summe aller Spaltenbreiten.

Die Höhe einer Spalte ist gleich der vertikalen Ausdehnung der *Tabelle* über alle beanspruchten Seiten.

Beispiel 2–8 Spalten

Spalte 1	Spalte 2

Definition 2–9 Tabelle

Eine **Tabelle** besteht aus *n Spalten* und *m Reihen* ($n, m > 0$).

Die Breite ergibt sich entweder aus der Summe aller Spaltenbreiten (bei ausschließlich absoluten Werten, s.o.) oder wird für die Tabelle angegeben. Wird sie bei der Tabelle angegeben, so kann entweder ein absoluter Wert (z.B. 140mm) oder relativer Wert bzgl. der Seiten- oder Textspaltenbreite angegeben werden.[8)]

Die Höhe der Tabelle ergibt sich aus der Summe aller Reihenhöhen.

Die horizontale Ausrichtung einer Tabelle kann linksbündig, rechtsbündig oder zentriert sein.

Der Seitenumbruch einer Tabelle geschieht zwischen zwei *Reihe*n. Diese *Reihen* dürfen keine gemeinsame zusammengefügte Zelle beinhalten[9)].

Definition 2–10 Tabellenfußnote[10)]

Eine **Tabellenfußnote** (s. Bsp. 2–9) ist eine spezielle Fußnote. Der Fußnotenanker (z.B. eine hochgestellte Zahl) ist bei einem Zelleninhalt plaziert. Der Fußnotentext ist nicht, wie bei *normalen* Fußnoten üblich, unten auf der Seite, sondern in der allerletzten Reihe der Tabelle oder direkt unterhalb der Tabelle plaziert, in der sich auch der Fußnotenanker befindet. Diese Reihe besteht nur aus einer Zelle, die über alle Spalten horizontal zusammengefügt ist.

---

8) Die Eigenschaft, daß eine Tabelle im Querformat auf der Seite erscheint (sog. gedrehte Tabelle), ist eine Eigenschaft der Seitenformatierung nicht der Tabelle an sich.

9) Sollte eine zusammengefügte *Zelle* höher als eine Seite sein, so muß natürlich innerhalb dieser *Zelle* zwischen zwei *Reihen* umgebrochen werden.

10) Eine Tabellenfußnote ist eigentlich kein direkter Bestandteil des Layouts der Tabelle; sie läßt sich aber auch nicht der Kategorie Zelleninhalte zuordnen. Deshalb ist sie hier unter den Begriffen aufgeführt.

Beispiel 2–9 Tabellenfußnoten

Zelleninhalt mit einer Tabellenfußnote[1)]	Zelle 2	Zelle 3
Zelle 4	Zelle 5	Zelleninhalt mit einer Tabellenfußnote[2)]
Zelle 6	Zelle 7	Zelle 8

[1)] Diese Fußnote gilt für alle Zellen der ersten Spalte

[2)] Diese Fußnote gilt für alle anderen Zellen.

3 Techniken zur Beschreibung von Tabellen in SGML

Bei der möglichen Vielfältigkeit von Tabellen liegt es auf der Hand, daß es keine einheitliche Beschreibung **aller** Tabellen durch **eine** Tabellen-DTD geben kann; dazu sind die Ansprüche, die an eine solche DTD gestellt werden, zu unterschiedlich. Die nachfolgende Auflistung[11] von Techniken zur (SGML-) Beschreibung von Tabellen soll eine Klassifikation der Tabellen-DTDs (s. Kapitel 4) ermöglichen.

3.1 Bedeutung der Zelle

Generell kann man zwischen zwei Zellenarten unterscheiden: die *allgemeine Zelle* und die *Datenbank-Zelle*[12].

3.1.1 Allgemeine Zelle

Die **allgemeine Zelle** (s. Bsp. 3–1) wird mit einem SGML-Tag markiert, das keinerlei Schlüsse auf den Inhalt der Zelle zuläßt. Der Vorteil hierbei ist, daß die Zelle beliebige Daten beinhalten kann.

Beispiel 3–1 Allgemeine Zellen

```
<!-- Teil-DTD: -->
<!ELEMENT c  - - (#PCDATA) >

<!-- Instanz: -->
<c>XYZ 1234</>
<c>15,50 DM</c>
```

3.1.2 Datenbank-Zelle

Die **Datenbank-Zelle** besitzt entweder ein *dediziertes Tag* (s. Bsp. 3–2), das eindeutig einem bestimmten Datum z. B. aus einer Datenbank zugeordnet ist, oder ein *allgemeines, attributiertes Zellen-Tag* (s. Bsp. 3–3), das mit einem Attribut versehen ist, welches Auskunft über den Inhalt der Zelle gibt. Beim dedizierten Tag verliert man jegliche Allgemeingültigkeit, man kann aber in der DTD sicherstellen, daß ein bestimmtes Datum in der Tabelle angegeben werden muß. Bei der Lösung mit einem Attribut bleibt das Tag ein allgemeines Zellen-Tag und trotzdem kann der Typ des Zelleninhaltes identifiziert werden. Der Vorteil beider Arten der Datenbank-Zelle ist, daß eine Volltext-Recherche in dem SGML-Dokument nach bestimmten Tabellendaten sehr einfach ist.

Beispiel 3–2 Dedizierte Tags

```
<!-- Teil-DTD: -->
<!ELEMENT (partno | cost)  - - (#PCDATA) >

<!-- Instanz: -->
<partno>XYZ 1234</>
<cost>15,50</> <!-- "DM" ist fester Text des End-Tags -->
```

---

[11] Diese Aufteilung ist an die Idee von David Sklar (Electronic Book Technologies, Inc.) angelehnt, die von Ludo Van Vooren (Avalanche Development Company) auf der GCA (Graphics Communication Association) Konferenz SGML Europe 93 in Amsterdam [36][47] vorgestellt wurde.

[12] D. Sklar benutzt hier die engl. Begriffe „unknown cell“ und „database record“.

Beispiel 3–3 Allgemeines, attributiertes Tag

```
<!-- Teil-DTD: -->
<!ELEMENT c  - - (#PCDATA) >
<!ATTLIST c  type   CDATA   #REQUIRED >

<!-- Instanz: -->
<c type="partno">XYZ 1234</>
<c type="cost">15,50</> <!-- "DM" wird beim End-Tag generiert -->
```

3.2 Komplexität des Zelleninhaltes

Die Komplexität des Zelleninhaltes kann von einfachem Text, über einfache Absätze, über beliebige Unterelemente (Absätze, Aufzählungen, Graphiken, etc.) bis hin zu Tabellen reichen.

3.3 Logische Abgrenzung der einzelnen Reihen[13)]

Man kann zwischen *expliziter* und *impliziter* Beschreibung der Reihen unterschieden.

3.3.1 Explizite Beschreibung

Die **explizite** Beschreibung (s. Bsp. 3–4) kennzeichnet eine Reihe mit einem entsprechenden Tag als solche. Diese Art der Beschreibung wird häufig mit den „allgemeinen Zellen" kombiniert. Die Vorteile sind, daß die Formatierung einer solchen Tabelle sehr einfach ist und, wie oben bereits erwähnt, beliebige Daten in der Tabelle stehen können.

Beispiel 3–4 Explizite Reihen

```
<!-- Teil-DTD: -->
<!ELEMENT row  - - (c+) >
<!ELEMENT c    - - (#PCDATA) >

<!-- Instanz: -->
<row>
<c>UNIX</><c>Betriebssystem</>
</row>
<row>
<c>TeX</><c>Public Domain Formatierer</>
</row>
```

3.3.2 Implizite Beschreibung

Bei der **impliziten** Beschreibung der Reihen (s. Bsp. 3–5) kennzeichnet ein (Daten-) Tag den Beginn einer neuen Reihe. Dieses Tag repräsentiert das erste Datum der Reihe und stellt damit gleichzeitig den Reihenanfang dar. Die implizite Beschreibung muß mit den *Datenbank-Zellen* kombiniert werden, da allgemeine Zellen einen Reihenanfang nicht kennzeichnen können[14)]. Der Vorteil der impliziten Beschreibungsart ist, daß eine Tabelle in SGML nicht mehr als solche zu erkennen ist, sondern nur noch einen Satz von Daten beschreibt. Dieser Vorteil ist aber auch gleichzeitig ein Nachteil, da die Formatierung eines solchen Datensatzes als eine Tabelle von manchen SGML-Systemen überhaupt nicht oder nur mit hohem Integrationsaufwand durchgeführt werden kann.

---

13) Theoretisch können Tabellen auch spaltenweise beschrieben werden; da die meisten Formatierer aber reihenweise arbeiten, hat sich die reihenweise Beschreibung auch in SGML-DTDs etabliert.

14) Würde eine allgemeine Zelle für einen Reihenanfang stehen, wäre sie keine allgemeine Zelle mehr.

Beispiel 3–5 Implizite Reihen

```
<!-- Teil-DTD: -->
<!ELEMENT (term | definition)  - - (#PCDATA) >

<!-- Instanz: -->
<!-- impliziter Reihenanfang -->
<term>UNIX</><definition>Betriebssystem</>
<!-- implizites Reihenende -->
<!-- impliziter Reihenanfang -->
<term>TeX</><definition>Public Domain Formatierer</>
<!-- implizites Reihenende -->
```

3.4 Identifizierung der Kopf-/Fußreihen

Es sind vier Arten denkbar, eine Reihe als Kopf-/Fußreihe zu markieren. Die Verwendung von *Umhüllungselementen*, von *Spezial-Reihenelementen*, von *Reihen-Attributen* und von *Deklarationselementen* in Kombination mit Reihen-Attributen. Im Gegensatz zu einigen anderen Punkten kann man hier nicht sagen, welche Darstellung besser oder schlechter ist. Die einzelnen Darstellungen sollten allerdings einheitlich auf die gesamte Tabellenbeschreibung angewendet und nicht gemischt werden.

3.4.1 Umhüllungselemente

Die **Umhüllungselemente** (s. Bsp. 3–6) kennzeichnen den Beginn (und das Ende) eines Tabellenkopfes bzw. Tabellenfußes. Sie haben die „normalen“ Tabellenreihen zum Inhalt. Diese Reihen sind dann die Kopf-/Fußreihen[15)].

Beispiel 3–6 Umhüllungselemente

```
<!-- Teil-DTD: -->
<!ELEMENT table  - - (tabhead?, tabbody, tabfoot?) >
<!ELEMENT (tabhead | tabbody | tabfoot)
                 - - (row+) >
<!ELEMENT row    - - (c+) >
<!ELEMENT c      - - (#PCDATA) >
```

15) Um den Tabellenaufbau konsistent zu beschreiben, sollte bei der Verwendung von Umhüllungselementen auch der Tabellenrumpf von einem entsprechenden Element umhüllt werden und bei der Verwendung von Spezial-Reihenelementen, Reihen-Attributen oder Deklarationselementen sollte der Rumpf nicht umhüllt werden (s. u.).

```
<!-- Instanz: -->
<table>
<tabhead>
<row>
<c>Begriff</><c>Definition</>
</row>
<row>
<c>in engl.</><c>in deutsch</>
</row>
</tabhead>
<tabbody>
<row>
<c>UNIX</><c>Betriebssystem</>
</row>
<row>
<c>TeX</><c>Public Domain Formatierer</>
</row>
</tabbody>
<tabfoot>
<row>
<c>...</><c>...</>
</row>
</tabfoot>
</table>
```

3.4.2 Spezial-Reihenelemente

Die **Spezial-Reihenelemente** (s. Bsp. 3–7) kennzeichnen die einzelnen Kopf- bzw. Fußreihen. Ihr Inhaltsmodell sollte das gleiche wie das der „normalen“ Reihen sein.

Beispiel 3–7 Spezial-Reihenelemente

```
<!-- Teil-DTD: -->
<!ELEMENT table  - - (headrow*, bodyrow+, footrow*) >
<!ELEMENT (headrow | bodyrow | footrow)
                 - - (c+) >
<!ELEMENT c      - - (#PCDATA) >
```

```
<!-- Instanz: -->
<table>
<headrow>
<c>Begriff</><c>Definition</>
</headrow>
<headrow>
<c>in engl.</><c>in deutsch</>
</headrow>
<bodyrow>
<c>UNIX</><c>Betriebssystem</>
</bodyrow>
<bodyrow>
<c>TeX</><c>Public Domain Formatierer</>
</bodyrow>
<footrow>
<c>...</><c>...</>
</footrow>
</table>
```

3.4.3 Reihen-Attribute

Bei der Verwendung von Elementen mit **Reihen-Attributen** (s. Bsp. 3–8) haben alle Tabellenreihen ein oder mehrere Attribute, die die Zugehörigkeit der Reihe zum Tabellenkopf, -rumpf oder -fuß kennzeichnen. Diese Attribute werden im SGML-Dokument entsprechend gesetzt, wobei die Zugehörigkeit zum Rumpf der Defaultwert des Attributes sein sollte.

Beispiel 3–8 Reihen-Attribute

```
<!-- Teil-DTD: -->
<!ELEMENT table  - - (row+) >
<!ELEMENT row    - - (c+) >
<!ATTLIST row    rowtype   (head | body | foot)   "body" >
<!ELEMENT c      - - (#PCDATA) >

<!-- Instanz: -->
<table>
<row rowtype="head">
<c>Begriff</><c>Definition</>
</row>
<row rowtype="head">
<c>in engl.</><c>in deutsch</>
</row>
<row>
<c>UNIX</><c>Betriebssystem</>
</row>
<row>
<c>TeX</><c>Public Domain Formatierer</>
</row>
<row rowtype="foot">
<c>...</><c>...</>
</row>
</table>
```

3.4.4 Deklarationselemente

Die **Deklarationselemente** (s. Bsp. 3–9) sind ein Konzept, das auch bei der Angabe der geometrischen Spezifikationen und den zusammengefügten Zellen angewendet wird (s.u.). Dabei wird vor der eigentlichen Tabelle ein EMPTY-Tag eingefügt, dessen Attribute Listen von allen Kopf- bzw. Fußreihen enthalten. Die Reihen selbst erhalten über ein ID-Attribut eindeutige Bezeichner, die im Deklarationselement in einem IDREFS-Attribut referiert werden[16]. Alle im Deklarationselement unter dem Attribut für Kopfreihen aufgeführten Reihen gehören dem Tabellenkopf an und die dem Attribut für Fußreihen zugeordneten gehören dem Tabellenfuß an[17].

Beispiel 3–9 Deklarationselemente

```
<!-- Teil-DTD: -->
<!ELEMENT table    - - (rowdecl, row+) >
<!ELEMENT rowdecl  - O EMPTY >
<!ATTLIST rowdecl  heads   IDREFS   #IMPLIED
                   feet    IDREFS   #IMPLIED >
<!ELEMENT row      - - (c+) >
<!ATTLIST row      id   ID   #IMPLIED >
<!ELEMENT c        - - (#PCDATA) >
```

[16] Anstatt des ID-IDREFS-Mechanismus können auch die Attributtypen NUMBER und NUMBERS verwendet werden. Hierbei wird zwar vom SGML-Parser die Verknüpfung zwischen dem Deklarationselement und den Reihen nicht überprüft, dafür können aber bei mehreren Tabellen die gleichen Bezeichner (= Nummern) verwendet werden. Man könnte die Angabe der Nummern bei den Reihen sogar weglassen, da die Reihen ohnehin implizit durchnumeriert sind.

[17] Würde man beim Deklarationselement die eindeutige ID angeben und bei den Reihen nur darauf referieren, so entspräche dies in etwa der Angabe von Reihenattributen (s.o.).

```
<!-- Instanz: -->
<table>
<rowdecl heads="1 2" feet="3">
<row id="1">
<c>Begriff</><c>Definition</>
</row>
<row id="2">
<c>in engl.</><c>in deutsch</>
</row>
<row>
<c>UNIX</><c>Betriebssystem</>
</row>
<row>
<c>TeX</><c>Public Domain Formatierer</>
</row>
<row id="3">
<c>...</><c>...</>
</row>
</table>
```

3.5 Angabe von geometrischen Spezifikationen

Die geometrischen Spezifikationen umfassen die Beschreibung der Spaltenbreiten und die Ausrichtung der Daten in den Zellen (siehe auch Abschnitt 2 auf Seite 3). Sie legen zusammen mit den Angaben zum vertikalen und horizontalen Zusammenfügen von Zellen (s.u.) das Tabellenlayout exakt fest. Die geometrischen Angaben erfolgen in jedem Fall über Attribute und evtl. zusätzliche Deklarationselemente. Es sind allerdings verschiedene Varianten möglich: *abstrakte Spezifikation* durch Angabe des Tabellentyps (s. Bsp. 3–10) oder einer symbolische Spaltenbreite (s. Bsp. 3–11) und *konkrete Spezifikation* durch direkte Wertangabe.

3.5.1 Abstrakte Spezifikation

Bei der **abstrakten Spezifikation** werden keine Zahlenwerte für die Ausdehnung der Tabelle bzw. Spalten angegeben, sondern man verwendet abstrakte Namen. Dies kann der Name für einen bestimmten Tabellentyp oder ein symbolischer Name für die Breite einer Spalte sein. Für die beiden abstrakten Angaben muß der Formatierer wissen, wie breit die entsprechende Tabelle bzw. Spalte ist. Die Attributwerte können, wie bei der konkreten Spezifikation (s.u.), direkt bei den Zellen bzw. Spalten oder bei Deklarationselementen angegeben werden.

Beispiel 3–10 Tabellentyp

```
<!-- Teil-DTD: -->
<!ELEMENT table  - - (row+) >
<!ATTLIST table  type   (parts_01 | parts_02 | parts_03)   #REQUIRED >
<!ELEMENT row    - - (c+) >
<!ELEMENT c      - - (#PCDATA) >
```

```
<!-- Instanz: -->
<table type="parts_03">
<row>
<c>XYZ 1234</>
<c>15,50 DM</c>
</row>
</table>
```

Beispiel 3–11 Symbolische Spaltenbreiten

Die Attribute zur Angabe der Spaltenbreiten könnten Werte annehmen wie **"page"**, **"column"** oder **"narrow"**, **"medium"**, **"wide"**.

Die Ausrichtung der Daten in den Zellen kann bei der abstrakten Spezifikation ebenfalls durch z.B. den Tabellentyp festgelegt sein. Im allgemeinen werden die Ausrichtungen aber mit konkreten Spezifikationen angegeben (s.u.).

3.5.2 Konkrete Spezifikation

Konkrete Spezifikationen zeichnen sich dadurch aus, daß im Attributwert direkt die Breite einer Zelle in Form einer Maßangabe[18)] oder die Ausrichtung der Daten durch Angabe der „Formatierungsanweisung“ steht. Für beide Angaben sind sowohl die *direkte Form* (z.B. bei einem Zellenattribut (s. Bsp. 3–12) oder einem Deklarationselement (s. Bsp. 3–13)), die *indirekte Form* (mit ID-IDREF-Beziehungen und Deklarationselement (s. Bsp. 3–14, 3–15)) als auch das *Vererbungskonzept* (s. Bsp. 3–16) möglich.

> *Anmerkung*: Bei der Vererbung gibt es auf allen strukturellen Ebenen der Tabelle (Zelle, Reihe, Tabellenkopf, Tabellenrumpf, Tabellenfuß, Tabelle) die gleichen Attribute. Auf der höchsten Ebene, der Tabellenebene, müssen die Attributwerte angegeben werden; auf den weiteren Ebenen dagegen nicht. Wie breit die einzelne Zelle ist und wie ihre Daten ausgerichtet sind, ergibt sich aus dem Attributwert der Zelle (falls angegeben), der Reihe (falls angegeben), des Kopf/Rumpfes/Fußes (falls angegeben) oder der Tabelle. Die Information wird von der Tabelle an den Kopf/Rumpf/Fuß, vom Kopf/Rumpf/Fuß an die Reihen und von den Reihen an die Zellen weitergegeben, also vererbt.

Es ist durchaus sinnvoll, die unterschiedlichen Darstellungsmöglichkeiten für die verschiedenen Attribute zu kombinieren. So ist es sinnvoll, die Spaltenbreiten mit Deklarationselementen und die Ausrichtung der Zellendaten mit Vererbung oder direkten Wertangaben zu beschreiben.

Beispiel 3–12 Direkte Angabe der Spaltenbreite und Ausrichtung

```
<!-- Teil-DTD: -->
<!ELEMENT table  - - (row+) >
<!ELEMENT row    - - (c+) >
<!ELEMENT c      - - (#PCDATA) >
<!ATTLIST c      width    NUTOKEN                    #IMPLIED
                 halign   (left | center | right)    "left"
                 valign   (top | center | bottom)    "top" >

<!-- Instanz: -->
<table>
<row>
<c width="12mm">UNIX</>
<c width="40mm" halign="center" valign="center">Betriebssystem</>
<row>
<!-- Zellenbreiten ergeben sich aus Reihenfolge (= Spalten, s.o.) -->
<c halign="right" valign="bottom">TeX</>
<c valign="center">Public Domain Formatierer</>
</row>
</table>
```

[18)] Die Maßangabe kann, wie bereits erwähnt, ein absoluter Wert, eine Prozentangabe oder eine Proportionalzahl sein.

Beispiel 3–13 Direkte Angabe der Spaltenbreite mit Deklarationselement[19)]

```
<!-- Teil-DTD: -->
<!ELEMENT table  - - (row+) >
<!ATTLIST table  gridx   CDATA   #REQUIRED >
<!ELEMENT row    - - (c+) >
<!ELEMENT c      - - (#PCDATA) >

<!-- Instanz: -->
<table gridx="12mm 40mm">
<row>
<c>UNIX</>
<c>Betriebssystem</>
<row>
<c>TeX</>
<c>Public Domain Formatierer</>
</row>
</table>
```

Beispiel 3–14 Indirekte Angabe der Spaltenbreite mit Deklarationselementen

```
<!-- Teil-DTD: -->
<!ELEMENT table    - - (coldecl+, row+) >
<!ELEMENT coldecl  - O EMPTY >
<!ATTLIST coldecl  colname     ID        #REQUIRED
                   colwidth    NUTOKEN   #REQUIRED >
<!ELEMENT row      - - (c+) >
<!ELEMENT c        - - (#PCDATA) >
<!ATTLIST c        colname     IDREF   #IMPLIED >

<!-- Instanz: -->
<table>
<coldecl colname="term" colwidth="12mm">
<coldecl colname="definition" colwidth="40mm">
<row>
<!-- Zellen mit expliziten Verweisen auf Deklarationen -->
<c colname="term">UNIX</>
<c colname="definition">Betriebssystem</>
</row>
<row>
<!-- Zellenbreiten ergeben sich aus Reihenfolge (= Spalten, s.o.) -->
<c>TeX</>
<c>Public Domain Formatierer</>
</row>
</table>
```

[19)] Das Deklarationselement kann auch das Tabellen-Element sein. Die Angaben müssen nicht ausschließlich absolute Werte sein (s. o.). Der Attributtyp CDATA wurde gewählt, da die Einhaltung der Reihenfolge der Token in einem NUTOKENS-Attribut durch die Parser nicht gewährleistet ist.

Beispiel 3–15 Indirekte Angabe der Ausrichtung mit Deklarationselementen[20)]

```
<!-- Teil-DTD: -->
<!ELEMENT table     - - (celldecl, row+) >
<!ELEMENT celldecl  - O EMPTY >
<!ATTLIST celldecl  left      IDREFS   #IMPLIED
                    hcenter   IDREFS   #IMPLIED
                    right     IDREFS   #IMPLIED
                    top       IDREFS   #IMPLIED
                    vcenter   IDREFS   #IMPLIED
                    bottom    IDREFS   #IMPLIED >
<!ELEMENT row       - - (c+) >
<!ELEMENT c         - - (#PCDATA) >
<!ATTLIST c         id   ID   #IMPLIED >

<!-- Instanz: -->
<table>
<celldecl left="C1 C4" hcenter="C2"    right="C3"
          top="C1"     vcenter="C2 C4" bottom="C3">
<row>
<c id="C1">UNIX</>
<c id="C2">Betriebssystem</>
</row>
<row>
<c id="C3">TeX</>
<c id="C4">Public Domain Formatierer</>
</row>
</table>
```

Beispiel 3–16 Vererbte Ausrichtungen

```
<!-- Teil-DTD: -->
<!ELEMENT table  - - (row+) >
<!ATTLIST table  halign   (left | hcenter | right)   "left"
                 valign   (top | vcenter | bottom)   "top" >
<!ELEMENT row    - - (c+) >
<!ATTLIST row    halign   (left | hcenter | right)   #IMPLIED
                 valign   (top | vcenter | bottom)   #IMPLIED >
<!ELEMENT c      - - (#PCDATA) >
<!ATTLIST c      halign   (left | hcenter | right)   #IMPLIED
                 valign   (top | vcenter | bottom)   #IMPLIED >

<!-- Instanz: -->
<table valign="vcenter">
<row valign="bottom">
<c halign="hcenter">UNIX</>
<c>Betriebssystem</>
</row>
<row>
<c>TeX</>
<c halign="right" valign="top">Public Domain Formatierer</>
</row>
</table>
```

[20)] Auch hier können, wie bei den Kopf-/Fußreihen, die eindeutigen Bezeichner entweder bei den Zellen oder beim Deklarationselement angegeben werden. Beide Methoden sind denkbar. Da im nachfolgenden Beispiel die Bezeichner der Zellen einer Durchnumerierung entsprechen, könnte man, wie bei den Reihen, auf die ID-Angabe verzichten.

3.6 Vertikales und horizontales Zusammenfügen von Zellen

Die Informationen über das Zusammenfügen von einzelnen Zellen werden bei den Attributen der Zellen und/oder Deklarationselementen angegeben. Hier gibt es im wesentlichen drei Konzepte: *ID-IDREF-Beziehungen*, *numerierte Zellen* und *Zählerangaben*.

3.6.1 ID-IDREF-Beziehungen

Bei den **ID-IDREF-Beziehungen** (s. Bsp. 3–17) werden alle Spalten in Deklarationselementen mit ID-Bezeichnern versehen. In weiteren Deklarationselementen werden alle horizontalen zusammengefügten Zellen einzeln mit einem ID-Bezeichner versehen und mit zwei weiteren IDREF-Attributen die erste und die letzte Spalte referiert. Die zusammengefügten Zellen wiederum referieren mit einem IDREF-Attribut den Bezeichner der entsprechenden Zusammenfügung[21]. Die Zuordnung der Zellen zu den Spalten ergibt sich aus der Position der Zelle in der jeweiligen Reihe.

Beispiel 3–17 Horizontales Zusammenfügen mit ID-IDREF-Beziehungen

```
<!-- Teil-DTD: -->
<!ELEMENT table     - - (coldecl*, spanddecl*, row+) >
<!ELEMENT coldecl   - O EMPTY >
<!ATTLIST coldecl   colname    ID      #REQUIRED >
<!ELEMENT spandecl  - O EMPTY >
<!ATTLIST spandecl  spanname   ID      #REQUIRED
                    start      IDREF   #REQUIRED
                    end        IDREF   #REQUIRED >
<!ELEMENT row       - - (c+) >
<!ELEMENT c         - - (#PCDATA) >
<!ATTLIST c         spanname   IDREF   #IMPLIED >

<!-- Instanz: -->
<table>
<coldecl colname="A">
<coldecl colname="B">
<coldecl colname="C">
<spandecl spanname="BtoC" start="B" end="C">
<row>
<c>1. Zelle in 1. Reihe</>
<c spanname="BtoC">2. Zelle verl&auml;uft &uuml;ber 2 Spalten in 1. Reihe</>
<!-- Letzte Zelle wird durch Zusammenfuegen angelegt -->
</row>
<row>
<c>1. Zelle in 2. Reihe</>
<c>2. Zelle in 2. Reihe</>
<c>3. Zelle in 2. Reihe</>
</row>
</table>
```

3.6.2 Numerierte Zellen

Bei dem Konzept der **numerierten Zellen** (s. Bsp. 3–18) erhalten alle Zellen von links nach

[21] Was hier für horizontales Zusammenfügen über Spalten beschrieben ist, gilt analog für vertikales Zusammenfügen über Reihen.

rechts und von oben nach unten gezählt implizit eine eindeutige Nummer[22]. In einem Deklarationselement (das kann auch das Tabellen-Element sein) wird bei einem Attribut das „Arrangement“, das die Zellen bilden, mit den Zellennummern beschrieben. Den Umbruch von einer Reihe in die nächste kennzeichnet dabei ein bestimmtes Zeichen, z.B. der Schrägstrich „/“. Sind Zellen horizontal oder vertikal zusammengefügt, so tauchen ihre Nummern mehrfach in dem „Arrangement“-Attribut auf.

Beispiel 3–18 Horizontales und vertikales Zusammenfügen mit Arrangement

```
<!-- Teil-DTD: -->
<!ELEMENT table  - - (row+) >
<!ATTLIST table  arrange   CDATA   #REQUIRED >
<!ELEMENT row    - - (c+) >
<!ELEMENT c      - - (#PCDATA) >

<!-- Instanz: -->
<table arrange="1 2 2 / 1 3 4">
<row>
<c>1. Zelle verl&auml;uft &uuml;ber 2 Reihen in 1. Spalte</>
<c>2. Zelle verl&auml;uft &uuml;ber 2 Spalten in 1. Reihe</>
<!-- Letzte Zelle wird durch Zusammenfuegen angelegt -->
</row>
<row>
<!-- Erste Zelle ist bereits durch Zusammenfuegen vorhanden -->
<c>3. Zelle steht in 2. Reihe in 2. Spalte</>
<c>4. Zelle steht in 2. Reihe in 3. Spalte</>
</row>
</table>
```

3.6.3 Zählerangaben

Die Verwendung von **Zählerangaben** (s. Bsp. 3–19) ist ein relativ einfacher Mechanismus. Wird die Tabelle reihenweise beschrieben (wovon bei allen Beispielen ausgegangen wird), so kann jede Zelle bei zwei Attributen angeben, über wieviele Spalten bzw. Reihen sie sich erstreckt.

Beispiel 3–19 Horizontales und vertikales Zusammenfügen mit Zählerangaben

```
<!-- Teil-DTD: -->
<!ELEMENT table  - - (row+) >
<!ELEMENT row    - - (c+) >
<!ELEMENT c      - - (#PCDATA) >
<!ATTLIST c      spanrowcount   NUMBER   #IMPLIED
                 spancolcount   NUMBER   #IMPLIED >

<!-- Instanz: -->
<table>
<row>
<c spanrowcount="2">1. Zelle verl&auml;uft &uuml;ber 2 Reihen in 1. Spalte</>
<c spancolcount="2">2. Zelle verl&auml;uft &uuml;ber 2 Spalten in 1. Reihe</>
<!-- Letzte Zelle wird durch Zusammenfuegen angelegt -->
</row>
<row>
<!-- Erste Zelle ist bereits durch Zusammenfuegen vorhanden -->
<c>3. Zelle steht in 2. Reihe in 2. Spalte</>
<c>4. Zelle steht in 2. Reihe in 3. Spalte</>
</row>
</table>
```

[22] Implizit deshalb, weil die Nummer in keinem Zellenattribut oder ähnlichem auftaucht. Sie wird lediglich intern vom Formatiersystem generiert.

3.7 Tabellenlinien

Tabellenlinien lassen sich auf zwei Arten angeben: durch *Elemente* oder *Attribute*.

3.7.1 Elemente

Verwendet man **Elemente** (s. Bsp. 3–20) zur Angabe der Linien, so gibt es ein Element für die horizontale Linien zwischen zwei Reihen (bzw. Zellen) und ein Element für die vertikalen Linien zwischen zwei Spalten (bzw. Zellen). Sollen die Linien gestrichelt sein oder verschiedene Stärken etc. besitzen, so müssen die Elemente zusätzliche Attribute haben.

Beispiel 3–20 Eine Tabelle mit linken und rechten Umrandungslinien (Linienstärke 4 Punkt) und je einer Linie zwischen den beiden Reihen und den beiden Spalten (Linienstärke 2 Punkt)

```
<!-- Teil-DTD: -->
<!ELEMENT table  - - ((horirule*, row)+, horirule*) >
<!ELEMENT horirule  - O EMPTY >
<!ATTLIST horirule  width    NUTOKEN   #REQUIRED >
<!ELEMENT row    - - ((vertrule?, c)+, vertrule?) >
<!ELEMENT c      - - (#PCDATA) >
<!ELEMENT vertrule  - O EMPTY >
<!ATTLIST vertrule  width    NUTOKEN   #REQUIRED >

<!-- Instanz: -->
<table>
<row>
<vertrule width="4pt">
<c>UNIX</>
<vertrule width="2pt">
<c>Betriebssystem</>
<vertrule width="4pt">
</row>
<horirule width="2pt"><horirule width="2pt">
<row>
<vertrule width="4pt">
<c>TeX</>
<vertrule width="2pt">
<c>Public Domain Formatierer</>
<vertrule width="4pt">
</row>
</table>
```

3.7.2 Attribute

Die Festlegung der Linien durch **Attribute** kann durch *direkte* Angabe bei den Zellen, Reihen etc. oder durch *Vererbung* (s. o.) erfolgen. Bei der Vergabe der Attribute in der DTD ist darauf zu achten, daß jedes Linienstück gesetzt werden kann. Außerdem muß geregelt sein, was passiert, wenn sich unter Umständen zwei Attributwerte widersprechen. Es können *boolesche* Attribute oder ein *Aufzählungs*-Attribut verwendet werden. Bei booleschen Attributen wird für jede der vier Linien ein Attribut vorgesehen, das die Werte Linie sichtbar/unsichtbar annehmen kann (s. Bsp. 3–21). Bei einem Aufzählungs-Attribut werden die Sichtbarkeiten aller Linien in einem Attribut angegeben (s. Bsp. 3–22).

Beispiel 3–21 Direkte Angabe mit booleschen Attributen

```
<!-- Teil-DTD: -->
<!ELEMENT table  - - (row+) >
<!ELEMENT row    - - (c+) >
<!ATTLIST row    ruleabove    NUMBER    "1"
                 arulewidth   NUTOKEN   "2pt"
                 rulebelow    NUMBER    "1"
                 brulewidth   NUTOKEN   "2pt" >
<!ELEMENT c      - - (#PCDATA) >
<!ATTLIST c      ruleleft     NUMBER    "1"
                 lrulewidth   NUTOKEN   "2pt"
                 ruleright    NUMBER    "1"
                 rrulewidth   NUTOKEN   "2pt" >

<!-- Instanz: -->
<table>
<row ruleabove="0">
<c lrulewidth="4pt">UNIX</>
<c rrulewidth="4pt">Betriebssystem</>
</row>
<row rulebelow="0">
<c lrulewidth="4pt">TeX</>
<c rrulewidth="4pt">Public Domain Formatierer</>
</row>
</table>
```

Beispiel 3–22 Vererbung von Tabellenlinien mit einem Aufzählungs-Attribut

```
<!-- Teil-DTD: -->
<!ELEMENT table  - - (row+) >
<!ATTLIST table  rules        NAMES     "Left Right Top Bottom"
                 rulewidth    NUTOKEN   "2pt" >
<!ELEMENT row    - - (c+) >
<!ATTLIST row    rules        NAMES     #IMPLIED
                 arulewidth   NUTOKEN   #IMPLIED
                 brulewidth   NUTOKEN   #IMPLIED >
<!ELEMENT c      - - (#PCDATA) >
<!ATTLIST c      rules        NAMES     #IMPLIED
                 lrulewidth   NUTOKEN   #IMPLIED
                 rrulewidth   NUTOKEN   #IMPLIED >

<!-- Instanz: -->
<table rules="Left Right" rulewidth="4pt">
<row>
<c>UNIX</>
<c lrulewidth="2pt">Betriebssystem</>
</row>
<row rules="Top" arulewidth="2pt">
<c>TeX</>
<c lrulewidth="2pt">Public Domain Formatierer</>
</row>
</table>
```

3.8 Gebiete

Gebiete innerhalb von Tabellen dienen dazu, verschiedene Zellen einschließlich ihrer Daten zu gruppieren. Damit könnten z.B. sog. Vorzugswerte[23)] gekennzeichnet werden, die bei einer Recherche gezielt zu finden sind. Eine solche Gruppierung könnte optisch durch eine gemeinsame Hintergrundfarbe, eine Abtrennung durch eine ausgewählte Linienart von den übrigen Zel-

23) Vorzugswerte: Diese Werte sind bei der Auswertung bzw. Anwendung einer reinen Wertetabelle bevorzugt auszuwählen.

len oder ähnlichem dargestellt werden. Es scheint am sinnvollsten, Gebiete zunächst in Deklarationselementen zu definieren, dort (wenn notwendig) eine optische Darstellung festzulegen und die deklarierten Gebiete bei den einzelnen Zellen zu referieren (s. Bsp. 3–23). In der DTD sollte der Satz von möglichen Gebieten und/oder optischen Darstellungen festgelegt sein, um die Realisierung in einem SGML-System zu vereinfachen.

Beispiel 3–23 Ein Gebiet

```
<!-- Teil-DTD: -->
<!ELEMENT table     - - (areadecl, row+) >
<!ELEMENT areadecl  - O EMPTY >
<!ATTLIST areadecl  name      ID                                  #REQUIRED
                    type      (queryterm | important | select)    #REQUIRED >
<!ELEMENT row       - - (c+) >
<!ELEMENT c         - - (#PCDATA) >
<!ATTLIST c         arearef   IDREF                               #IMPLIED >

<!-- Instanz: -->
<table>
<areadecl name="terms" type="queryterm">
<row>
<c arearef=terms>UNIX</>
<c>Betriebssystem</>
</row>
<row>
<c arearef=terms>TeX</>
<c>Public Domain Formatierer</>
</row>
</table>
```

4 Klassifikation von Tabellen

Aus den genannten Kriterien zur strukturellen Darstellung von Tabellen lassen sich drei Klassen von Tabellenbeschreibungen in SGML ableiten: Standardtabellen, freie Tabellen oder freie Standardtabellen. Diese Einteilung ist natürlich sehr grob, sie spiegelt aber die Anforderungen, die an Tabellen in Dokumenten gestellt werden, wider.

> *Anmerkung*: Eine mehr Layout-orientierte Klassifizierung von Tabellen in SGML wird von R. Eric Thomas, Rutherford Appleton Laboratory in [40] beschrieben. Er unterscheidet die Klassen Aufzählung (*Lists*), Tabelle ohne Spaltenüberschrift (*No Headings*), Tabelle mit einzeiliger Spaltenüberschrift (*Single Line Column Headings*), Tabelle mit mehrzeiligen Spaltenüberschriften (*Multiline Column Headings*), Tabelle mit einer zusammengefügten Zelle mit Spaltenüberschriften (*Spanned Headings*), Tabelle mit einer zusammengefügten Zelle mit Haupt- und einzelnen Zellen mit Unterspaltenüberschriften (*Full Spanned Headings and Subheadings*), Tabelle mit mehreren zusammengefügten Zellen mit Haupt- und Unterüberschriften (*Partial Spanned Headings and Subheadings*) und komplexe Tabellen (*Complex*).

4.1 Standardtabellen

Bei Standardtabellen stehen die Daten und die Semantik der Daten im Vordergrund. Die Tabelle wird dabei nur als einheitliches Präsentationsmittel der Daten benutzt. Standardtabellen haben demnach eine feste Spaltenzahl, feste Spaltenbreiten und feste Spaltenüberschriften im Tabellenkopf. Lediglich die Anzahl der Reihen ist variabel. Die einzelnen Tabellenzellen haben abhängig von der Spalte, in der sie stehen, einen dedizierten Inhalt.

In SGML werden Standardtabellen reihenweise beschrieben, wobei nur die Zelleninhalte durch dedizierte Tags mit einfachem Inhalt markiert werden, nicht aber die Zellen selbst. Diese Tags können unter Umständen auch implizite Reihen einleiten. Bei der Definition der Tabelle werden die Angaben des Tabellenkopfes, des Tabellenfußes, der geometrischen Spezifikationen, die Angaben zum Zusammenfügen von Zellen, der Linien und der Gebiete nicht gebraucht und deshalb weggelassen. Alle diese Parameter haben für eine Standardtabelle feste Werte, die dem Formatierer für die einzelnen Standardtabellen bekannt sein müssen. Die Parameter sind **nicht** in der SGML-Datei des Dokumentes enthalten, ebenso tauchen die Spaltenüberschriften des Tabellenkopfes nicht in der SGML-Datei auf.

Standardtabellen eignen sich am besten zur Darstellung von fest vorgegebenen Datensätzen einer Datenbank. Da jedes Datum mit einem eindeutigen SGML-Tag markiert ist, können die Daten automatisch aus der Datenbank in das Dokument integriert werden (Compose-Mechanismus). Ebenso automatisch können die Daten aus dem Dokument extrahiert werden (Decompose-Mechanismus) und damit die Datenbank aufgebaut bzw. erweitert werden[24)].

Mit Standardtabellen lassen sich auch Datensätze darstellen, die baumartige Abhängigkeiten zwischen den einzelnen Daten besitzen. Eine solche Tabelle wird *hierarchische Standardtabelle* genannt (s. Bsp. 4–1, 4–2).

[24)] Natürlich ist dies auch mit Nicht-Standardtabellen möglich. Die Compose/Decompose-Mechanismen sind aber weitaus komplizierter. Dies gilt besonders für das Compose, da das Standard-Layout der Tabelle dem Programm in SGML-Notation bekannt sein muß, das den Compose-Vorgang durchführt.

Beispiel 4–1 Einfache hierarchische Standardtabelle

Software-Typ	Original	Derivate
Editor	GNU-Emacs	Lemacs
		Demacs
		Epoch
	VI	VIP
Betriebssystem	AT&T UNIX	BSD
		Solaris
		HP–UX
	MS-DOS	ATARI-TOS

Die Abhängigkeit der Daten untereinander wird in SGML durch geschachtelte Gruppierungen bzw. Unterelemente gekennzeichnet. Die Tabelle spiegelt diese Zusammenhänge durch vertikal zusammengefügte Tabellenzellen wider. Dabei werden soviele Zellen vertikal zusammengefügt, wie es logische Unterstrukturen gibt (s. Abb. 4–1). Der automatische Aufbau dieser Tabellen ist allerdings wesentlich aufwendiger. Dies gilt insbesonders für interaktive Editoren, da ständig sichergestellt sein muß, daß die richtige Anzahl von Zellen zusammengefügt ist[25].

Beispiel 4–2 Komplexe hierarchische Standardtabelle

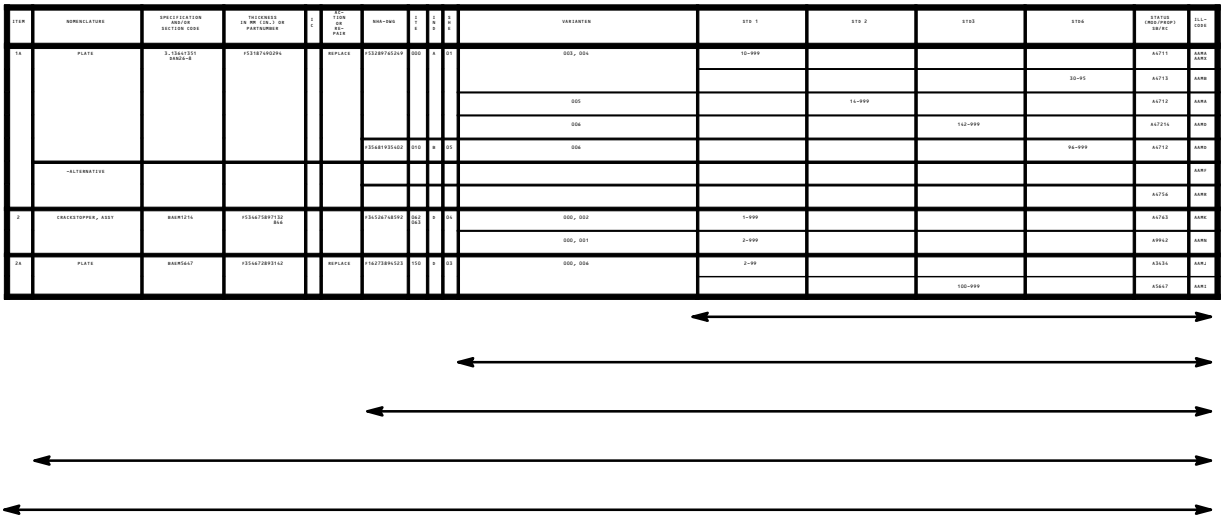

Abbildung 4–1 Hierarchienebenen in einer komplexen Standardtabelle

4.1.1 Anforderungen

Um eine Tabelle zu einer Standardtabelle zu machen, muß ihre Struktur wirklich fest sein, da diese in der DTD fixiert wird. Eine Standardtabelle enthält in der DTD keinerlei Möglichkeiten (Elemente, Attribute), um das Layout zu beeinflussen (siehe dazu Abschnitt 4.3 *Freie Standard-*

[25] Zusätzliche Probleme ergeben sich, wenn die Abhängigkeiten durch geschachtelte Unterelemente und nicht durch Gruppierungen beschrieben werden, da der Editor dann auch für die korrekte Anzahl von Start-/End-Tags und deren Plazierung sorgen muß.

tabellen). Da die „Programmierung“ von SGML-Systemen zur Darstellung (Formatierung) von Tabellen im allgemeinen und Standardtabellen im besonderen sehr aufwendig werden kann, sollte die Anzahl der Standardtabellen, die in einer DTD vorhanden sind nicht zu groß werden. Aufwand[26)] und Nutzen[27)] sollten in einem vernünftigen Verhältnis zueinander stehen. Dies bedeutet aber auch, daß bei der Verwendung eines SGML-Systems, in das sich keine neuen Tabellenstrukturen integrieren lassen, auf Standardtabellen verzichtet werden muß.

4.1.2 DTDs für Standardtabellen

An zwei Beispielen wird aufgezeigt, wie eine einfache und eine hierarchische Standardtabelle formatiert und die dazugehörenden DTDs und die jeweiligen SGML-Instanzen aussehen könnten.

Beispiel 4–3 Eine einfache Standardtabelle (Definitionsliste)

Term	Definition
UNIX	Betriebssytem
TeX	Public Domain Formatierer

Die DTD:

```
<!ELEMENT dl – – (term, definition)+ >
<!ELEMENT (term | definition) – O (#PCDATA) >
```

Die Instanz:

```
<dl>
 <term>UNIX
 <definition>Betriebssystem
 <term>TeX
 <definition>Public Domain Formatierer
</dl>
```

Beispiel 4–4 Eine hierarchische Standardtabelle

Nation	Hauptstadt	Land	Hauptstadt	Stadt
Deutschland	Berlin	Hessen	Wiesbaden	Darmstadt
				Frankfurt
				Kassel
		Bayern	München	Augsburg
				Nürnberg

26) Kosten zur Realisierung der Standardtabellen im SGML-System.

27) Einfacher Datenbankzugriff; Compose/Decompose-Mechanismen; Recherche-Möglichkeiten; einfache Benutzung durch Autoren, da Layout immer stimmt und nicht eingestellt werden muß.

Nation	Hauptstadt	Land	Hauptstadt	Stadt
U.S.A.	Washington	California	Sacramento	Los Angeles
				San Diedo
				San Francisco
		Texas	Austin	Dallas
				Houston

Die DTD mit Gruppierungen:

```
<!ELEMENT nations – – (nation, capital, (state, capital, city+)+)+ >
<!ELEMENT (nation | capital | state | city) – O (#PCDATA) >
```

Die Instanz:

```
<nations>
 <nation>Deutschland
 <capital>Berlin
 <state>Hessen
 <capitel>Wiesbaden
 <city>Darmstadt
 <city>Frankfurt
 <city>Kassel
 <state>Bayern
 <capital>M&uuml;nchen
 <city>N&uuml;rnberg
 <city>Augsburg
 <nation>U.S.A.
 <capital>Washington
 <state>California
 <capital>Sacramento
 <city>Los Angeles
 <city>San Diego
 <city>San Francisco
 <state>Texas
 <capial>Austin
 <city>Dallas
 <city>Houston
</nations>
```

Die DTD mit Unterelementen:

```
<!ELEMENT nations – – (nation)+ >
<!ELEMENT nation – – (nationname, capital, state+) >
<!ELEMENT state – – (statename, capital, city+) >
<!ELEMENT (nationname | capital | statename | city) – – (#PCDATA) >
```

Die Instanz:

```
<nations>
 <nation>
  <nationname>Deutschland
  <capital>Berlin
  <state>
   <statename>Hessen
   <capitel>Wiesbaden
   <city>Darmstadt
   <city>Frankfurt
   <city>Kassel
  </state>
  <state>
```

```
    <statename>Bayern
    <capital>M&uuml;nchen
    <city>N&uuml;rnberg
    <city>Augsburg
   </state>
  </nation>
  <nation>
   <nationname>U.S.A.
   <capital>Washington
   <state>
    <statename>California
    <capital>Sacramento
    <city>Los Angeles
    <city>San Diego
    <city>San Francisco
   </state>
   <state>
    <statename>Texas
    <capial>Austin
    <city>Dallas
    <city>Houston
   </state>
  </nation>
</nations>
```

4.2 Freie Tabellen

Bei freien Tabellen steht aus SGML-Sicht das Tabellenlayout im Vordergrund. Das Aussehen einer freien Tabelle wird sehr aufwendig mit einer Vielzahl von SGML-Markierungen und Attributen beschrieben, während die Bedeutung (Semantik) der Inhalte der Tabellenzellen nicht so genau bestimmt ist wie bei Standardtabellen. Insbesondere ist die Anzahl von Reihen und Spalten nicht vorhersehbar und darf dementsprechend nicht in der DTD fixiert sein.

Eine freie Tabelle besitzt allgemeine Zellen-Tags mit meist komplexem Inhalt. Die Reihen (oder Spalten) werden explizit angegeben; ebenso sind Tabellenkopf und Tabellenfuß notwendig. Der Umfang der Angaben von geometrischen Spezifikationen, von Tabellenlinien und Gebieten sowie das Zusammenfügen von Zellen sind von den Anforderungen der Dokumentart abhängig. In allgemeingültigen Tabellen-DTDs sollten sie aber in vollem Umfang unterstützt werden.

Da die Implementierung von freien Tabellen in einem SGML-Editor grundsätzlich sehr aufwendig ist, sollte man auf solche SGML-Tabellenbeschreibungen zurückgreifen, die bereits von den verschiedenen Editoren unterstützt werden. Dabei handelt es sich entweder um eine proprietäre (hersteller-spezifische) oder eine normierte Tabellen-DTD. Der Vorteil einer normierten Tabellen-DTD liegt eindeutig in der Verfügbarkeit in mehreren SGML-Systemen. Eine solche normierte Beschreibung ist der Tabellenteil der CALS-DTD[28)]. Die DTD ist in dem militärischen U.S.-Standard MIL-M-28001 [42] niedergeschrieben. Leider deckt diese Notation nicht alle Anforderungen, die beliebige Tabellen an die DTD stellen, ab (s. u.), da sie ursprünglich für die Dokumente bzw. Tabellen des DoD festgelegt wurde. Für die CALS-Notation von Tabellen spricht aber ihre große Verbreitung in SGML-Systemen und ihre Verwendung in zahlreichen

[28)] CALS: Computer Aided Acquisition and Logistics Support. Initiative des U.S.-amerikanischen Verteidigungsministeriums (DoD, Department of Defemse) zur einheitlichen Handhabung von DoD-Dokumenten in SGML.

anderen DTDs (z.B. Automobilindustrie: J2008, Normen: EUROSTD, Ölindustrie: Shell U.K. Exploration and Production Ltd. (EXPRO)).

4.2.1 Anforderungen

Es ist nur dann notwendig, eine freie Tabelle in die DTD aufzunehmen, wenn in den Dokumenten beliebige Tabellen vorkommen. Sind freie Tabellen notwendig, hängen alle weiteren Entscheidungen stark von der Anpaßbarkeit des einzusetzenden SGML-Systems (Editor, Formatierer) ab. Darunter fallen,

- ob eine neue oder eine bereits existierende (proprietäre oder normierte) Tabellen-Notation in die DTD aufgenommen wird,
- ob man bei der Wahl einer existierenden Notation eine echte Untermenge[29)] von dieser bildet, die auf die eigenen Anforderungen zugeschnitten ist, falls man nicht alle Eigenschaften benötigt, und
- ob freie Tabellen als Standard-Tabellen verwendet werden sollen, wie dies in der DTD der Automobilindustrie J2008 gemacht wird[30)].

  Anmerkung: Entgegen dem Grundgedanken von SGML, das Dokumente unabhängig von Hard- und Software beschreiben will, werden Tabellen häufig sehr wohl systemabhängig beschrieben. Dies hat damit zu tun, daß die SGML-Systeme zwar beliebige DTDs zulassen, deren Layoutumsetzung aber im Bereich Tabellen meist nicht anpaßbar ist. Hier sind von den Software-Herstellern noch Entwicklungsarbeiten zu leisten.

4.2.2 DTDs für freie Tabellen

DTDs für freie Tabellen werden ausführlich im Kapitel 5 untersucht und bewertet.

4.3 Freie Standardtabellen

Hierbei handelt es sich um Standardtabellen, die in durch die DTD fest vorgegebenen Grenzen veränderbar, also frei sind. Dies kann sich auf die Breite bestimmter Spalten, auf unterschiedliche Spaltenüberschriften oder ähnliches beziehen. D.h. man hat eine parametrisierbare Standardtabelle. Die Parameter können sich auf die Zelleninhalte, aber auch auf Eigenschaften der Spalten, Reihen und Zellen beziehen. Sind alle Eigenschaften mit einem Parameter belegt, so erhält man im Prinzip wieder eine freie Tabelle.

4.3.1 Anforderungen

Für freie Standardtabellen gelten analog die gleichen Anforderungen wie für Standardtabellen.

4.3.2 DTDs für freie Standardtabellen

Das nachfolgende Beispiel 4–5 ist eine Erweiterung der zweispaltigen Standardtabelle aus Beispiel 4–3.

[29)] Untermenge (Subset) bedeutet, daß alle nach der Subset-DTD erstellten Tabellen auch nach der original DTD gültig sind, aber nicht umgekehrt.

[30)] In der J2008-DTD sind alle Standardtabellen als „General Entities“ in der Notation der CALS-Tabellen (s.u.) mit vorbelegten Daten für Spaltenüberschriften, Spaltenbreiten, Tabellenlinien etc. definiert. Diese Entities müssen in den Dokumenten nur referiert und die eigentlichen Tabellendaten eingegeben werden.

Je eine Kopf- und Fußreihe kann angegeben werden, die Breite der ersten Spalte läßt sich beeinflussen[31)] und die horizontale Ausrichtung der Zelleninhalte kann auf linksbündig bzw. zentriert gestellt werden.

Beispiel 4–5 Zweispaltige, freie Standardtabelle

Term	**Definition**
UNIX	Betriebssytem
TeX	Public Domain Formatierer

Die DTD:

```
<!ELEMENT tab–2–col – – (headrow?, row+, footrow?) >
<!ATTLIST tab–2–col col1–width NUTOKEN ”35mm”
<!ELEMENT (headrow | row | footrow) – – (cell, cell) >
<!ELEMENT cell – – (#PCDATA) >
<!ATTLIST cell align (left | center) ”left” >
```

Die Instanz:

```
<tab–2–col col1–width=”20mm”>
<headrow>
<cell align=”center”>Term</>
<cell>Definition</>
</headrow>
<row>
<cell align=”center”>UNIX</>
<cell>Betriebssystem</>
</row
<row>
<cell align=”center”>TeX</>
<cell>Public Domain Formatierer</>
</row>
</tab–2–col>
```

[31)] Die zweite Spalte paßt sich in der Breite automatisch an, so daß die Tabelle immer Seitenbreite hat.

5 Untersuchung von DTDs für freie Tabellen

Anhand einer Beispieltabelle sollen die verschiedenen existierenden Notationen für freie Tabellen vorgestellt und kurz bewertet werden. Im Kapitel 6 wird eine Idee zur Darstellung von Tabellen mit „Koordinatensystemen“ vorgestellt.

Beispiel 5–1 Eine freie Tabelle[32)]

Tabelle 1: Weinpreise der letzten Jahre

Jahrgang	Weinsorte	
	Riesling [DM/Liter]	Silvaner [DM/Liter]
1990	2,48 – 7,32	3,35 – 8,11
1991	4,12 – 9,87	3,48 – 8,56
1993	5,05 – 10,63	4,97 – 10,33

5.1 Tabellen des Verlages Addison-Wesley

TextBook-DTD

Im Buch *SGML — An Author's Guide to the Standard Generalized Markup Language* [4] veröffentlichte der Autor Martin Bryan die TextBook-DTD des Addison-Wesley Verlages. Nach dieser DTD ist (nach Angabe des Autors) auch das Buch selbst erfaßt.

Der Tabellenmodell (<table>) der DTD (s. Anhang A.1) ist relativ einfach gehalten. Er erlaubt die Angabe einer Tabellennummer (<nt>), einer Tabellenüberschrift (<ht>), des Tabellenkopfes (<hc>), des Tabellenrumpfes (<bt>) und des Tabellenfußes (<ft>). Der Tabellenkopf, der Tabellenrumpf und der Tabellenfuß bestehen jeweils aus mehreren Reihen (<r>), die wiederum mehrere Zellen (<c>) enthalten. Eine Reihe kann aber auch wieder einen Tabellenkopf, Tabellenrumpf oder Tabellenfuß enthalten; damit werden vertikal zusammengefügte Zellen beschrieben (s.u.). Horizontales Zusammenfügen erlaubt ein Zellen-Attribut (straddle).

Es fehlen Attribute zu jeglichen Angaben bzgl. Tabellenlinien; ebenso können keine Spaltenbreiten angegeben werden, ein Ausrichten der Zelleninhalte ist weder in vertikaler noch in horizontaler Richtung möglich und es können keine Gebiete und Hintergrundmuster definiert werden.

Die Beispieltabelle in Addison-Wesley-Notation:

```
<table cols=”3”>                    <!-- Attr. fuer Linienbreite fehlt.          -->
                                    <!-- Attr. fuer Spaltenbreiten fehlt.        -->
 <nt>1
 <ht>Weinpreise der letzten Jahre
 <hc>
  <r>
```

32) Das Beispiel ist angelehnt an das Beispiel von R.Eric Thomas in [40], S. 15 ff.

```
   <c>Jahrgang                          <!-- Attr. zum vertikalen Ausrichten fehlt.   -->
                                        <!-- Attr. zum horizontalen Ausrichten fehlt. -->
   <hc cols="2">                        <!-- Attr. zum Ausblenden der Linien fehlt.   -->
    <r><c straddle="2">Weinsorte
    <r><c>Riesling [DM/Liter]
       <c>Silvaner [DM/Liter]
   </hc>
  </r>
 </hc>
 <bt>
  <r><c>1990<c>2,48 - 7,32<c>3,35 - 8,11
  <r><c>1991<c>4,12 - 9,87<c>3,48 - 8,56
  <r><c>1992<c>5,05 - 10,63<c>4,97 - 10,33
 </bt>
</table>
```

5.2 Air Transport Association (ATA) of America

DTD für Aircraft Maintenance Manual

In dem Handbuch ATA-100 [1] legt die ATA den Aufbau (Struktur, Layout) der verschiedenen, in der Luftfahrt verwendeten Handbücher fest. Dies sind im wesentlichen verschiedene Wartungshandbücher für Flugzeuge. Diese Festlegungen existieren schon sehr lange und haben sich sowohl bei den Flugzeugherstellern als auch bei den Fluggesellschaften international durchgesetzt. In den letzten Jahren arbeitet die ATA daran, diese (Struktur-) Festlegungen auf SGML abzubilden. Dabei entstehen eine Reihe von DTDs, die unter Umständen nicht konform zueinander sind. Der hier betrachtete Tabellenteil ist ein Extrakt aus der DTD des Aircraft Maintenance Manuals (s. Anhang A.2).

Die Tabellennotation der ATA ist verwandt mit der CALS-Notation (s. Abschnitt 5.5). Sie ist eine angepaßte Untermenge von dieser und hat entsprechend weniger Variationsmöglichkeiten. Das Tabellenmodell (<ctable>) besteht aus einer Tabellenüberschrift (<title>), den Spaltendeklarationen (<coldef>), dem Tabellenkopf (<thead>) und dem Tabellenrumpf (<tbody>). Tabellenkopf und Tabellenrumpf bestehen aus mehreren Reihen (<ctabrow>), die aus mehreren Zellen (<cell>) zusammengesetzt sind.

Für die Tabelle werden mittels Attribute die Spaltenzahl (totalcol) und die Linien zwischen den Spalten und Reihen (colsep, rowsep) spezifiziert. Mit der Spaltendeklaration wird für jede Tabellenspalte die Breite (width), die horizontale Ausrichtung der Zellendaten (posn) und die Linie zwischen zwei Spalten (colsep) durch Attribute angegeben. Zellen können über Attribute horizontal (colspan) und vertikal (rowspan) zusammengefügt werden.

Es können keine Linienbreiten und -arten angegeben und Linienstücke ausgeblendet werden. Außerdem gibt es keine Möglichkeit zum vertikalen Ausrichten von Zelleninhalten und es können keine Fußreihen spezifiziert werden. Zudem können keine Gebiete definiert werden und auch der Zellenhintergrund kann nicht mit einer Farbe oder einem Muster belegt werden.

Die Beispieltabelle in ATA-100-Notation:

```
<ctable totalcol="3">
 <title>Weinpreise der letzten Jahre
 <coldef colnum="1" width="30mm" posn="l">
 <coldef column="2" width="50mm" posn="c">
 <coldef column="3" width="50mm" posn="c">
```

```
                                        <!-- Attr. fuer Linienbreiten fehlt.       -->
 <thead>
  <ctabrow>                            <!-- Attr. zum Ausblenden der Linie fehlt.  -->
   <cell rowspan="2">Jahrgang          <!-- Attr. fuer vertikale Ausrichtung fehlt. -->
   <cell colspan="2">Weinsorte
  <ctabrow>
   <cell>Riesling [DM/Liter]           <!-- Attr. zum Ausblenden der Linie fehlt.  -->
   <cell>Silvaner [DM/Liter]
 </thead>
 <tbody>
  <ctabrow><cell>1990<cell>2,48 - 7,32<cell>3,35 - 8,11
  <ctabrow><cell>1991<cell>4,12 - 9,87<cell>3,48 - 8,56
  <ctabrow><cell>1992<cell>5,05 - 10,63<cell>4,97 - 10,33
 </tbody>
</ctable>
```

5.3 ArborText Publisher

Interne DTD

Der Publisher der ArborText, Inc. [2] ist ein SGML-Editor, der beliebige DTDs mit in FOSI spezifiziertem Layout verarbeiten kann. Obwohl der Editor als Ausgabe auch die CALS-Tabellen-Notation unterstützt, basiert seine interne Darstellung der Tabellen auf einer anderen, produktspezifischen DTD (s. Anhang A.3). Der Benutzer kommt allerdings mit den Tags der internen DTD nicht in Berührung, da der Editor zur Bearbeitung von Tabellen eine graphisch-interaktive Benutzerschnittstelle besitzt.

Die DTD ist (scheinbar) sehr eng an die programminterne Darstellung angelehnt, um den Konvertierungsaufwand beim Lesen und Schreiben der SGML-Instanzen so gering wie möglich zu halten. Dadurch ist der Tabellenteil der DTD sehr layoutorientiert. Entsprechend sind die DTD und die damit erstellten Instanzen nicht sehr gut lesbar. Das Tabellenmodell (<table>) ist sehr einfach, es bestehlt aus einer Reihenlinie (<rowrule>, horizontale Tabellenlinie) gefolgt von mehreren Tabellenreihen (<tablerow>) und Reihenlinien. Die Reihe besteht aus einer Zellenlinie (<cellrule>, vertikale Tabellenlinie) gefolgt von mehreren Zellen (<tablecell>) und Zellenlinien. Alle weiteren Angaben über Kopfreihen, Eigenschaften der Tabellenlinien, Zusammenfügen von Zellen etc. werden durch Attribute festgelegt. Fußreihen können nicht angegeben werden.

Es können keine Gebiete definiert werden.

Da diese Tabellen-Notation eher wie die ASCII-Ausgabe eines *normalen* Publishing Systems aussieht, ist diese Tabellen-DTD, was die Layoutanforderungen anbelangt, sehr mächtig. Die Daten sind aber nahezu unstrukturiert.

Die Beispieltabelle in ArborText-Notation:

```
<!-- Die Angabe von Tabellenueberschriften ist im Tabellenmodell nicht vorgesehen.  -->
<table cwl="1.18in:1.97in:1.97in" ncols="3" rth="0.3pt"
       wdm="abs" clmarg="3 mm" crmarg="3 mm" dispwid="5.15in">
 <rowrule rtl="+:+:+">
 <tablerow hdr="1">
  <cellrule rty="+">
  <tablecell vspn="2"></tablecell>
  <cellrule rty="-">
  <tablecell chj="c" spn="2">Weinsorte</tablecell>
```

```
 <cellrule rty="+">
</tablerow>
<rowrule rtl="x:.:.">
<tablerow hdr="1">
 <cellrule rty="+">
 <tablecell cvj="c">Jahrgang</tablecell>
 <cellrule rty="-">
 <tablecell chj="c">Riesling [DM/Liter]</tablecell>
 <cellrule rty=".">
 <tablecell chj="c">Silvaner [DM/Liter]</tablecell>
 <cellrule rty="+">
</tablerow>
<rowrule rtl="+:+:+">
<tablerow>
 <cellrule rty="+">
 <tablecell>1990</tablecell>
 <cellrule rty="-">
 <tablecell chj="c">2,48 - 7,32</tablecell>
 <cellrule rty="-">
 <tablecell chj="c">3,35 - 8,11</tablecell>
 <cellrule rty="+">
</tablerow>
<rowrule rtl="-:-:-">
<tablerow>
 <cellrule rty="+">
 <tablecell>1991</tablecell>
 <cellrule rty="-">
 <tablecell chj="c">4,12 - 9,87</tablecell>
 <cellrule rty="-">
 <tablecell chj="c">3,48 - 8,56</tablecell>
 <cellrule rty="+">
</tablerow>
<rowrule rtl="-:-:-">
<tablerow>
<cellrule rty="+">
 <tablecell>1993</tablecell>
 <cellrule rty="-">
 <tablecell chj="c">5,05 - 10,63</tablecell>
 <cellrule rty="-">
 <tablecell chj="c">4,97 - 10,33</tablecell>
 <cellrule rty="+">
</tablerow>
<rowrule rtl="+:+:+">
</table>
```

5.4 Association of American Publishers (AAP)

DTD für Komplexe Tabellen

Die AAP hat eine Reihe von DTDs entwickelt, nach denen Artikel, Zeitschriften und Bücher von Autoren erstellt und von Verlagen publiziert werden können. Diese DTDs wurden national in den U.S.A. standardisiert [31] und befinden sich zur Zeit, nach einer Überarbeitung hinsichtlich der Elementnamen, in der internationalen Normierung [27]. Die DTDs beinhalten ein sehr einfaches Tabellenmodell. Die AAP hat aber auch ein Modell für komplexere Tabellen entwickelt, das in [3] beschrieben wird (s. Anhang A.4). Dieses komplexere Modell wird hier untersucht.

Die Tabelle (<ctbl>) besteht aus dem Tabellenkopf (<cthd>), dem Tabellenrumpf (<ctby>) und dem Tabellenfuß (<ctbf>). Im Tabellenkopf können die Tabellennummer (<no>), die Tabellenüberschrift (<ctt>), die Tabellenunterüberschrift (<ctst>), eine Anmerkung (<cthn>) und die

Kopfreihen (<cthr>) angegeben werden.[33] Eine Kopfreihe beinhaltet entweder mehrere Spaltenüberschriften (<cth>) oder mehrere Spaltenunterüberschriften (<ctsh>), wobei beides Zellen sind. Mit der Kombination von z.B. einer Spaltenüberschrift in der einen und zwei Spaltenunterüberschriften in der nächsten Kopfreihe werden horizontal zusammengefügte Zellen im Tabellenkopf realisiert (s.u.).

Der Tabellenrumpf setzt sich aus mehreren Reihen (<ctr>) zusammen. In den Reihen können mehrere *normale* Datenzellen (<cte>) plaziert sein. Vor der ersten Datenzelle in jeder Reihe kann eine *Kontroll*-Zelle eingefügt werden (*Stub*, engl. für Kontrollabschnitt), die bis zu vier Hierarchiestufen markiert (<ctsb1>, <ctsb2>, <ctsb3>, <ctsb4>).[34] Im Tabellenfuß können nur Anmerkungen (<ctc>) und Tabellenfußnoten stehen (<ctfn>).

Die Tabellenlinien (und deren Eigenschaften) zwischen den Spalten (cs) und Reihen (rs) und die horizontale Ausrichtung der Zellendaten (ca) werden mit Attributen beim Tabellenelement angegeben. Bei den Reihen gibt es ein Attribut für die vertikale Ausrichtung (ra) der Zelleninhalte. Bei den einzelnen Zellen werden die jeweilige horizontale/vertikale Ausrichtung (align, valign) und die Angaben zum vertikalen/horizontalen Zusammenfügen von Zellen (rb, re, cb, ce) definiert. Außerdem können die Zellen mit einem Attribut diagonal aufgesplittet werden (split).

Bei der AAP-Notation fehlen Attribute zur Angabe von Eigenschaften einzelner Linienstücke. Außerdem können keine Gebiete und Hintergrundmuster definiert werden.

Die Beispieltabelle in AAP-Notation:

```
<ctbl rs="0 b 1 bl 2 b 3 s 4 s F b"
      cs="0 b 1 s 2 s F b"
      ca="1 l 2 c 3 c">                        <!-- Attr. fuer Spaltenbreiten fehlt. -->
 <cthd>
  <no>Tabelle 1:<ctt>Weinpreise der letzten Jahre
  <cthr>
   <cth ra="m" rb="1" re="2">Jahrgang
   <cth rb="1" cb="2" ce="3">Weinsorte
  </cthr>
  <cthr>
   <ctsh cb="2">Riesling [DM/Liter]
                                  <!-- Attr. zum Ausblenden des Linienstuecks fehlt. -->
   <ctsh>Silvaner [DM/Liter]
  </cthr>
 </cthd>
 <ctby>
  <ctr><cte>1990<cte>2,48 - 7,32<cte>3,35 - 8,11
  <ctr><cte>1991<cte>4,12 - 9,87<cte>3,48 - 8,56
  <ctr><cte>1992<cte>5,05 - 10,63<cte>4,97 - 10,33
 </ctby>
</ctbl>
```

[33] Die DTD ist an dieser Stelle sehr nachlässig formuliert, da alle diese Elemente in beliebiger Reihenfolge beliebig oft wiederholt werden dürfen.

[34] Eine Kontroll-Zelle wäre z.B. eine Überschrift (der Begriff Seitenschrift wäre passender), die die Inhalte der Daten der Reihe näher erläutert.

5.5 Computer Aided Acquisition and Logistics Support (CALS)

Doc-DTD für MIL–M–38784 Dokumente (MIL–M–28001)

Der militärische U.S.-Standard MIL–M–28001 [42] beinhaltet unter anderem eine DTD für Dokumente nach MIL–M–38784 [44]. Da diese DTD für den amerikanischen Markt sehr relevant ist, wird sie von vielen SGML-Editoren unterstützt. Damit dürfte auch die enthaltene Tabellen-Notation (s. Anhang A.5) die am weitesten verbreitete sein.

Das CALS-Tabellenmodell (<table>) beginnt mit der Tabellenüberschrift (<title>), der die eigentliche Tabelle (<tgroup>) folgt. Dies Element beinhaltet die Deklarationselemente für Spalten (<colspec>) und das horizontale Zusammenfügen von Zellen (<spanspec>), den Tabellenkopf (<thead>), den Tabellenfuß (<tfoot>) und den Tabellenrumpf (<tbody>). Der Tabellenkopf bzw. Tabellenfuß setzt sich aus weiteren Spaltendeklarationen und den Reihen (<row>) zusammen. Der Rumpf besteht nur aus Reihen. Eine Reihe kann aus mehrere Zellen (<entry>) oder Zellenblöcken (<entrybl>) bestehen. Eine Zellenblock ist faktisch eine Tabelle in der Tabelle; er besteht wieder aus Deklarationselementen (<colspec>, <spanspec>), Tabellenkopf und Tabellenrumpf. In den Reihen des Zellenblocks sind weitere Zellenblöcke ausgeschlossen.

Für die gesamte Tabelle wird die Umrandung (frame) und, ob Linien zwischen den Reihen (rowsep) bzw. Spalten (colsep) vorhanden sind, mit Attributen definiert. Zudem kann bestimmt werden, ob die Tabelle im Hoch- oder im Querformat ausgegeben wird (orient) und ob sie Seitenbreite annehmen soll (pgwide). Für die eigentliche Tabelle muß die Spaltenzahl (cols) angegeben werden. Die Linien können bei der eigentlichen Tabelle, bei den beiden Deklarationselementen, bei den Reihen und den einzelnen Zellen individuell gesetzt werden (rowsep, colsep). Die horizontale Ausrichtung kann für die Tabelle, eine Spalte, eine zusammengefügte Zelle und eine einzelne Zelle angegeben werden (align, charoff, char). Die vertikale Ausrichtung gibt es beim Tabellenkopf, -rumpf und -fuß und der Zelle (valign). Über das Deklarationselement <spanspec> werden Zellen horizontal zusammengefügt; vertikales Zusammenfügen wird mit einem Zellenattribut (morerows) realisiert.

Es ist nicht möglich, verschiedene Linienarten bzw. Linienbreiten anzugeben. Zudem können keine Gebiete definiert werden und auch der Zellenhintergrund kann nicht mit einer Farbe oder einem Muster belegt werden.

Die Beispieltabelle in CALS-Notation:

```
<table frame="all" colsep="1" rowsep="1">
 <title>Weinpreise der letzten Jahre
 <tgroup cols="3" align="center">
  <colspec colnum="1" colname="c1" align="left" colwidth="30mm">
  <colspec column="2" colname="c2" colwidth="50mm">
  <colspec column="3" colname="c3" colwidth="50mm">
  <spanspec namest="c2" nameend="c3" spanname="c2-c3" align="center">
                                        <!-- Attr. fuer Linienbreiten fehlt.       -->
  <thead valign="top">
   <row rowsep="0">
    <entry morerows="1" valign="middle">Jahrgang
    <entry spanname="c2-c3">Weinsorte
   <row>
    <entry colsep="0">Riesling [DM/Liter]
    <entry>Silvaner [DM/Liter]
```

```
  </thead>
  <tbody>
   <row><entry>1990<entry>2,48 – 7,32<entry>3,35 – 8,11
   <row><entry>1991<entry>4,12 – 9,87<entry>3,48 – 8,56
   <row><entry>1992<entry>5,05 – 10,63<entry>4,97 – 10,33
  </tbody>
 </tgroup>
</table>
```

5.6 Deutsches Forschungsnetz (DFN)

DAPHNE Report DTD

DAPHNE steht für *Document Application Processing in a Heterogenous Network Environment*. Der Name spiegelt bereits die Intention dieser Entwicklung wider: (wissenschaftliche) Dokumente sollen von verschiedenen Autoren erstellt und über ein WAN (Wide Area Network) verteilt werden [5]. Der DFN-Verein hat als Betreiber des Deutschen Forschungsnetzes mehrere DTDs und zugehörige Programme entwickelt, die es dem Anwender erlauben, Berichte, Artikel, Briefe und Folien einheitlich in SGML zu erstellen und mit TeX [28], troff [35] oder DCF [15] zu formatieren. Die Tabellen-Notation (s. Anhang A.6) ist Teil der DTD für Berichte (Report).

Das Tabellenmodell (<table>) setzt sich aus einem Tabellenkopf (<tabhead>) und mehreren Reihen (<row>) zusammen. Vor dem Tabellenkopf, zwischen Kopf und der ersten Reihe, zwischen den einzelnen Reihen und nach der letzten Reihe können horizontale Linien (<rule>) plaziert werden. Am Ende der Tabelle kann ein Untertitel (<subtitle>) stehen. Der Tabellenkopf besteht aus mehreren Zellen, den Spaltenüberschriften (<colhead>). Die Reihe besteht aus Spalten (<col>), die mehrere Textzeilen (<cline>) beinhalten.

Über ein Attribut der Tabelle wird die horizontale Ausrichtung der Zelleninhalte in den einzelnen Spalten und die durchgehenden Linien zwischen den Spalten bestimmt (schema). Dies ist auch bei den einzelnen Spalten (Zellen) des Tabellenkopfes möglich, nicht aber bei denen des Rumpfes. Weitere Attribute der Tabelle legen die Spaltenbreiten fest (c1, c2, ... c9, call)[35)]. Die Spalten des Tabellenkopfes können zusammengefügt werden (columns), die anderen Spalten (Zellen) wiederum nicht.

Es fehlt das vertikale Ausrichten von Zelleninhalten im Tabellenkopf, das vertikale Zusammenfügen von Zellen überhaupt und das horizontale Zusammenfügen im Tabellenrumpf. Außerdem können im Tabellenrumpf zum einen die Zelleninhalte nicht individuell für jede Zelle ausgerichtet werden und zum anderen die durch die Tabellen-Attribute vorgegebenen Linien nicht individuell für eine Zelle gesetzt werden. Es fehlt die Möglichkeit, Breite und Art der Linien anzugeben. Ebenso können keine Gebiete und Hintergrundmuster definiert werden.

Die Beispieltabelle in DAPHNE-Report-Notation:

```
<!-- Die Angabe von Tabellenueberschriften ist im Tabellenmodell nicht vorgesehen.  -->
<table c1="30" c2="50" c3="50" schema="|l|c|c|">
                                        <!-- Attr. fuer Linienbreiten fehlt.        -->
 <rule>
 <tabhead>
```

35) Die maximale Anzahl an Spalten ist bei der DAPHNE-Report-DTD auf neun beschränkt.

```
 <colhead>Jahrgang       <!-- Attr. zum vertikalen Zusammenfuegen von Zellen fehlt. -->
 <colhead columns="2" schema="|c|">Weinsorte
</tabhead>
<tabhead>
 <colhead>                              <!-- Attr. zum vertikalen Ausrichten fehlt. -->
 <colhead schema="|c ">Riesling [DM/Liter]
 <colhead schema=" c|">Silvaner [DM/Liter]
</tabhead>
<rule>
<row><col><cline>1990<col><cline>2,48 - 7,32<col><cline>3,35 - 8,11
<rule>
<row><col><cline>1991<col><cline>4,12 - 9,87<col><cline>3,48 - 8,56
<rule>
<row><col><cline>1992<col><cline>5,05 - 10,63<col><cline>4,97 - 10,33
<rule>
</table>
```

5.7 Exoterica Corporation

Exoterica Complex Tables

Die Tabellen-DTD *Exoterica Complex Tables* [9] wurde mit dem Ziel entwickelt, eine allgemeingültige und anwendbare DTD, die präsentationsorientiert aufgebaut ist, für Tabellen bereitzustellen. Die DTD ist bewußt so ausgelegt, daß sie einfach an die jeweiligen Anforderungen angepaßt werden kann. Dies wird durch die durchgängige Verwendung von Parameter Entities für alle Elemente und die zusätzlich benötigten Attribute erreicht (s. Anhang A.7).

Die Tabelle (<tbl>) besteht aus einen Tabellentitel (<t.title>), der sowohl Tabellenüberschrift als auch Tabellenunterschrift sein kann, aus einem optionalen Tabellenkopf (<t.head>) und dem erforderlichen Tabellenrumpf (<t.body>). Der Tabellenkopf und der Tabellenrumpf bestehen aus mehreren Reihen (<t.row>), die wiederum aus mehreren Spalten (<t.col>) bestehen. Die Spalten bzw. Zellen können wieder Tabellen beinhalten. Mit einer Tabelle in einer Zelle werden vertikal zusammengefügte Zellen beschrieben.

Mit Attributen können die Breite der Tabelle (wd) und der Spalten (cw), die Linien zwischen Reihen und Spalten (rs, cs), die horizontale und vertikale Ausrichtung der Zelleninhalte (ha, va) und der Abstand des Textes zu den vertikalen und horizontalen Linien (gu, irs) beeinflußt werden. Diese Attribute werden sinnvoll von der Tabelle an Kopf und Rumpf und von diesen an die Reihen vererbt. Ein Zellenattribut bestimmt, wieviele Zellen horizontal zusammengefügt werden sollen (span).

Die Angabe eines Tabellenfußes ist nicht möglich. Ebenso können keine Gebiete und Hintergrundmuster angegeben werden. Ansonsten ist dieses Tabellenmodell vollständig und erfüllt alle Anforderungen.

Die Beispieltabelle in Exoterica Complex Table Notation:

```
<tbl cw="2 3 3" rs="0 b H b A s F b" cs="0 b A s F b" ha="1 l 2 c 3 c" va="A m">
 <title>Weinpreise der letzten Jahre
 <t.head>
  <t.row>
   <t.col>Jahrgang
   <t.col>
    <!-- Begin Subtable -->
    <tbl cw="1 1" rs="0 b 1 k F b" cs="0 s 1 k F b" ha="A c" va="A m">
     <t.body>
      <t.row>
       <t.col span="2">Weinsorte
      </t.row>
      <t.row>
       <t.col>Riesling [DM/Liter]
       <t.col>Silvaner [DM/Liter]
      </t.row>
     </t.body>
    </tbl>
    <!-- End Subtable -->
  </t.row>
 </t.head>
 <t.body>
  <t.row><t.col>1990<t.col>2,48 - 7,32<t.col>3,35 - 8,11
  <t.row><t.col>1991<t.col>4,12 - 9,87<t.col>3,48 - 8,56
  <t.row><t.col>1992<t.col>5,05 - 10,63<t.col>4,97 - 10,33
 </t.body>
</tbl>
```

5.8 HaL Computer Systems International, Ltd. and O'Reilly & Associates, Inc.

DocBook-DTD

Die von HaL Computer Systems International, Ltd. und O'Reilly & Associates, Inc. entwickelte DTD *DocBook* beschreibt Dokumentation für Computer-Systeme und technische Bücher [13]. Das Hauptziel der Entwicklung war, die Migration von existierenden Dokumenten der genannten Dokumentklassen nach SGML zu unterstützen. Die (Teil-) Dokumente liegen in Formaten, wie FrameMaker (Elektronisches Publikations System), troff (UNIX Formatierer) und RTF (Rich Text Format, ASCII-Format von MS-Word) vor. Aus diesem Grund wurde die DTD so entworfen, daß die einzelnen Dokumentteile getrennt voneinander nach SGML konvertiert werden können.

Das Tabellenmodell (<Table>, s. Anhang A.8) ist sehr einfach und setzt sich aus der Tabellenüberschrift (<Title>), der Überschrift-Abkürzung (<TitleAbbrev>), einer Graphik (<Graphic>)[36], einer Tabelle-Spezifikation (<TableSpec>), mehreren Kopfreihen (<TableColHead>) und mehreren Reihen (<TableRow>) zusammen. Die Kopfreihe und die Reihe bestehen aus mehreren Zellen (<TableCell>). Das Spezifikationselement besitzt im Gegensatz zu allen ande-

[36] Die Graphiken sind Referenzen auf externe Dateien, die eine Tabelle in einem anderen Format enthalten. Diese *Graphik*-Tabelle soll bei der Formatierung anstelle dieses Tags erscheinen.

ren hier betrachteten Tabellen-Notationen keine Attribute. Die Layout-Informationen werden als Text des Elementes angegeben[37].

Außer dem Spezifikationselement gibt es keine weiteren Möglichkeiten (Attribute), zur Spezifikation von Linien, Spaltenbreiten, Ausrichtung der Zelleninhalte, zusammengefügten Zellen und der Angabe von Gebieten und Hintergrundmustern.

Die Beispieltabelle in DocBook-Notation:

```
<table>
 <title>Weinpreise der letzten Jahre
 <tablespec>         <!-- ??? Hier koennen Formatierinfos stehen ???                -->
                     <!-- Es fehlen Moeglichkeiten zur Angabe der Spaltenbreiten,    -->
                     <!-- zum horizontalen/vertikalen Ausrichten der Zelleninhalte,  -->
                     <!-- zum Ausblenden von Linien, zur Angabe der Linienstaerken   -->
                     <!-- und zum horizontalen/vertikalen Zusammenfuegen von Zellen. -->
 <tablecolhead>
  <tablecell>Jahrgang
  <tablecell>Weinsorte
  <tablecell>
 <tablecolhead>
  <tablecell>
  <tablecell>Riesling [DM/Liter]
  <tablecell>Silvaner [DM/Liter]
 <tablerow><tablecell>1990<tablecell>2,48 - 7,32<tablecell>3,35 - 8,11
 <tablerow><tablecell>1991<tablecell>4,12 - 9,87<tablecell>3,48 - 8,56
 <tablerow><tablecell>1992<tablecell>5,05 - 10,63<tablecell>4,97 - 10,33
</table>
```

5.9 International Committee on Accessible Document Design (ICADD)

DTD for Braille, Large Print and Computer-Voice

Das International Committee on Accessible Document Design (ICADD) unterstützt die Umsetzung von Publikationen[38] in Darstellungformen für sehbehinderte Menschen. Dies sind die Braille-Schrift, Large Print (Druck der Texte mit großen Schrifttypen) und Computer Voice (Texte werden vom Compter *vorgelesen*). Weitere Informationen zu diesem Thema sind in [33][41] zu finden. Bei der Umsetzung der existierenden (Nachrichten-) Texte in die drei genannten Formen, wurde SGML als Zwischenformat gewählt und eine entsprechende DTD entwickelt. Beliebige Formate werden nach SGML konvertiert und von dort einheitlich nach Braille, Large Print oder Computer Voice überführt. Die DTD muß zum einen die Strukturen der unterschiedlichsten Eingangstexte abdecken und zum anderen eine einfache Konvertierung in jede der drei Ausgabeformen sicherstellen. Aufgrund dieses allgemeinen Ansatzes hat der Tabellenteil der DTD eine gewisse Relevanz, so daß es sinnvoll ist, diese DTD zu untersuchen.

Das Tabellenmodell (<table>, s Anhang A.9), das Ähnlichkeiten mit der SoftQuad-DTD (s. Abschnitt 5.12) und der AAP-DTD (s. Abschnitt 5.4) hat, beinhaltet die Tabellenüberschrift (<h3>), eine Anmerkung (<note>) und die eigentliche Tabelle (<tgroup>). Diese besteht aus einem Deklarationselement für jede Spalte (<coldef>) dem Tabellenkopf (<tabhead>) und in

---

[37] Da in der Dokumentation der DTD leider keinerlei Informationen zu finden sind, wie die Layout-Angaben anzugeben sind, fehlen Sie im nachfolgenden Beispiel.

[38] Es stehen die Formen Tageszeitung und Nachrichtenmagazin im Vordergrund.

beliebiger Reihenfolge[39] dem Tabellenrumpf (<tabbody>) und dem Tabellenfuß (<tabfoot>). Im Tabellenkopf können spezielle Spaltendeklarationen und Kopfreihen (<hdrow>) plaziert werden. Dies gilt auch für den Rumpf; dort sind es allerdings *normale* Reihen (<row>). Der Tabellenfuß kann mehrere Querverweise (<xref>) und Fußnoten (<ftn>) beinhalten. Eine Kopfreihe enthält die Spaltenüberschrfiten (<hdcell>) und Kurztexte (<shortxt>)[40]. Eine Reihe besteht aus verschiedenen Kontroll-Zellen (s.o., (<stubcell>, <sstcell>), *normalen* Zellen (<cell>) und Kurztexten.

Beim Tabellenmodell werden mittels Attributen die Tabellenumrandung (frame) und die Tabellengröße (tblwd, tblht) spezifiziert. Die letzten zwei Angaben können auch bei der eigentlichen Tabelle, beim Tabellenkopf, -rumpf und -fuß angegeben werden. Über die Spaltendeklarationen werden für jede Spalte die Breite (colwd), die horizontale Ausrichtung der Zelleninhalte (halign, alignchr, charpos), die rechte (colsep) und die obere Linie (topsep) der Spalte angegeben. Bei den Kopfreihen und den Reihen wird die vertikale Ausrichtung der Zelleninhalte (valign), die untere (rowsep) und die linke Linie (leftsep) sowie die Reihenhöhe (rowht) bestimmt. Bei den einzelnen Zellen können wiederum die Linien und Ausrichtungen individuell gesetzt werden. Außerdem stehen hier die Informationen zum horizontalen und vertikalen Zusammenfügen der Zellen (colstart, colspan, rowstart, rowspan).

Die Tabellennotation ist sehr mächtig, es fehlen die Möglichkeiten, Linienarten und Linienbreiten anzugeben. Zudem können keine Gebiete definiert werden und auch der Zellenhintergrund kann nicht mit einer Farbe oder einem Muster belegt werden.[41]

Die Beispieltabelle nach der ICADD-DTD:

```
<table frame="box" tblunits="mm">              <!-- Attr. fuer Linienbreiten fehlt. -->
 <h3>Weinpreise der letzten Jahre
 <tgroup>
  <coldef colsep="1" halign="left" colwd="30">
  <coldef colsep="1" halign="center" colwd="50">
  <coldef colsep="1" halign="center" colwd="50">
  <thead>
   <coldef colsep="1" halign="left" colwd="30">
   <coldef colsep="0" halign="center" colwd="50">
   <coldef colsep="1" halign="center" colwd="50">
   <hdrow rowsep="hno" valign="middle">
    <hdcell rowstart="1" rowspan="2">Jahrgang
    <hdcell rowstart="1" colstart="2" colspan="2" halign="center">Weinsorte
   <hdrow rowsep="hyes" valign="middle">
    <hdcell rowstart="2" colstart="2">Riesling [DM/Liter]
    <hdcell rowstart="2" colstart="3">Silvaner [DM/Liter]
  </thead>
  <tbody>
   <row><cell>1990<cell>2,48 - 7,32<cell>3,35 - 8,11
   <row><cell>1991<cell>4,12 - 9,87<cell>3,48 - 8,56
   <row><cell>1992<cell>5,05 - 10,63<cell>4,97 - 10,33
  </tbody>
```

[39] Hiermit kommt man den unterschiedlichen Eingabeformaten entgegen.

[40] Kurztext ist für die Sprachausgabe gedacht. Er dient als Alternativtext, wenn der eigentliche Zelleninhalt zu umfangreich ist.

[41] Es stellt sich auch die Frage, ob solche Angaben für den Verwendungszweck der DTD überhaupt sinnvoll sind.

```
 </tgroup>
</table>
```

5.10 International Organization for Standardization (ISO)

Normen-DTD (ISO/IEC 9573–11:1992)

Im ISO Zentralsekretariat in Genf wurde eine DTD entwickelt, nach der ISO/IEC Normen und Technische Berichte (Technical Report) erstellt werden können [25]. Da Normen komplexe Dokumente sind, ist diese DTD sehr umfangreich. Sie wurde in zwei Teile zerlegt: einen ISO-spezifischen Teil, der die ISO-spezifischen Informationen der Titelseite und des Schlußteils (Frontmatter und Backmatter) beinhaltet, und einen allgemeinen Teil, der die Struktur des Hauptteils von Normen umfaßt. Durch diese Zweiteilung ist es möglich, daß andere Normungsinstitute eigene (Teil-) DTDs für Front- und Backmatter definieren und ebenfalls den gemeinsamen Teil verwenden. Dies wird für Normen des DIN (Deutsches Indtitut für Normung e.V.) und für Werknormen gemacht [8]. Auch auf europäischer Ebene will man so vorgehen. Die Komplexität der Normen im Bereich Tabellen spiegelt sich auch in der DTD wider. Leider ist der Tabellenbereich sehr eng an das zur Zeit bei der ISO verwendete Satzsystem DCF angelehnt, so daß der Aufbau der Tabelle reihenorientiert ist und dadurch manche Informationen nur sehr umständlich angegeben werden können.

Das Tabellenmodell (<tab>, s. Anhang A.10) besteht aus der Tabellenüberschrift (<tabcap>), einer Beschreibung (<tabdesc>), einem Kommentar (<tabcomm>), der eigentlichen Tabelle (<tabmat>) und weiteren Outline-Elementen. Die eigentliche Tabelle setzt sich aus Tabellenkopf (<tabhead>), Tabellenfuß (<tabfoot>) und dem Tabellenrumpf (<tabbody>) zusammen. Tabellenkopf und Tabellenfuß beinhalten je eine *arrangierte* Reihe (<arow>); der Tabellenrumpf besteht aus mehreren von diesen. Eine solche Reihe umfaßt mehrere Zellen (<c>), die sich wiederum aus Unterzellen (<sc>) zusammensetzen. Mit den Unterzellen lassen sich die Zellen vertikal aufsplitten. In den Unterzellen sind neben dem Inhaltsmodell auch Querlinien (<hrule>) zulässig, die eine Unterzelle horizontal aufspalten.[42)]

Beim Tabellenmodell gibt es Attribute zur Angabe der Tabellenbreite (width), ihrer horizontalen Ausrichtung (align) und der Plazierung auf der Seite (place). Mittels Vererbung werden bei der eigentlichen Tabelle, beim Tabellenrumpf, bei den Reihen und Zellen Attribute zur Angabe der Linien (trules, rrules, crules), der Ausrichtungen (cvalign, calign(s), calignp(s)) dem Zusammenfügen von Zellen (arrange) und der Schriftgröße (pointsz) gesetzt. Gebiete können für die Tabelle definiert werden (domains); die einzelnen Zellen können dann die definierten Gebiete referieren (domain). Dei Tabelle kann in 90°-Schritten (rotate) gedreht werden.

Die ISO-Tabellen-Notation ist streng reihenorientiert aufgebaut. Dadurch sind Layoutänderungen, die z.B. nur eine Spalte betreffen sehr aufwendig. Die DTD ist ansonsten sehr mächtig, lediglich die Breite und die Art von Tabellenlinien kann nicht spezifiziert werden. Die Festle-

[42)] Die Layout-Auswirkungen des vertikalen bzw. horizontalen Aufsplittens lassen sich auch durch entsprechende Zusammenfügungen von Zellen erreichen.

gung der Gebiete ist leider ungenügend spezifiziert, so daß hier große Interpretationsfreiheit möglich ist.[43)]

Die Beispieltabelle in ISO-Notation:

```
<tab>
 <tabcap>Weinpreise der letzten Jahre
 <tabmat pointsz="10" gridx="30mm 50mm 50mm" cvalign="C" caligns="L C C">
                                        <!-- Attr. fuer Linienbreiten fehlt.       -->
  <tabhead>
   <arow arrange="1 2 2 / 1 3 4">
    <c><sc>Jahrgang
    <c crules="T L R"><sc>Weinsorte
    <c crules="L B"><sc>Riesling [DM/Liter]
    <c crules="R B"><sc>Silvaner [DM/Liter]
  </tabhead>
  <tabbody>
   <arow><c><sc>1990<c><sc>2,48 - 7,32<c><sc>3,35 - 8,11
   <arow><c><sc>1991<c>sc>4,12 - 9,87<c><sc>3,48 - 8,56
   <arow><c><sc>1992<c><sc>5,05 - 10,63<c><sc>4,97 - 10,33
 </tbody>
 </tabmat>
</tab>
```

5.11 Open Software Foundation (OSF)

OSF-Book-DTD

Die OSF hat für die Dokumentation von Computer Software und Hardware eine DTD entwickelt [32]. Aus den erstellten SGML-Instanzen sollen gleichzeitig die gedruckte wie auch die Online-Dokumentation generiert werden.

Der Aufbau des Tabellenteils (s. Anhang A.11) ist relativ einfach gehalten. Die Tabelle (<table>) besteht aus mehreren Kopfreihen (<title-row>) oder *normalen* Reihen (<row>). Die Kopfreihe beinhaltet Zellen (<cell>), während die Reihe zusätzlich noch komplexe Zellen (<c-cell>)[44)] enthält.

Über vererbte Attribute der Tabelle, Reihen und Zellen lassen sich Linien (colsep, rowsep), die Ausrichtungen (halign, valign) und zusammengefügte Zellen (model) spezifizieren. Bei den Tabellen wird die Anzahl (ncols), die Breite (colwidth) und die Wichtigkeit (colweight, hat keine Layout-Auswirkungen) der Spalten angegeben.

Die Tabellen-Notation ist vollständig; es fehlt die Möglichkeit, Breite und Art der Linien anzugeben. Ebenso können keine Gebiete und Hintergrundmuster definiert werden. Aus der Dokumentation geht nicht klar hervor, ob eine Zelle gleichzeitig horizontal und vertikal mit einer anderen Zelle zusammengefügt werden kann.

Die Beispieltabelle nach dem OSF-Book:

```
<!-- Die Angabe von Tabellenueberschriften ist im Tabellenmodell nicht vorgesehen.  -->
<table ncols="3" colwidth="1.18 1.97 1.97"
```

43) Ein ausführlicher Vergleich der CALS- und ISO-Tabellen-Notationen ist in [38] zu finden.

44) Die komplexen Zellen unterscheiden sich in ihren Inhaltsmodellen, was hier aber nicht näher betrachtet werden soll, da die Tabellenstruktur im Vordergrund steht.

```
      halign="left center center" valign="center center center"
      colsep="1 1 1" rowsep="1 1 1"
      frame="all" repeathead="1">              <!-- Attr. fuer Linienbreiten fehlt. -->
 <title-row model="text,text,span" rowsep="1 0 0">
  <cell>Jahrgang
  <cell>Weinsorte
  <cell>
 <title-row modell="vspan,text,text" colsep="1 0 1">
  <cell>
  <cell>Riesling [DM/Liter]
  <cell>Silvaner [DM/Liter]
 <row><cell>1990<cell>2,48 - 7,32<cell>3,35 - 8,11
 <row><cell>1991<cell>4,12 - 9,87<cell>3,48 - 8,56
 <row><cell>1992<cell>5,05 - 10,63<cell>4,97 - 10,33
</table>
```

5.12 SoftQuad Author/Editor

Interne DTD

Wie bei der ArborText DTD handelt es sich hier um eine Editor-interne DTD des SoftQuad Author/Editor Systems [37]. Der Editor ist (nach Aussage des Herstellers) in der Lage, andere Tabellen-Notationen zu verarbeiten. Der Benutzer kommt allerdings beim Bearbeiten der Tabellen mit SGML nicht in Berührung, da der Editor eine graphisch-interaktive Benutzerschnittstelle besitzt.

Das interne Tabellenmodell (s. Anhang A.12) ist ein Parameter Entity, der den Tabellenkopf (<TblHead>), den Tabellenfuß (<TblFoot>) oder den Tabellenrumpf (<TblBody>) als Auswahl beinhaltet. Jeder dieser drei Tabellenteile besteht aus Spaltendeklarationen (<TblCDefs>) und Reihen (<TblRows>). Die Spaltendeklarationen beinhalten für jede Spalte eine Spaltendeklaration (<TblCDef>). Analog beinhalten die Reihen einzelne Reihen (<TblRow>). Die einzelnen Reihen bestehen aus Zellen (<TblCell>).

Bei der Gesamt-Spaltendeklaration und den einzelnen Spaltendeklarationen werden mit Vererbung die rechten und oberen Linien (ColSep, TopSep), die horizontale Ausrichtung der Zelleninhalte (HAlign, AlignChr, AlignPos) und die Spaltenbreite (ColWd) angegeben. Analog werden bei der Gesamt-Reihe und den einzelnen Reihen die linken und unteren Linien (LeftSep, RowSep), die vertikale Ausrichtung der Zelleninhalte (VAlign) und die Reihenhöhe (RowHt) angegeben. Bei den einzelnen Zellen können die Werte für die Ausrichtungen und die Linien individuell gesetzt werden. Die Informationen über zusammengefügte Zellen stehen ebenfalls bei den Zellen.

Die Tabellen-Notation ist sehr mächtig. Sie erlaubt allerdings nicht die Definition von Gebieten und das Setzen des Zellenhintergrundes.

Die Beispieltabelle in SoftQuad-Notation:

```
<!-- Die Angabe von Tabellenueberschriften ist im Tabellenmodell nicht vorgesehen. -->
<TBLHEAD>
 <TBLCDEFS COLSEP="VSingle" HALIGN="Center" CHARPOS="0"
           COLWD="50" TBLUNITS="mm" TOPSEP="HBold">
  <TBLCDEF COLWD="30" TBLUNITS="mm">
  <TBLCDEF>
  <TBLCDEF COLSEP="VBold">
 </TBLCDEFS>
```

```
 <TBLROWS ROWSEP="HSingle" VALIGN="Middle" LEFTSEP="VBold">
  <TBLROW>
   <TBLCELL HALIGN="Left" CHARPOS="0" COLSTART="1" ROWSPAN="2">Jahrgang</TBLCELL>
   <TBLCELL ROWSEP="HBlank" COLSTART="2" COLSPAN="2">Weinsorte</TBLCELL>
  </TBLROW>
  <TBLROW>
   <TBLCELL COLSEP="VBlank" COLSTART="2">Riesling [DM/Liter]</TBLCELL>
   <TBLCELL COLSTART="3">Silvaner [DM/Liter]</TBLCELL>
  </TBLROW>
 </TBLROWS>
</TBLHEAD>
<TBLBODY>
 <TBLCDEFS COLSEP="VSingle" HALIGN="Left" CHARPOS="0"
           COLWD="50" TBLUNITS="mm" TOPSEP="HBold">
  <TBLCDEF COLWD="30" TBLUNITS="mm">
  <TBLCDEF HALIGN="Center" CHARPOS="0">
  <TBLCDEF COLSEP="VBold" HALIGN="Center" CHARPOS="0">
 </TBLCDEFS>
 <TBLROWS ROWSEP="HSingle" VALIGN="Top" LEFTSEP="VBold">
  <TBLROW>
   <TBLCELL COLSTART="1">1990</TBLCELL>
   <TBLCELL COLSTART="2">2,48 - 7,32</TBLCELL>
   <TBLCELL COLSTART="3">3,35 - 8,11</TBLCELL>
  </TBLROW>
  <TBLROW>
   <TBLCELL COLSTART="1">1991</TBLCELL>
   <TBLCELL COLSTART="2">4,12 - 9,87</TBLCELL>
   <TBLCELL COLSTART="3">3,48 - 8,56</TBLCELL>
  </TBLROW>
  <TBLROW ROWSEP="HBold">
   <TBLCELL COLSTART="1">1992</TBLCELL>
   <TBLCELL COLSTART="2">5,05 - 10,63</TBLCELL>
   <TBLCELL COLSTART="3">4,97 - 10,33</TBLCELL>
  </TBLROW>
 </TBLROWS>
</TBLBODY>
```

6 Tabellen als Koordinatensysteme

Die Idee, freie Tabellen auf ein Koordinatensystem abzubilden, wurde dem Autor gegenüber bereits vor ein paar Jahren von Anders Berglund, Berglund Consulting Inc. geäußert. Weitere Ideen wurden mit Ludo Van Vooren, Avalanche Development Company, Paul Grosso, Arbor-Text Inc. und Yuri Rubinski, SoftQuad Inc. auf der SGML Europe 93 in Amsterdam in einer lebhaften Diskussion über Tabellennotationen ausgetauscht. An dieser Stelle soll das Konzept erstmals umfassend erläutert werden.

6.1 Konzept der Koordinaten-Tabellen

Das Prinzip besteht darin, die Daten einer Reihe/Spalte einer Koordinatenachse in einem mehrdimensinalen Koordinatensystem zuzuordnen. Die Anzahl an Dimensionen entspricht der Anzahl der gleichartigen Reihen-/Spaltenüberschriften, die gleichzeitig den Koordinatenachsen die Namen geben (s. Bsp. 6–1 und Abb. 6–1).

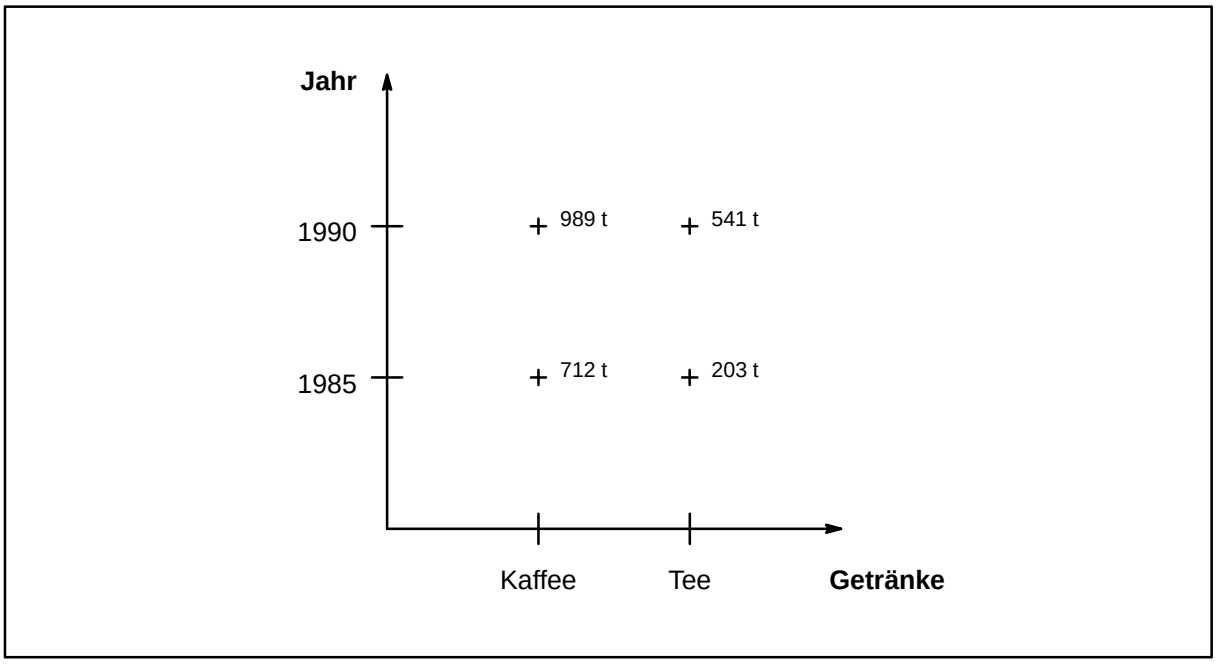

Abbildung 6–1 Die Tabelle aus Bsp. 6–1 als Koordinatensystem

Beispiel 6–1 Eine einfache Tabelle

		Getränke	
		Kaffee	**Tee**
Jahr	**1985**	712 t	203 t
	1990	989 t	541 t

In SGML könnte der Aufbau folgender sein: In einer Koordinaten-Tabelle (<cotab>) spezifiziert man die einzelnen Achsen (<axis>), indem man den Achsennamen (<axname>) und die Achsen-Koordinaten (<axcoord>) angibt. Jede einzelne Achse bekommt über Attribute einen Bezeichner (id) und die Plazierung (place) zugeordnet. Die Werte der Tabelle sind dann die

Koordinaten (<coord>), die sich über Attribute auf die jeweiligen Achsen (axref) und Achsen-Koordinaten (axcoref) beziehen. Die Layout-Informationen, wie Ausrichtung (halign, valign) und Linien (rrule, lrule, trule, brule), werden über Attribute bei <cotab>, <axname>, <axcoord> und <coord> nach dem Vererbungskonzept (s. Abschnitt 3.5) angegeben. Das Zusammenfügen von Zellen ist bei Koordinaten-Tabellen **keine** Layout-Information, diese Informationen sind durch die logischen Zusammenhänge der Daten (Koordinaten) gegeben.

In SGML könnte die Tabelle aus Bsp. 6–1 folgendermaßen aussehen:

```
<cotab halign="center" valign="center">
 <axis id="beverage" place="top">
  <axname brule="0">Getr&auml;nke
  <axcoord>Kaffee<axcoord>Tee
 </axis>
 <axis id="year" place="left">
  <axname rrule="0">Jahr
  <axcoord>1985<axcoord>1990
 </axis>
 <coord axref="beverage year" axcoref="1 1">712 t
 <coord axref="beverage year" axcoref="1 2">989 t
 <coord axref="beverage year" axcoref="2 1">203 t
 <coord axref="beverage year" axcoref="2 2">541 t
</cotab>
```

Beispiel 6–2 Eine etwas kompliziertere Tabelle

		Kalte Getränke		*Warme Getränke*	
		Cola	**Limonade**	**Kaffee**	**Tee**
Jahr	**1985**	11329 t	9799 t	712 t	203 t
	1990	23456 t	24986 t	989 t	541 t

Die neue Spalte „Kalte Getränke“ in Bsp. 6–2 wird einfach als weitere Achse definiert und ihre Daten werden als Koordinaten mit den entsprechenden Referenzen angegeben. Ein bildliche Darstellung ist hier nicht so einfach, da es zwischen den Daten der „Kalten Getränke“ und der „Warmen Getränke“ bewußt keine Beziehung gibt (s. Abb. 6–2).

In SGML würde die Tabelle aus Bsp. 6–2 folgendermaßen aussehen:

```
<cotab halign="center" valign="center">
 <axis id="cold-bev" place="top">
  <axname brule="0">Kalte Getr&auml;nke
  <axcoord>Cola<axcoord>Limonade
 </axis>
 <axis id="warm-bev" place="top">
  <axname brule="0">Warme Getr&auml;nke
  <axcoord>Kaffee<axcoord>Tee
 </axis>
 <axis id="year" place="left">
  <axname rrule="0">Jahr
  <axcoord>1985<axcoord>1990
 </axis>
 <coord axref="cold-bev year" axcoref="1 1">11329 t
 <coord axref="cold-bev year" axcoref="1 2">23456 t
 <coord axref="cold-bev year" axcoref="2 1">9799 t
 <coord axref="cold-bev year" axcoref="2 2">24986 t
 <coord axref="warm-bev year" axcoref="1 1">712 t
```

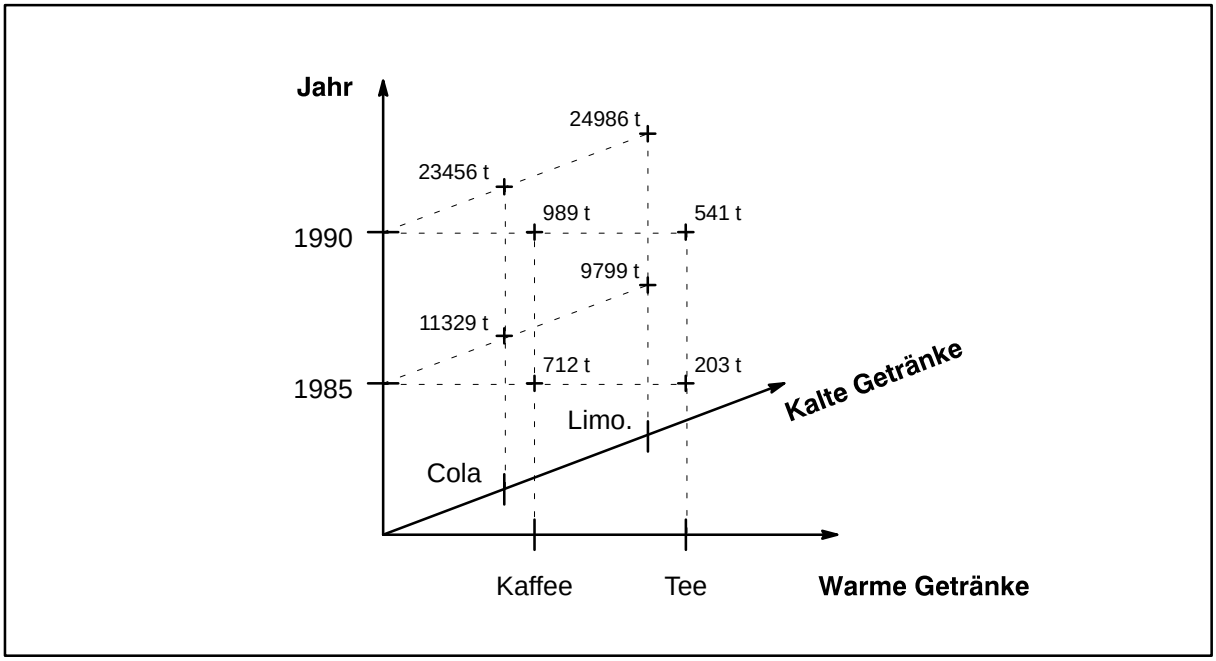

Abbildung 6–2 Die Tabelle aus Bsp. 6–2 als Koordinatensystem

```
 <coord axref="warm-bev year" axcoref="1 2">989 t
 <coord axref="warm-bev year" axcoref="2 1">203 t
 <coord axref="warm-bev year" axcoref="2 2">541 t
</cotab>
```

Beispiel 6–3 Eine noch kompliziertere Tabelle

		Getränke			
		Kalte Getränke		*Warme Getränke*	
		Cola	**Limonade**	**Kaffee**	**Tee**
Jahr	**1985**	11329 t	9799 t	712 t	203 t
	1990	23456 t	24986 t	989 t	541 t

Die in Bsp. 6–3 dargestellte Tabelle ist mit der Spaltenüberschrift „Getränke“ um eine weitere Koordinaten-Achse erweitert worden. Das Besondere an dieser neuen Achse ist aber, daß sie die beiden Achsen „Kalte Getränke“ und „Warme Getränke“ als *Unterachsen* enthält. Dadurch sind diese Unterachsen die Achsen-Koordinaten der neuen „Getränke“-Achse. Da die bildliche Darstellung die Zusammenhänge nicht anschaulicher machen würde, gibt es zu dieser Tabelle keine Abbildung.

Die SGML-Darstellung ist analog zu dem eben Beschriebenen. Die Koordinaten-Tags müssen auf jeden Fall die Achsen der untersten *Achsen-Hierarchie* referieren, um eindeutige Zuordnungen zu erhalten.

In SGML würde die Tabelle aus Bsp. 6–3 folgendermaßen aussehen:

```
<cotab halign="center" valign="center">
 <axis id="beverages" place="top">
  <axname brule="0">Getr&auml;nke
  <axis id="cold-bev" place="top">
```

```
   <axname brule="0">Kalte Getr&auml;nke
   <axcoord>Cola<axcoord>Limonade
  </axis>
  <axis id="warm-bev" place="top">
   <axname brule="0">Warme Getr&auml;nke
   <axcoord>Kaffee<axcoord>Tee
  </axis>
 </axis>
 <axis id="year" place="left">
  <axname rrule="0">Jahr
  <axcoord>1985<axcoord>1990
 </axis>
 <coord axref="cold-bev year" axcoref="1 1">11329 t
 <coord axref="cold-bev year" axcoref="1 2">23456 t
 <coord axref="cold-bev year" axcoref="2 1">9799 t
 <coord axref="cold-bev year" axcoref="2 2">24986 t
 <coord axref="warm-bev year" axcoref="1 1">712 t
 <coord axref="warm-bev year" axcoref="1 2">989 t
 <coord axref="warm-bev year" axcoref="2 1">203 t
 <coord axref="warm-bev year" axcoref="2 2">541 t
</cotab>
```

6.2 Vorteile von Koordinaten-Tabellen

Der Vorteil dieser Betrachtungsweise liegt darin, daß durch einfaches Austauschen der Achsen (ähnlich wie in einem Tabellenkalkulationsprogramm) die Tabelle unter verschiedenen Gesichtspunkten betrachtet werden kann, ohne daß sich die SGML-Darstellung stark ändert. Es müssen lediglich die Reihenfolgen und Plazierungsangaben der Achsen- bzw. Unterachsen-Definitonen verändert werden.

Das Layout der Tabelle bzgl. zusammengefügten Zellen ist implizit durch die Abhängigkeiten der Koordinaten zueinander gegeben. Spezielle Hervorhebungen, Ausrichtungen, Linien oder Ähnliches müssen in SGML angegeben werden; sie sind aber nicht unbedingt notwendig, da die angegebenen, logischen Zusammenhänge der Daten bereits das Layout implizit festlegen.

6.3 Nachteile von Koordinaten-Tabellen

Der Nachteil besteht darin, daß sich komplexere Tabellen, in denen die Abhängigkeiten der Daten zueinander nicht klar erkennbar sind[45)], nur sehr schwer auf ein Koordinatensystem abbilden lassen. Außerdem ist der Entwurf einer allgemeingültigen DTD hierfür recht kompliziert und nachfolgender Ansatz kann nur als ein erster Versuch bewertet werden, der sicherlich noch nicht alle denkbaren Möglichkeiten der logischen Datenstrukturierung in Tabellen abdeckt.

6.4 DTD für Koordinaten-Tabellen

```
<!ELEMENT cotab    - - (axis+,coord+) >
<!ATTLIST cotab    halign    (left | right | center | justif)  "left"
                   valign    (top | middle | bottom)           "top"
                   lrule     NUMBER                            "1"
                   rrule     NUMBER                            "1"
                   trule     NUMBER                            "1"
                   brule     NUMBER                            "1" >
```

[45)] Ist kein Zusammenhang zwischen den Daten erkennbar, stellt sich die Frage, ob sich der Autor beim Aufbau der Tabelle überhaupt „was gedacht“ hat.

```
<!ELEMENT axis     – – (axname?,(axcoord | axis)) >
<!ATTLIST axis     id        NMTOKEN                          #REQUIRED
                   place     (top | left | bottom | right)    #REQUIRED >

<!ELEMENT (axname | axcoord | coord)
                   – O %m.cell; >
<!ATTLIST (axname | axcoord)
                   halign    (left | right | center | justif)  #IMPLIED
                   valign    (top | middle | bottom)          #IMPLIED
                   rrule     NUMBER                           #IMPLIED
                   brule     NUMBER                           #IMPLIED >
<!ATTLIST coord    axref     CDATA                            #REQUIRED
                   axcoref   CDATA                            #REQUIRED
                   halign    (left | right | center | justif)  #IMPLIED
                   valign    (top | middle | bottom)          #IMPLIED
                   rrule     NUMBER                           #IMPLIED
                   brule     NUMBER                           #IMPLIED >
```

6.5 Resümee bzgl. der Koordinaten-Tabellen

Generell scheinen sich Koordinaten-Tabellen am besten für reine Daten-Auflistungen zu eignen. Auch hierarchische Beziehungen der Daten lassen sich sehr elegant beschreiben. Schwierigkeiten könnte die Layout-Umsetzung in einen interaktiven SGML-Editor mit WYSIWYG-Tabellen bereiten, da dieses Konzept erheblich von allen anderen (Editor-internen) Darstellungen abweicht. Da es aber gerade den immer aktueller werdenden Bezug zwischen SGML und Datenbanken sehr gut unterstützt, sind weitere Untersuchungen dieser Tabellennotation sicher lohnenswert.

7 Zusammenfassung und Ausblick

Die Beschreibung von Tabellen in SGML ist prinzipiell kein Problem. SGML ist mächtig genug, die zur eindeutigen Beschreibung einer Tabelle notwendigen Informationen abzubilden. Wesentlich ist allerdings, daß Tabellen zwar Daten strukturiert darstellen, die Tabellen selbst aber eigentlich nur aus Layout-Informationen (Linien, Spaltenbreiten, Ausrichtung der Zellendaten, etc.) bestehen. Ein Tabellen-Layout mit SGML zu beschreiben, widerspricht eigentlich dem Grundgedanken von SGML, Dokumentstrukturen zu bearbeiten.

Da bei Tabellen eigentlich das Layout im Vordergrund steht, gibt es natürlich auch beliebige Variationsmöglichkeit des Layouts. Entsprechend ist eine allgemeingültige Beschreibung immer mit Einschränkungen verbunden. Um für die Betrachtungen eine einheitliche Grundlage bzgl. Begriffen und Anforderungen zu haben, wurde diese im Kapitel 2 festgelegt. Sie bilden die Grundlage für alle weiteren Betrachtungen.

Die verschiedenen, in Kapitel 3 vorgestellten Techniken zur Beschreibung von Tabellen in SGML deuten an, daß es unendlich viele Möglichkeiten gibt. Es zeigte sich aber, daß bestimmte Techniken für die verschiedenen Anforderungen, die an die Tabelle und deren Weiterverarbeitung gestellt werden, gut und weniger gut geeignet sind.

Aufbauend auf diesen Ergebnissen wurden Tabellen in drei Klassen eingeteilt. *Standardtabellen* bieten keinerlei Flexibilität, erlauben aber die einfache Abbildung von Datenbank-Sätzen auf SGML, da die Zelleninhalte mit dedizierten Tags markiert sind. *Freie Tabellen* unterstützen dagegen die variable Gestaltung der Tabelle. Die logischen Zusammenhänge der Daten in den Zellen lassen sich aber nur über das Tabellen-Layout (gleiche Spalten, gleiche Reihen, etc.) erkennen. Eine Zwischenform sind *freie Standardtabellen*. Dort werden die bei der Standardtabelle durch den Tabellentyp fixierten Layout-Eigenschaften wieder mit veränderbaren Attributen versehen. Dadurch wird das Layout in fest vorgegebenen Grenzen wieder variabel.

Da die freien Tabellen die wichtigste Klasse sind, wurden bereits existierende Tabellen-DTDs untersucht und bewertet. Es zeigte sich, daß bei allen Notationen die Abbildung des Tabellen-Layouts auf SGML-Strukturen im Vordergrund stand. Am extremsten zeigt sich das bei der internen ArborText-DTD, die scheinbar programminterne Layout-Informationen direkt in SGML umsetzt. Bei zwei DTDs sind aber auch Ansätze zur logischen Strukturierung der Daten erkennbar, die über die Strukturierung durch Layoutmittel, wie Spalten-/Reihenzugehörigkeit und Zusammenfügen von Zellen hinausgehen. Dies wird in der AAP-DTD und in der ICADD-DTD durch Kontroll-Zellen (engl. *Stub*) erreicht. Diese Zellen stehen am Beginn einer Reihe und kennzeichnen bzw. gruppieren die nachfolgenden Daten. Dieses Konzept wird noch durch Elemente für die verschiedenen Gruppierungshierarchien erweitert. Eine Gesamtübersicht der Eigenschaften der DTDs ist nachfolgend in Tabelle 7–1 zu finden.

Allen SGML-Notationen für freie Tabellen ist gemeinsam, daß die DTDs nicht die Korrektheit des Tabellenaufbaus sicherstellen können. So ist es möglich, daß in der SGML-Instanz die eine Tabellenreihe fünf Spalten und die nächste Reihe nur vier Spalten besitzt. Dieser Fehler wird nicht vom SGML-Parser, sondern erst durch den Formatierer entdeckt, der die Tabellenstruktur und die Attribute semantisch auswertet.

Abschließend wurde ein neuer, vielversprechender Ansatz zur logisch strukturierten Beschreibung von beliebigen Daten-Tabellen vorgestellt. Bei diesem Ansatz wird davon ausgegangen, daß sich wohlstrukturierte Tabellen, die nur Daten beinhalten, auf Koordinatensysteme abbilden lassen. Die Beziehung zwischen Koordinatenachsen und den Koordinatenwerten wurde auf SGML mit einer speziellen DTD für *Koordinaten-Tabellen* abgebildet. Koordinaten-Tabellen unterstützen in einfachster Weise die Datenbanksicht auf die Tabellen. Die Betrachtungsweise, unter der die Tabelle gesehen werden soll, läßt sich sehr einfach durch Austauschen der Koordinatenachsen verändern. Die in SGML abgebildeten hierarchischen und logischen Abhängigkeiten der Daten bestimmen bei diesem Ansatz das Tabellen-Layout und nicht umgekehrt.

Aus den Arbeiten zu diesem Bericht lassen sich zwei Schlußfolgerungen ziehen. Zum einen ist es notwendig, daß Tabellen in einer allgemeineren Form von den SGML-Editoren unterstützt werden, da die Anforderungen an die Flexibilität der Systeme gerade im Datenbank-Bereich ($\rightarrow$ Standardtabellen) sehr hoch sind. Zum anderen muß das Konzept der Koordinaten-Tabellen weiter verfolgt werden, da es einen Ausweg aus der Sackgasse der layoutorientierten Beschreibung von freien Tabellen darstellt. Wichtig scheint hier die Realisierung in einem SGML-Editor, um praktische Erfahrungen sammeln zu können.

Tabelle 7–1 Eigenschaften der Tabellen-DTDs im Überblick

	Addi-son-Wesley	ATA	Arbor-Text	AAP	CALS	DAPHNE	Exo. Compl. Tables	HaL u. O'Reilly	ICADD	ISO	OSF	Soft-Quad	Erläuterung
Tabellenaufbau:													
Tabellenüberschrift	+	+	–*)	+	+	–**)	+	+	+	+	–*)	–*)	*): nicht Teil des Tabellenmodells; **): es gibt nur Tab.-Untertitel
Tabellenkopf	+	+	Kopf-reihen	+	+	Kopf-reihen	+	Kopf-reihen	+	+	Kopf-reihen	+	
Tabellenrumpf	+	+	Reihen	+	+	Reihen	+	Reihen	+	+	Reihen	+	
Tabellenfuß	+	–	–	–*)	+	–	–	–	+	+	–	+	*): nur Anmerkung oder Tab.-Fußnote
Reihe	+	+	+	+	+	+	+	+	+	+	+	+	
Spalte (Deklarationselement)	–	+	–	–	+	–	–	–	+	–	–	+	
Zelle	+	+	+	+	+	+	+	+	+	+	+	+	
Zellenblock	+	–	–	–	+	–	+*)	–	–	–	–	–	*): Tabelle in Zelle
Kontroll-Zelle	–	–	–	+	–	–	–	–	+	–	–	–	
Gebiete	–	–	–	–	–	–	–	–	–	+	–	–	
Spalten:													
Spaltenbreiten	–	+	+	–	+	+*)	+	–	+	+	+	+	*): auf neun Spalten begrenzt
Zusammengefügte Zellen:													
Horizontal	+	+	+	+	+	+*)	+	–	+	+	+	+	*): nur im Tab.-Kopf
Vertikal	+	+	+	+	+	–	+	–	+	+	+*)	+	*): offen, ob gleichzeitig horizontal u. vertikal
Beeinflussung der Tabellenlinien:													
Horizontal	–	+	+	+	+	+	+	–	+	+	+	+	
Vertikal	–	+	+	+	+	+	+	–	+	+	+	+	

	Addi-son-Wesley	ATA	Arbor-Text	AAP	CALS	DAPHNE	Exo. Compl. Tables	HaL u. O'Reilly	ICADD	ISO	OSF	Soft-Quad	Erläuterung
Linienarten	–	–	+*)	+**)	–	–	+***)	–	–	–	–	+**)	*): einfach, doppelt, fett; **): einfach, doppelt, dreifach, gestrichelt, gepunktet, fett, unsichtbar, nicht vorhanden ***): einfach, doppelt, fett, unsichtbar
Individuell für Zelle	–	–	+	–	+	+*)	+	–	+	+	+	+	*): vertikal nur im Tab.-Kopf
Attribut (A) oder Element (E)	A	A	E	A	A	vert. A, horiz. E	A	–	A	A	A	A	
Ausrichtung des Zelleninhaltes:													
Horizontal	–	+	+	+	+	+	+	–	+	+	+	+	
Horizontal nach Zeichen	–	–	+	–	+	–	+*)	–	+	+	–	+	*): nur nach Dezimalpunkt
Vertikal	–	–	+	+	+	+*)	+	–	+	+	+	+	*): nur im Tab.-Rumpf
Für Tabelle	–	–	+	spalten-weise	horiz.	spalten-weise	+	–	–	+	spalten-weise	–	
Für Kopf/Rumpf/Fuß	–	–	–	–	vertikal	–	+	–	–	+	–	–	
Für Reihe	–	–	vertikal	vertikal	–	vertikal*)	+	–	vertikal	+	spalten-weise	vertikal	*): nur im Tab.-Rumpf
Für Spalte	–	horiz.	–	–	horiz.	–	–	–	horiz.	–	–	horiz.	
Individuell für Zelle	–	–	+	+	+	horiz.*)	+**)	–	+	+	+	+	*): nur im Tab.-Kopf **): Angabe bei Tabelle, Kopf/Rumpf oder Reihe

	Addi-son-Wesley	ATA	Arbor-Text	AAP	CALS	DAPHNE	Exo. Compl. Tables	HaL u. O'Reilly	ICADD	ISO	OSF	Soft-Quad	Erläuterung
Zelleneigenschaften:													
Hintergrund	–	–	+*)	–	–	–	–	–	–	+**)	–	–	*): fünf Schattierungsstufen; **): indirekt durch Gebiet
Diagonal aufsplitten	–	–	–	+	–	–	–	–	–	–	–	–	
Rotation des Inhaltes	–	–	–	–	+	–	–	–	–	+	–	–	
Schriftgröße	–	–	+	–	–	–	–	–	–	+	–	–	
Abstände (oben, unten, rechts, links) zu Linien	–	–	+*)	–	–	–	+*)	–	–	–	–	–	*): gleichzeitig für alle Zellen der Tabelle
Datenbank-Info-Attribut	–	–	–	–	–	–	–	–	–	+*)	–	–	*): indirekt durch Gebiet
Bemerkungen:													
CALS: Die Bedeutung des Attributes tgroupstyle (<tgroup>, <entrybl>) ist unbekannt. Es kann unter Umständen Auswirkungen auf die Zelleneigenschaften haben.													
HaL u. O'Reilly: Die Spezifikationsmöglichkeiten beim Element <TableSpec> sind unbekannt. Die Tab.-Notation kann u.U. mächtiger sein.													
OSF: Mit dem Attribut colweight (<table>) können einzelne Spalten je nach Bedeutung gewichtet werden.													
SoftQuad: Es gibt kein Element, das die Gesamt-Tabelle kennzeichnet. Tab.-Kopf, Rumpf und Fuß werden einzeln markiert.													

8 Literatur

[1] Air Transport Association of America: *Specification for Manufacturers' Technical Data — A.T.A. Specification No. 100*, Air Transport Association of America, Washington (D.C.), 1991.

[2] ArbotText Inc.: *Publisher — User Manual*, Ann Arbor (Michigan, U.S.A.).

[3] Association of American Publishers: *Electronic Manuscript Series — Markup of Tabular Material*, Version 2.0 Revised Edition, Electronic Publishing Special Interest Group (EPSIG), Dublin (Ohio), 1989, ISBN 1-55653-082-X (series).

[4] Bryan, M.: *SGML — An Author's Guide to the Standard Generalized Markup Language*, Addison-Wesley, 1988, ISBN 0-201-17535-5.

[5] Deutsches Forschungsnetz: *DAPHNE — Document Application Processing in a Heterogeneous Network Environment — User Instruction Manual*, Deutsches Forschungsnetz (DFN), DFN-Bericht Nr. 60, Berlin, 1989.

[6] Deutsches Institut für Normung (Hrsg.): *Informationsverarbeitung — Textverarbeitung und -kommunikation — Genormte Verallgemeinerte Auszeichnungssprache (SGML)*, DIN EN 28879:1991, Beuth-Verlag, 1991.

[7] Deutsches Institut für Normung (Hrsg.): *Elektronischer Datenaustausch für Verwaltung, Wirtschaft und Transport (EDIFACT) — Syntax-Regeln auf Anwendungsebene*, DIN EN 29735:1992, Beuth-Verlag, 1992.

[8] Deutsches Institut für Normung (Hrsg.): *Elektronisches Publizieren in der Fachinformation — Rechnergestützte Dokumentbearbeitung von Normen*, DIN V 33900:1992 Teile 1–3 und Beiblatt 1, Beuth Verlag, 1992.

[9] Exoterica Corporation: *Exoterica Complex Tables*, Bericht Nr. EUM09-0393 der Exoterica Corporation, Ottawa (Kanada), 1993.

[10] Goldfarb, C.F.; Mosher, E.J.; Peterson, T.I.: *An Online System for Integrated Text Processing*, In: Proceedings of the American Society for Information Science, 7, S. 147–150, 1970.

[11] Goldfarb, C.F.: *The SGML Handbook*, Oxford University Press, 1990, ISBN 0-19-853737-9.

[12] Häfemeier, F.; Meißner, E.: *Elektronische Speicherung und Handhabung technischer Dokumente — Arbeitspunkt 5: Transportschale*, Bundesministerium für Wirtschaft, Verbundprojekt Nr. 68504, 1992/93.

[13] HaL Computer Systems International and O'Reilly & Associates: *DocBook DTD*, O'Reilly & Associates, Inc., Sebastopol (California), 1992.

[14] Hofmeyer, J.; Rath, H.H.; Wiedling, H.-P.: *Elektronische Speicherung und Handhabung technischer Dokumente — Arbeitspunkt 4: Vergleich FOSI – DSSSL*, Bundesministerium für Wirtschaft, Verbundprojekt Nr. 68504, 1992/93.

[15] IBM: *Document Composition Facility: Generalized Markup Language Starter Set Reference*, IBM Forum SH20-9187-3, Colorado (U.S.A.), 1985.

[16] International Organization for Standardization: *Information processing systems — Open Systems Interconnection — Basic Reference Model*, ISO 7498:1984, ISO, 1984.

[17] International Organization for Standardization: *Information processing — Text and office systems — Standard Generalized Markup Language (SGML)*, ISO 8879:1986, ISO, 1986.

[18] International Organization for Standardization: *Information processing — SGML support facilities — SGML Document Interchange Format (SDIF)*, ISO 9069:1988, ISO, 1988.

[19] International Organization for Standardization: *Electronic data interchange for administration, commerce and transport (EDIFACT) — Application level syntax rules*, ISO 9735:1988, ISO, 1988.

[20] International Organization for Standardization: *Information processing — Text and office systems — Office Document Architecture (ODA) and interchange format*, ISO 8613:1989 Teile 1, 2, 4–8, 10, ISO, 1989.

[21] International Organization for Standardization: *Information technology — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1)*, ISO/IEC 8824:1990, ISO/IEC, 1990.

[22] International Organization for Standardization: *Information technology — Open Systems Interconnection — Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*, ISO/IEC 8825:1990, ISO/IEC, 1990.

[23] International Organization for Standardization: *Information technology — ISO 7-bit coded character set for information interchange*, ISO/IEC 646:1991, ISO/IEC, 1991.

[24] International Organization for Standardization: *Information technology — Text and office systems — Document Style Semantics and Specification Language (DSSSL)*, ISO/IEC DIS 10179:1991, ISO/IEC, 1991.

[25] International Organization for Standardization: *Information processing — SGML support facilities — Techniques for using SGML — Part 11: Application at ISO Central Secretariat for International Standards and Technical Reports*, ISO/IEC TR 9573-11:1992, ISO/IEC, 1992.

[26] International Organization for Standardization: *Information technology — Hypermedia/Time-based Structuring Language (HyTime)*, ISO/IEC 10744:1992, ISO/IEC, 1992.

[27] International Organization for Standardization: *Electronic manuscript preparation and markup*, ISO CD 12083:1993, ISO, 1993.

[28] Knuth, D.E.: *The TeXbook*, Addison-Wesley, Massachusetts, 1984.

[29] Lamport, L.: *LaTeX: A Document Preparation System*, Addison-Wesley, Massachusetts, 1986.

[30] Mason, T.; Brown, D.: *lex & yacc*, O'Reilly & Associates, Sebastopol (California), 1990, ISBN 0-937175-49-8.

[31] National Information Standards Organization: *Electronic Manuscript Preparation and Markup*, ANSI/NISO Z39.59-1988, Transaction Publishers, New Brunswick (New York), 1991, ISBN 0-88738-945-7.

[32] Open Software Foundation: *The OSF Document Type Definitions User's Guide*, Open Software Foundation, Inc., Cambridge, (Massachusetts), 1993.

[33] Rubinsky, Yuri: *Example document using DTD for the print disabled*, In: Proceedings GCA (Graphics Communication Association) SGML Europe 93, Amsterdam, 1993.

[34] Salomaa, A.K.: *Formale Sprachen*, Übersetzt aus dem Englischen von E.-W. Dieterich, Studienreihe Informatik, W. Brauner, G. Goos (Hrsg.), Springer-Verlag, 1978.

[35] Schirmer, C.: *troff-Programmierung*, Carl Hanser Verlag, München, Wien, 1990, ISBN 3-446-15649-6.

[36] Sklar, D.: *A Nomenclature for Describing Table Markup Strategies*, In: Proceedings GCA (Graphics Communication Association) SGML Europe 93, Amsterdam, 1993.

[37] SoftQuad Inc.: *Author/Editor — User Manual*, Toronto (Canada).

[38] STEP Stürtz Electronic Publishing GmbH: *Tabellen in SGML — ein Vergleich der CALS und ISO Dokumenttypdefinitionen*, Studie im Auftrag des Beuth-Verlages GmbH, Würzburg/Berlin, 1992.

[39] Thomas, R.E.: *SGML in Practice — The PHIGS Slide Set*, Rutherford Appleton Laboratory, Informatics Department, interner Bericht RAL-93-029, Chilton (DIDCOT, Oxon, U.K.), 1992.

[40] Thomas, R.E.: *SGML Tables for the PHIGS Slide Set*, Rutherford Appleton Laboratory, Informatics Department, interner Bericht RAL-93-029, Chilton (DIDCOT, Oxon, U.K.), 1993.

[41] Tobin, Chr.; Wesley, T.: *ISO & the print disabled*, In: Proceedings GCA (Graphics Communication Association) SGML Europe 93, Amsterdam, 1993.

[42] U.S. Department of Defense: *Markup requirements and generic style specification for electronic printed output and exchange of text*, MIL-M-28001A, U.S. Department of Defense, 1990.

[43] U.S. Department of Defense: *Markup requirements and generic style specification for electronic printed output and exchange of text — Appendix B: Output specification*, MIL-M-28001A, U.S. Department of Defense, 1990.

[44] U.S. Department of Defense: *Manuals, Technical: General Style and Format Requirements*, MIL-M-38784C, U.S. Department of Defense, 1990.

[45] U.S. Department of Defense: *Automated interchange of technical information*, MIL-STD-1840B, U.S. Department of Defense, 1992.

[46] Van Herwijnen, E.: *Practical SGML*, U.S.A., 1990, ISBN 0-7923-0635-X.

[47] Van Vooren, L.: *Background for a Generalized Semantic Attachment for Tables*, Vortrag während GCA (Graphics Communication Association) Konferenz SGML Europe 93, Amsterdam, 1993.

Anhang A DTDs für freie Tabellen

Das Inhaltsmodell der Zellen wird in jeder DTD durch das eine fiktive Entity **%m.cell;** repräsentiert, da die unterschiedlichen Zelleninhalte für diesen Vergleich nicht relevant sind. Die fiktiven Entities **%m.outline;** und **%m.inline;** repräsentieren die Inhaltsmodelle der DTDs für *Outline*-Elemente (wie Absatz, Aufzählung, etc.) bzw. *Inline*-Elemente (wie einfacher Text (#PCDATA), Querverweise, Hervorhebungen, etc.).

Anmerkung: Trotz mehrmaligen Korrekturlesens kann nicht ausgeschlossen werden, daß sich in die nachfolgenden DTD-Fragmente der eine oder andere Schreibfehler eingeschlichen hat. Aus diesem Grund übernimmt der Autor keine Garantie bzgl. der 100%-igen Richtigkeit der DTDs und haftet entsprechend nicht für Schäden, die sich direkt oder indirekt aus der Verwendung dieser Fragmente ergeben.

Anmerkung: Die Kopier- und Nutzungsrechte der angegebenen DTD-Fragmente liegen, selbst wenn nicht explizit angegeben, bei den Herausgebern der DTDs.

A.1 Addison-Wesley

TextBook-DTD

```
<!ELEMENT table    – – (nt?, ht?, hc?, bt+, ft?) >
<!ATTLIST table    id       ID       #IMPLIED
                   cols     NUMBER   #REQUIRED
                   tabs     NUMBERS  ”1 10 20 30 40 50 60 70” >

<!ELEMENT nt       – O %m.inline;
                   –– number or other identifier of table ––>

<!ELEMENT ht       – O %m.outline;
                   –– heading of table ––>

<!ELEMENT hc       – O (r)+ –(bt,ft)
                   –– heading of columns ––>
<!ATLLIST (hc|bt|ft)
                   cols     NUMBER   #IMPLIED >

<!ELEMENT bt       – O (r)+
                   –– body of table ––>

<!ELEMENT ft       – O (r)+ –(hc,bt)
                   –– foot of table ––>

<!ELEMENT r        O O (c|hc|bt|ft)+
                   –– row in table ––>

<!ELEMENT c        – O %m.cell;
                   –– cell of table ––>
<!ATTLIST c        straddle NUMBER   ”1” >
```

A.2 Air Transport Association (ATA) of America

DTD für Aircraft Maintenance Manual

```
<!ELEMENT ctable – – (title?,coldef+,thead?,tbody)>
<!ATTLIST ctable
  id        CDATA     #IMPLIED
  totalcol  NUMBER    #REQUIRED
  rowsep    NUMBER    ”1”
  colsep    NUMBER    ”1” >

<!ELEMENT coldef – O EMPTY >
<!ATTLIST coldef
  colnum    NUMBER    #REQUIRED
  width     CDATA     #IMPLIED
  posn      (l|r|c)   ”l”
  colsep    NUMBER    #IMPLIED >
```

```
<!ELEMENT thead – O (ctabrow+) >

<!ELEMENT tbody – – (ctabrow+) >

<!ELEMENT ctabrow – – (cell+) >
<!ATTLIST ctabrow
  rowsep    NUMBER    ”1” >

<!ELEMENT cell – O %m.cell; >
<!ATTLIST cell
  colspan   NUMBER    ”1”
  rowspan   NUMBER    ”1” >
```

A.3 ArborText Publisher

Interne DTD

```
<!ELEMENT table         – – (rowrule,(tablerow,rowrule)+)>
<!ATTLIST table         acl             CDATA   #IMPLIED
                        chj             CDATA   #IMPLIED
                        csl             CDATA   #IMPLIED
                        cst             CDATA   #IMPLIED
                        ctl             CDATA   #IMPLIED
                        cwl             CDATA   #REQUIRED
                        hff             CDATA   #IMPLIED
                        hfs             CDATA   #IMPLIED
                        htm             CDATA   #IMPLIED
                        hts             CDATA   #IMPLIED
                        jst             CDATA   #IMPLIED
                        ncols           CDATA   #IMPLIED
                        off             CDATA   #IMPLIED
                        rth             CDATA   #IMPLIED
                        rst             CDATA   #IMPLIED
                        rvj             CDATA   #IMPLIED
                        tff             CDATA   #IMPLIED
                        tfs             CDATA   #IMPLIED
                        tts             CDATA   #IMPLIED
                        unt             CDATA   #IMPLIED
                        wdm             CDATA   #REQUIRED
                        ctmarg          CDATA   #IMPLIED
                        cbmarg          CDATA   #IMPLIED
                        clmarg          CDATA   #IMPLIED
                        crmarg          CDATA   #IMPLIED
                        dispwid         CDATA   #IMPLIED>

<!––
Argument values in capital letters generally represent constructs used
by one or more arguments, and are listed below in alphabetical order.

COLTYPE is one of the following single characters:
        b       for both right and left justified,
        l       for flush left,
        r       for flush right,
        c       for centered,
        d       for decimal/character alignment, or
        *       for the default.

DIMEN is a number followed immediately by a unit, such as ”12pt” or ”3.14in”.
        Accepted units are ”in”, ”pt”, ”cm”, ”mm”, and ”pc” (for pica).  Never
        insert a space between the number and its unit, since spaces separate
        argument=”value” pairs.

FONTFAM is one of the following three–character codes:
        TIM     for Times Roman,
        HLV     for Helvetica, or
        COU     for Courier.
```

```
ROWTYPE is one of the following single characters:
        t       for top justified,
        c       for centered, or
        b       for bottom justified.

RULETYPE is one of the following single characters:
        .       for blank rules,
        –       for single rules,
        =       for double rules,
        +       for bold rules, or
        *       for the default.

TYPESIZE is one of the following available type sizes in points: 7, 8, 9, 10,
        11, 12, 14, or 18.  Note that no ”pt” is necessary.

TYPESTY is one of the following two–character codes:
        rm      for roman,
        it      for italic,
        bf      for boldface, or
        bi      for bold italic.

Immediately after the <rowrule> tag, one of the following processing
instructions can be inserted to determine whether to force or inhibit a page
break after that row:
        <?Pub Lcl breakpenalty=”10000”>         inhibit
        <?Pub Lcl breakpenalty=”–10000”>        force

Required attributes:
====================

        wdm=VAL         where VAL is either ”abs” or a whole number between 1
                        and 100.  Use ”abs” to create a table with absolute
                        width mode.  Use the number to indicate a relative
                        width table, with the number representing the width of
                        the table as a percent of the page or text width.

        cwl=LIST        where LIST is a set of column widths separated by a
                        colon.  The format of the width values depends on the
                        value of the ”wdm” tag described above.
                        If wdm=abs, the widths are dimens.
                        Example:  cwl=”45pt:2in:2.5in” for a three column
                        table.
                        if wdm=NUMBER, the numbers represent the relative width
                        of each column as well as their width in pixels as they
                        appear on the screen.  For example, cwl=”120:300:220”
                        is more reasonable than cwl=”6:15:11”, even though the
                        ratio is the same, since the latter example creates
                        very skinny columns on the screen – the whole table is
                        only 0.4in (6+15+11=32 pixels at 80dpi) wide.

Optional attributes:
====================

        acl=LIST        Alignment character list.  Used in conjuction with
                        decimal/character alignment to change the alignment
                        character.  The list consists of the alignment
                        character for each column separated by a colon.  For
                        decimal alignment, something like acl=”.:.:.:.” is
                        reasonable.

        chj=COLTYPE     Cell horizontal justification.

        csl=LIST        List of specific vertical rule types.  For an n–column
                        table, needs n+1 RULETYPE values separated by a colon.

        cst=RULETYPE    Vertical rule type for the table.

        ctl=LIST        List of specific horizontal justifications for each
                        column in the table.

        cwl=LIST        See ”required arguments” above.
```

```
        hff=FONTFAM     The font family used in columnheads.

        hfs=TYPESIZE    The type size used in columnheads.

        htm=VAL         Height mode.  VAL is ”abs” for absolute mode, or ”flx”
                        for flexible height mode.  Absolute height mode puts
                        up the vertical ruler.

        hts=TYPESTY     The type style used in columnheads.

        jst=VAL         Document table justification.  VAL is either ”L”, ”R”,
                        or ”C”, depending on whether the table is to be left,
                        right or centered on the page.

        ncols=VAL       The number of columns in the table.  This value should
                        agree with the number of columns expressed by the
                        required cwl attribute.

        off=DIMEN       If jst=”L”, the offset from the left margin to the
                        left side of the table.

        rth=VAL         Rule thickness in 300dpi pixels.  VAL should be a whole
                        number.

        rst=RULETYPE    Horizontal rule type for the table.

        rvj=ROWTYPE     Row vertical justification for the table.

        tff=FONTFAM     Font family for cells of the table.

        tfs=TYPESIZE    Font size for cells of the table.

        tts=TYPESTY     Type style for cells of the table.

        unt=VAL         Unit for the rulers.  VAL can be either ”in” or ”cm”.

        wdm=VAL         See ”required arguments” above.
-->

<!ELEMENT tablerow      - O (cellrule,(tablecell,cellrule)+)>
<!ATTLIST tablerow      hdr             CDATA   #IMPLIED
                        rht             CDATA   #IMPLIED
                        rvj             CDATA   #IMPLIED>

<!--
        hdr=VAL         Used to indicate a columnhead.  Only applies to first
                        row.  VAL is 1 for enabled, 0 for disabled.

        rht=DIMEN       Row height.

        rvj=ROWTYPE     Vertical justification for text in the row.
-->

<!ELEMENT tablecell     - - %m.cell;>
<!ATTLIST tablecell     cff             CDATA   #IMPLIED
                        cfs             CDATA   #IMPLIED
                        chj             CDATA   #IMPLIED
                        cts             CDATA   #IMPLIED
                        cvj             CDATA   #IMPLIED
                        shd             CDATA   #IMPLIED
                        spn             CDATA   #IMPLIED
                        vspn            CDATA   #IMPLIED>

<!--
        cff=FONTFAM     Font family for that cell.
                        Default: inherit from <table> tag

        cfs=TYPESIZE    Font size for that cell.
                        Default: inherit from <table> tag

        chj=COLTYPE     Horizontal justification for that cell.
                        Default: left justified

        cts=TYPESTY     Type style for that cell.
                        Default: inherit from <table> tag

        cvj=ROWTYPE     Vertical justification for that cell.
                        Default: top justified
```

```
        spn=VAL         For horizontally spanned columns.  VAL is a whole
                        number representing how many columns are spanned.
                        Note that for horizontal spans, the text appears
                        in the LEFTMOST cell in the span, and all other
                        cells in the span should be void of text.
                        Default: 1

        vspn=VAL        For vertically spanned rows.  VAL is a whole number
                        representing how many rows are spanned.  Note that
                        for vertical spans, the text appears in the LOWEST
                        cell in the span, and all other cells in the span
                        should be void of text.
                        Default: 1

        shd=VAL         Cell shading.  VAL is a number 0..5, where 0 means
                        no shading, and 5 means darkest shading.
                        Default: no shading
-->

<!ELEMENT rowrule       - O EMPTY>
<!ATTLIST rowrule       rty             CDATA   #IMPLIED
                        rtl             CDATA   #IMPLIED>

<!--
        rtl=LIST        For an n-column table, a list of n+1 RULETYPEs
                        separated by a colon, to represent each segment in
                        that rule.

        rty=RULETYPE    Rule type for the entire rule.
-->

<!ELEMENT cellrule      - O EMPTY>
<!ATTLIST cellrule      rty             CDATA   ".">

<!--
        rty=RULETYPE    Rule type for that segment.
-->
```

A.4 Association of American Publishers (AAP)

DTD für komplexe Tabellen

```
<!ENTITY % t.align "align (l|c|r|d|e|en) #IMPLIED"
         --align attribute for columns, cells-->

<!ENTITY % t.val "valign (t|m|b) #IMPLIED"
         --vertical alignment-->

<!ENTITY % t.split "split (y|n) n"
         --will split cell diagonally, default no-->

<!ELEMENT ctbl - - (cthd, ctby, ctbf) >
<!ATTLIST ctbl id  ID       #IMPLIED
               cs  NMTOKENS "A bl"
               rs  NMTOKENS "A bl"
               ca  NMTOKENS "A c" >

<!ELEMENT cthd - O (no|ctt|ctst|cthn|cthr)* >

<!ELEMENT cthn - O %m.outline; -(cthnr) >
<!ATTLIST cthn id  ID       #IMPLIED >

<!ELEMENT (ctt|ctst) - O %m.inline; >

<!ELEMENT cthr - O (cth*|ctsh*) >
<!ATTLIST cthr ra  NMTOKENS "A m" >

<!ELEMENT (cth|ctsh)
               - O %m.cell; >
<!ATTLIST (cth|ctsh)
               %t.align;
```

```
                %t.val;
                %t.split;
                cb  NUMBER   #IMPLIED
                ce  NUMBER   #IMPLIED
                rb  NUMBER   #IMPLIED
                re  NUMBER   #IMPLIED >

<!ELEMENT ctby – O (ctr)* >

<!ELEMENT ctr  – O ((ctsb1?,cte)*|(ctsb2?,cte)*|
                    (ctsb3?,cte)*|(ctsb4?,cte)*|ctsh*) >
<!ATTLIST ctr  ra  NMTOKENS ”A m” >

<!ELEMENT (ctsb1|ctsb2|ctsb3|ctsb4)
                – O %m.cell; >
<!ATTLIST (ctsb1|ctsb2|ctsb3|ctsb4)
                %t.align;
                %t.val; >

<!ELEMENT cte  – O %m.cell; >
<!ATTLIST cte  %t.align;
                %t.val;
                %t.split;
                cb  NUMBER   #IMPLIED
                ce  NUMBER   #IMPLIED
                rb  NUMBER   #IMPLIED
                re  NUMBER   #IMPLIED >

<!ELEMENT ctbf – O (ctc|ctfn) >

<!ELEMENT ctc  – O %m.outline; >

<!ELEMENT ctfn – O %m.outline; –(ctfnr) >
<!ATTLIST ctfn id  ID       #IMPLIED >

<!ELEMENT (cthnr|ctfnr)
                – O %m.cell; –(cthnr|ctfnr) >
<!ATTLIST (cthnr|ctfnr)
                rid IDREF    #IMPLIED >
```

A.5 Computer Aided Acquisition and Logistics Support (CALS)

Doc-DTD für MIL–M–38784 Dokumente (MIL–M–28001)

```
<!ENTITY % yesorno ”NUMBER”                                          >

<!ELEMENT table    – – (title, tgroup+) –(table)                     >
<!ATTLIST table         tabstyle    NMTOKEN            #IMPLIED
                        tocentry    %yesorno;          ”1”
                        shortentry  %yesorno;          #IMPLIED
                        frame       (top | bottom |
                                     topbot | all |
                                     sides | none)     #IMPLIED
                        colsep      %yesorno;          #IMPLIED
                        rowsep      %yesorno;          #IMPLIED
                        orient      (port | land)      #IMPLIED
                        pgwide      %yesorno;          #IMPLIED
                        %bodyatt;  -- a lot of useful (?) attributes -->

<!ELEMENT tgroup   – o (colspec*, spanspec*, thead?, tfoot?, tbody)  >
<!ATTLIST tgroup        cols            NUMBER         #REQUIRED
                        tgroupstyle     NMTOKEN        #IMPLIED
                        colsep          %yesorno;      #IMPLIED
                        rowsep          %yesorno;      #IMPLIED
                        align     (left | right | center | justify | char )
                                                       ”left”
                        charoff         NUTOKEN        ”50”
                        char            CDATA          ””           >
```

```
<!ELEMENT colspec  – o EMPTY                                          >
<!ATTLIST colspec       colnum          NUMBER          #IMPLIED
                        colname         NMTOKEN         #IMPLIED
                        align     (left | right | center | justify | char )
                                                        #IMPLIED
                        charoff         NUTOKEN         #IMPLIED
                        char            CDATA           #IMPLIED
                        colwidth        CDATA           #IMPLIED
                        colsep          %yesorno;       #IMPLIED
                        rowsep          %yesorno;       #IMPLIED        >

<!ELEMENT spanspec  – o EMPTY                                         >
<!ATTLIST spanspec      namest          NMTOKEN         #REQUIRED
                        nameend         NMTOKEN         #REQUIRED
                        spanname        NMTOKEN         #REQUIRED
                        align     (left | right | center | justify | char )
                                                        ”center”
                        charoff         NUTOKEN         #IMPLIED
                        char            CDATA           #IMPLIED
                        colsep          %yesorno;       #IMPLIED
                        rowsep          %yesorno;       #IMPLIED        >

<!ELEMENT (thead | tfoot)  – o (colspec*, row+)
                           –(entrytbl)                                >
<!ATTLIST thead         valign          (top | middle | bottom)
                                                        ”bottom”        >
<!ATTLIST tfoot         valign          (top | middle | bottom)
                                                        ”top”           >

<!ELEMENT tbody  – o (row+)                                           >
<!ATTLIST tbody         valign          (top | middle | bottom)
                                                        ”top”           >

<!ELEMENT row  – o (entry | entrytbl)+                                >
<!ATTLIST row           rowsep          %yesorno;       #IMPLIED        >

<!ELEMENT entry     – o %m.cell;                                      >
<!ATTLIST entry         colname         NMTOKEN         #IMPLIED
                        spanname        NMTOKEN         #IMPLIED
                        namest          NMTOKEN         #IMPLIED
                        nameend         NMTOKEN         #IMPLIED
                        morerows        NUMBER          ”0”
                        colsep          %yesorno;       #IMPLIED
                        rowsep          %yesorno;       #IMPLIED
                        rotate          %yesorno;       ”0”
                        valign          (top | bottom | middle)
                                                        ”top”
                        align   (left | right | center | justify | char )
                                                        #IMPLIED
                        charoff         NUTOKEN         #IMPLIED
                        char            CDATA           #IMPLIED        >

<!ELEMENT entrytbl  – – (colspec*, spanspec*, thead?, tbody)+
                        –(entrytbl)                                   >
<!ATTLIST entrytbl      cols            NUMBER          #REQUIRED
                        tgroupstyle     NMTOKEN         #IMPLIED
                        colname         NMTOKEN         #IMPLIED
                        spanname        NMTOKEN         #IMPLIED
                        colsep          %yesorno;       #IMPLIED
                        rowsep          %yesorno;       #IMPLIED
                        align   (left | right | center | justify | char )
                                                        #IMPLIED
                        charoff         NUTOKEN         #IMPLIED
                        char            CDATA           #IMPLIED        >
```

A.6 Deutsches Forschungsnetz (DFN)

DAPHNE Report-DTD

```
<!ELEMENT  TABLE    – O ((TABHEAD|RULE)*,ROW,(ROW|RULE)*,SUBTITLE?) >
<!ATTLIST  TABLE
    POSITION (CENTER|LEFT)     CENTER
    BOUND    (KEEP|SPLIT)      KEEP
    SCHEMA   CDATA             "|c|c|"
    ID       ID                #IMPLIED
    C1       CDATA             ""
    C2       CDATA             ""
    C3       CDATA             ""
    C4       CDATA             ""
    C5       CDATA             ""
    C6       CDATA             ""
    C7       CDATA             ""
    C8       CDATA             ""
    C9       CDATA             ""
    CALL     CDATA             "*"
    NEWPAGE (NEWPAGE|NONEWPAG) NONEWPAG >

<!ELEMENT  TABHEAD  – O (COLHEAD+) >

<!ELEMENT  COLHEAD  – O %m.cell; >
<!ATTLIST  COLHEAD
    SCHEMA   CDATA      "|c|"
    COLUMNS  CDATA      "1"   >

<!ELEMENT  ROW      – O (COL+) >
<!ATTLIST  ROW
    VALIGN  (T|C|B)     T >

<!ELEMENT  COL      – O (CLINE+) >

<!ELEMENT  CLINE    O O %m.cell; >

<!ELEMENT  RULE     – O EMPTY >

<!ELEMENT  SUBTITLE – O %m.inline; >
<!ATTLIST  SUBTITLE
    ID     ID           #IMPLIED  >
```

A.7 Exoterica Corporation

Exoterica Complex Tables DTD

```
<!-- Copyright (C) 1988–1991 by Exoterica Corporation -->

<!ENTITY % t.tbl     "tbl">

<!ENTITY % t.head    "t.head">

<!ENTITY % t.body    "t.body">

<!ENTITY % t.row     "t.row">

<!ENTITY % t.col     "t.col">

<!ENTITY % t.title   "t.title">

<!ENTITY % t.sec     "%t.title;">

<!ENTITY % t.text    "%m.outline;">

<!ENTITY % t.cont    "(%m.cell; | tbl)*">

<!ENTITY % t.tblat   "">

<!ENTITY % t.rowat   "">

<!ENTITY % t.rsep    "s | d | k | b">
```

```
<!--
<!ENTITY % t.simpl  "INCLUDE">
-->
<!ENTITY % t.simpl  "IGNORE">
<!ELEMENT %t.tbl;     - - ((%t.sec;) | ((%t.head;)?, %t.body;)))+>
<!ATTLIST %t.tbl;
          wd      NUMBER    #IMPLIED
          cw      NUMBERS   #REQUIRED
          gu      NUMBER    #IMPLIED
          cs      NMTOKENS  #IMPLIED
          rs      NMTOKENS  #IMPLIED
          box     NMTOKENS  #IMPLIED
          ha      NMTOKENS  #IMPLIED
          va      NMTOKENS  #IMPLIED
          irs     NUMBER    #IMPLIED
          %t.tblat;
          %t.rowat;>
<!ELEMENT %t.head;    - O (%t.row;)+>
<!ATTLIST %t.head;
          cs      NMTOKENS  #IMPLIED
          rs      NMTOKENS  #IMPLIED
          ha      NMTOKENS  #IMPLIED
          va      NMTOKENS  #IMPLIED
          irs     NUMBER    #IMPLIED
          %t.rowat;>
<!ELEMENT %t.body;    - O (%t.row;)+>
<!ATTLIST %t.body;
          cs      NMTOKENS  #IMPLIED
          rs      NMTOKENS  #IMPLIED
          ha      NMTOKENS  #IMPLIED
          va      NMTOKENS  #IMPLIED
          irs     NUMBER    #IMPLIED
          %t.rowat;>
<!ELEMENT %t.row;     O O (%t.col;)+>
<!ATTLIST %t.row;
          cs      NMTOKENS  #IMPLIED
          crs     %t.rsep;  #IMPLIED
          ha      NMTOKENS  #IMPLIED
          va      NMTOKENS  #IMPLIED
          irs     NUMBER    #IMPLIED
          %t.rowat;>
<![%t.simpl;  -- INCLUDEd if simple tables are requested --  [
<!ELEMENT %t.col;     O O (%t.cont;) -(%t.tbl;)>
<!ENTITY % t.cmplx  "IGNORE">
]]>
<!ENTITY % t.cmplx  "INCLUDE">
<![%t.cmplx;  -- IGNOREd for simple tables --  [
<!ELEMENT %t.col;     O O (%t.cont;)>
]]>
<!ATTLIST %t.col;
          span    NUMBER    #IMPLIED>
<!ELEMENT %t.title;   - O (%t.text;)>
```

A.8 HaL Computer Systems, Inc. and O'Reilly and Associates, Inc.

DocBook-DTD

```
<!ENTITY % tablecontent.gp "(TableSpec?, TableColHead*, TableRow+)">
<!ELEMENT Table         - -  (Title?, TitleAbbrev?, Graphic?, (%tablecontent.gp;))>
<!ATTLIST Table         id       ID  #IMPLIED>
```

```
<!ELEMENT Title         – –  ((%m.inline;)+)>
<!ATTLIST Title         id       ID #IMPLIED
                        Pagenum   CDATA  #IMPLIED>

<!ELEMENT TitleAbbrev   – –  ((%m.inline;)+)>

<!ELEMENT Graphic       – –  RCDATA>
<!ATTLIST Graphic id    ID #IMPLIED
                        Type    (GIF|TIFF|DVI|EPS|tbl|Tex|bitmap) GIF >

<!ELEMENT TableSpec     – –  %m.inline;>

<!ELEMENT TableColHead  – –  (TableCell+)>

<!ELEMENT TableCell     – –  %m.cell;>

<!ELEMENT TableRow      – –  (TableCell+)>
```

A.9 International Committee on Accessible Document Design (ICADD)

DTD for Braille, Large Print and Computer-Voice

```
<!ENTITY % cellmdl     "(#PCDATA)"                    >

<!ENTITY % headmdl     "(#PCDATA)"                    >

<!ENTITY % vborder
        "(vyes | vno)"  -- useful in Braille and large print -- >

<!ENTITY % hborder
        "(hyes | hno)"  -- useful in Braille and large print -- >

<!ENTITY % valign  "(top|middle|bottom)"
                           -- useful in large print only -- >

<!ENTITY % halign  "(left|both|center|right|char)"
                           -- useful in large print -- >

<!ENTITY % tblunit      "(percent|pixels|points|picas|mm|cm|inches)"
                           -- useful in large print only -->

<!ENTITY % rprops
        'rowsep         %hborder;  #IMPLIED
         valign         %valign;   #IMPLIED'    >

<!ENTITY % cprops
        'colsep         %vborder;       #IMPLIED
         halign         %halign;        #IMPLIED
         alignchr       CDATA           #IMPLIED
         charpos        CDATA           #IMPLIED'
         -- useful in large print only      --  >

<!-- TABLE is the top-level table element. H3 is the Braille
     heading level for a table title. [Non ICADD DTDs may wish
     to replace H3 with TABLETITLE or equivalent.]          -->
<!ELEMENT table    - -    (h3?, note?, tgroup)  >
<!ATTLIST table
        frame           (box | nobox)   nobox
        tblwd           CDATA           #IMPLIED
        tblht           CDATA           #IMPLIED
        tblunits        %tblunit;       #IMPLIED
        -- the dimensions will be meaningless for the Braille and
        computer voice versions but may be useful for large print.
        They cover the entire table unless values are assigned
        on the TGROUP, THEAD, TBODY and TFOOT elements.
           The TGROUP model allows TFOOT either before or after
        the TBODY to support multi-page tables whose footers need
        to be available for the first page during formatting.   -- >

<!ELEMENT tgroup    o o    (coldef+, thead?,
                                ((tfoot?, tbody+) | (tbody+, tfoot?))) >
```

```
<!ELEMENT thead     – –    (coldef+, hdrow+)  >

<!ELEMENT tbody     – –    (coldef+, row+)    >

<!ELEMENT tfoot     – –    (xref?, ftn)+      >
<!ATTLIST (tgroup | thead | tbody | tfoot)
        tblwd           CDATA           #IMPLIED
        tblht           CDATA           #IMPLIED
        tblunits        %tblunit;       #IMPLIED
        –– the dimensions will be meaningless for the Braille and
        computer voice versions but may be useful for large print.
          Values established for THEAD, TBODY and TFOOT, either in
        attributes or within the COLDEF attributes, override
        the comparable settings in TGROUP –– >

<!–– Table Column Definitions: ––>
<!–– there is exactly one COLDEF for each column in the table ––>
<!ELEMENT coldef   – o    EMPTY –– only exists to hold attributes ––>
<!ATTLIST coldef
        colsep          %vborder;       ”vyes”
        halign          %halign;        ”Left”
        alignchr                CDATA           #IMPLIED
        charpos         CDATA           #IMPLIED
        colwd           CDATA           #IMPLIED
        tblunits             %tblunit;          #IMPLIED
        topsep          %hborder;       ”hyes”
        –– these attributes will be meaningless for the Braille and
        computer voice versions but may be useful for large print
        with the potential exception of vertical borders in Braille. ––>

<!–– Table Row Definitions: ––>
<!ELEMENT hdrow    – o    (hdcell, shortxt?)+
         –– SHORTXT may be included with any column head cell and covers
            the voice requirement for alternative text to be provided
            if the cell content is too long to appear in the
            voiced header.              ––    >

<!ELEMENT row   – o    ((stubcell, shortxt?)?, (sstcell, shortxt?)*,
                              (cell, shortxt?)+)
            –– STUBCELL and SSTCELL (essentially sub–stubcell) support
            differentiation between, for example, items and totals.
            This augments the original AAP TSB element by allowing
            the sub–stub to be repeatable (and thereby become sub–sub–stub
            and so on).
               SHORTXT may be included with any stub–type cell and covers
            the voice requirement for alternative text to be provided
            if the cell content is too long to appear in the
            voiced stub.
               [In non–ICADD DTDs, SHORTXT may be used with any head or
            stub cell to carry axis markers: In a database application,
            for example, SHORTXT might carry the names of fields     –– >
<!ATTLIST (row | hdrow)
        rowsep          %hborder;       ”hyes”
        valign          %valign;            ”top”
        rowht           CDATA           #IMPLIED
        tblunits             %tblunit;      #IMPLIED
        leftsep         %vborder            ”vyes”  >

<!–– Table Cell Definitions: ––>
<!ELEMENT cell   – o    %m.cell;     >

<!ELEMENT (hdcell | stubcell | sstcell)   – o    %m.cell;     >
```

```
<!ELEMENT shortxt   – o    (#PCDATA)     >
<!ATTLIST (hdcell | stubcell | sstcell | cell)
                ID              ID              #IMPLIED
        %rprops;           -- can override settings in row --
        %cprops;           -- can override settings in coldef --
        -- cell positioning: --
        colstart        NUMBER          #IMPLIED
        colspan         NUMBER          "1"          -- span >= 1 --
        rowstart        NUMBER          #IMPLIED
        rowspan         NUMBER          "1"          -- span >= 1 -->

<!-- H3, NOTE, XREF and FTN are declared in the basic ICADD-22 DTD
        and repeated here for convenience: [Non-ICADD users of this
        DTD fragment may wish to re-declare or eliminate these. -->
<!ELEMENT (h3 | note | xref | ftn)    - -      (#PCDATA)            >
```

A.10 International Organization for Standardization (ISO)

Normen-DTD (ISO/IEC 9573–11:1992)

```
<!-- (C) International Organization for Standardization 1988, 1989, 1990, 1991.
      Permission to copy in any form is granted for use with conforming SGML systems
      and applications as defined in ISO 8879, provided this notice is included in
      all copies.
      Creator:   Anders Berglund.
-->

<!ELEMENT tab         - -  (tabcap?, tabdesc?, tabcomm?, tabmat,
                            (%m.outline;)*)           -(tab)           >
<!ATTLIST tab               id       ID        #IMPLIED
                            width    CDATA     page
                            place    (top|fixed|bottom)  top
                            align    (left|center|right) center
                            type     (tab|chart|table|form|listing)
                                              tab
                            number   CDATA     #IMPLIED                >

<!ELEMENT  tabcap     - O  %m.inline;               -- Table caption -->

<!ELEMENT  tabdesc    - O  %m.outline;          -- Table description -->

<!ELEMENT  tabcomm    - O  %m.inline;               -- Table comment -->

<!ELEMENT tabmat      - -  (tabhead?, tabfoot?, tabbody)
                            -(tabmat)                 -- Table matter -->
<!ATTLIST tabmat            rotate   (0|90|180|270) 0
                            width    CDATA     #IMPLIED
                            compact  (compact) #IMPLIED
                            pointsz  (6|7|8|9|10|11|12|14|16|18|20) "9"
                            trules   CDATA     "T B L R"
                            domains  NMTOKENS  none
                        -- attributes specifying defaults for lower levels --
                            gridx    CDATA     "*"
                            gridy    CDATA     "*"
                            arrange  CDATA     "none"
                            rrules   CDATA     "T B L R"
                            crules   CDATA     "T B L R"
                            cvalign  (T|C|B)   "T"
                            caligns  CDATA     "L"
                            calignps CDATA     "-"                     >

<!ELEMENT (tabhead|tabfoot)    - O   (arow)         -- header/footer -->
<!ATTLIST tabhead           headhi   (0|1|2|3) "2"                     >
```

```
<!ELEMENT tabbody     – O   (arow+)       --Body of tabular material -->
<!ATTLIST tabbody     --   brules   CDATA     "T B L R"  --
                   -- attributes specifying defaults for lower levels --
                         gridx    CDATA     #IMPLIED
                         gridy    CDATA     #IMPLIED
                         arrange  CDATA     #IMPLIED
                         rrules   CDATA     #IMPLIED
                         crules   CDATA     #IMPLIED
                         cvalign  (T|C|B)   #IMPLIED
                         caligns  CDATA     #IMPLIED
                         calignps CDATA     #IMPLIED                >

<!ELEMENT arow        – O  (c+)                     -- Arranged row -->
<!ATTLIST arow             pointsz  (6|7|8|9|10|11|12|14|16|18|20) #IMPLIED
                           split    (yes|no)  no
                           gridx    CDATA     #IMPLIED
                           gridy    CDATA     #IMPLIED
                           arrange  CDATA     #IMPLIED
                           rrules   CDATA     #IMPLIED
                   -- attributes specifying defaults for lower levels --
                           crules   CDATA     #IMPLIED
                           cvalign  (T|C|B)   #IMPLIED
                           caligns  CDATA     #IMPLIED
                           calignps CDATA     #IMPLIED                >

<!ELEMENT c           – O  (sc+)                -- Table matter cell -->
<!ATTLIST c                nr       NUTOKEN   #IMPLIED
                           pointsz  (6|7|8|9|10|11|12|14|16|18|20) #IMPLIED
                           type     (head|body) body
                           rotate   (0|90|180|270) 0
                           crules   CDATA     #IMPLIED
                           cvalign  (T|C|B)   #IMPLIED
                           calign   CDATA     #IMPLIED
                           calignp  CDATA     #IMPLIED
                           domain   NMTOKEN   "none"                  >

<!ELEMENT sc          O O  (%m.cell;|hrule)* +(tn)
                                            --Table matter sub-cell -->

<!ELEMENT tn          – –  %m.outline;                  -- Tablenote -->
<!ATTLIST tn               id       ID        #IMPLIED                >

<!ELEMENT hrule       – O  EMPTY                     -- horizontal rule -->
```

A.11 Open Software Foundation (OSF)

OSF-Book-DTD

```
<!ENTITY % common-attr
"       qualify         CDATA    #IMPLIED
        id              ID" >

<!ELEMENT table         – O     ( title-row | row )*    -(table)        >
<!ATTLIST table
        ncols           NUMBER   #IMPLIED -- num of cells/row should match --
        model           CDATA    #IMPLIED -- column prototypes for this table --
        colwidth        NUTOKENS #IMPLIED -- absolute widths of cols --
        colweight       NUMBERS  #IMPLIED -- column weights --
        halign          CDATA    #IMPLIED -- horiz alignment for columns --
        valign          CDATA    #IMPLIED -- vertical alignment for columns --
        colsep          NUMBERS  #IMPLIED -- use col separators (lines)? --
        rowsep          NUMBERS  #IMPLIED -- use row separators (lines)? --
        wrap            NUMBERS  #IMPLIED -- wrap hints for columns --
        frame           (top | bottom | topbot | all |  sides | none)   #IMPLIED
                                          -- style of frame for entire table --
        repeathead      NUMBER   #IMPLIED -- carry title rows to other pages --
        %common-attr;            #IMPLIED >
```

```
<!ELEMENT title-row     - O     ( cell* )       >

<!ELEMENT row           - O     ( ( cell | c-cell )* )  >
<!ATTLIST ( title-row | row )
        model           CDATA    #IMPLIED -- model override for this row --
        halign          CDATA    #IMPLIED -- horiz alignment for columns --
        valign          CDATA    #IMPLIED -- vertical alignment for columns --
        colsep          NUMBERS  #IMPLIED -- use col separators (lines)? --
        rowsep          NUMBERS  #IMPLIED -- use row separators (lines)? --
        wrap            NUMBERS  #IMPLIED -- wrap hints for columns --
        %common-attr;            #IMPLIED >

<!ELEMENT cell          - O     %m.cell;           >

<!ELEMENT c-cell        - O     %m.cell;    >
<!ATTLIST ( cell | c-cell )
        model           CDATA   #IMPLIED -- model override for this cell --
        halign          CDATA   #IMPLIED -- horiz alignment for cell --
        valign          CDATA   #IMPLIED -- vertical alignment for cell --
        colsep          NUMBER  #IMPLIED -- use col separators (lines)? --
        rowsep          NUMBER  #IMPLIED -- use row separators (lines)? --
        wrap            NUMBERS #IMPLIED -- wrap hints for cell --
        %common-attr;           #IMPLIED >
```

A.12 SoftQuad Author/Editor

Interne DTD

```
<!ENTITY % TblBody      "(TblHead | TblFoot | TblBody)">

<!ENTITY % TblCDfs      "TblCDefs">

<!ENTITY % TblCDef      "TblCDef">

<!ENTITY % TblRows      "TblRows">

<!ENTITY % TblRow       "TblRow">

<!ENTITY % TblCell      "TblCell">

<!ENTITY % CellMdl      "(para)+" >

<!ENTITY % VBorder
        "(VSingle|VDouble|VTriple|VDash|VDot|VBold|VBlank|VNone)">

<!ENTITY % HBorder
        "(HSingle|HDouble|HTriple|HDash|HDot|HBold|HBlank|HNone)">

<!ENTITY % VAlign  "(Top|Middle|Bottom)">

<!ENTITY % HAlign  "(Left|Both|Center|Right|Char)">

<!ENTITY % TblUnit      "(percent|pixels|points|picas|mm|cm|inches)">

<!ENTITY % RProps
        'RowSep         %HBorder;  #IMPLIED
         VAlign         %VAlign;   #IMPLIED'>

<!ENTITY % CProps
        'ColSep         %VBorder;       #IMPLIED
         HAlign         %HAlign;        #IMPLIED
         AlignChr       CDATA           #IMPLIED
         CharPos        CDATA           #IMPLIED'>

<!-- TblBody is the top-level table element -->
<!ELEMENT %TblBody; - -  ((%TblCDfs;), (%TblRows;))>
<!ATTLIST %TblBody;
        TblWd           CDATA           #IMPLIED
        TblHt           CDATA           #IMPLIED
        TblUnits        %TblUnit;       #IMPLIED >
```

```
<!-- Table Column Definitions: -->
<!ELEMENT %TblCDfs;  - -   (%TblCDef;)+>
<!ATTLIST %TblCDfs;
        -- CProps with default values to apply to all columns --
        ColSep          %VBorder;       "VSingle"
        HAlign          %HAlign;        "Left"
        AlignChr        CDATA           #IMPLIED
        CharPos         CDATA           #IMPLIED
        -- default width for all cells in all columns --
        ColWd           CDATA           #IMPLIED
        TblUnits        %TblUnit;       #IMPLIED
        -- default top border for all cells in row 1 --
        TopSep          %HBorder;       "HSingle" >

<!-- there is exactly one TblCDef for each column in the table -->
<!ELEMENT %TblCDef;  - -   EMPTY -- Only exists to hold attributes -->
<!ATTLIST %TblCDef;
        %CProps;                -- overrides settings in TblCDfs --
        -- width for all cells in this column --
        ColWd           CDATA           #IMPLIED
        TblUnits        %TblUnit;       #IMPLIED
        -- top border for cell in row 1 --
        TopSep          %HBorder;       #IMPLIED        -- overrides TblCDfs defn -->

<!-- Table Rows: -->
<!ELEMENT %TblRows;  - -   (%TblRow;)+>
<!ATTLIST %TblRows;
        -- RProps with default values to apply to all rows --
        RowSep          %HBorder;       "HSingle"
        VAlign          %VAlign;        "Top"
        -- default height of all rows --
        RowHt           CDATA           #IMPLIED
        TblUnits        %TblUnit;       #IMPLIED
        -- default left border for all cells in column 1 --
        LeftSep         %VBorder        "VSingle" >

<!ELEMENT %TblRow;  - -   (%TblCell;)+>
<!ATTLIST %TblRow;
        %RProps;        -- can override settings in TblRows --
        -- height of this row --
        RowHt           CDATA           #IMPLIED
        TblUnits        %TblUnit;       #IMPLIED
        -- left border for cell in column 1 --
        LeftSep         %VBorder        #IMPLIED        -- overrides TblRows setting -->

<!-- Table Cells -->
<!ELEMENT %TblCell;  - -   %m.cell;>
<!ATTLIST %TblCell;
        %RProps;                -- can override settings in TblRow --
        %CProps;                -- can override settings in TblCDef --
        -- cell positioning --
        ColStart        NUMBER          #IMPLIED
        ColSpan         NUMBER          "1"             -- span >= 1 --
        RowStart        NUMBER          #IMPLIED
        RowSpan         NUMBER          "1"             -- span >= 1 -->
```