XHTML を利用した語彙データの格納形式

高橋　洋成

(筑波大学)

s025035@u.tsukuba.ac.jp

0　はじめに

収集された語彙データを電子化することにより、紙媒体のみならず、デスクトップ端末や携帯端末など、多種多様な媒体にデータを持ち運ぶことが可能となる。また、関連するデータ同士をリンクさせ、相互参照を行うことも容易になる。

しかしながら、データを電子化する際にどのような形式を用いるかは、熟慮すべき課題である。もし、特定のアプリケーションに依存する形式で保存されれば、そのアプリケーションを有していない利用者に多大な制限を与えることになる。また、そのアプリケーションが開発終了するなどして利用不可能になった場合、データを読み込むことさえ困難になってしまう。

専用のアプリケーションを用いれば、簡単に高精度のデータ管理が可能になり、そのこと自体は問題ではない。問題が生じうるのはデータを公開し、長期に渡って保存・更新していくときである。データの形式は、特定のアプリケーションに依存せず、長期の使用に耐えうる堅牢性と、必要に応じて細部を変更できる柔軟な形式であることが望ましい。

XML は、まさにこのような必要性から生み出されたものである。だが一方で、XML の応用言語を一から作り出し、実際に使用しながら完成度を高めていくやり方では、学習コストも運用コストも高くつく。そこで、本稿では語彙データの管理に (X)HTML の表形式を利用することを提案する。そのメリットを以下に挙げる。

- 汎用性。HTML ブラウザさえあれば、どのような環境からでもデータを利用できる。また、(X)HTML 形式は多くのエディタやワードプロセッサ

*本稿は 2007 年度～2009 年度科学研究費基盤研究 (B)「オモ・クシ系少数言語の調査研究及び地理情報システムを用いたデータベース構築」代表：乾秀行（山口大学）（研究課題番号：19401023）による研究成果の報告である。

でサポートされており、データの入力や編集における障壁が比較的少ない。さらに、XML である XHTML は文字符号化方式として UTF-8 または UTF-16 が前提となっており、特殊文字の使用に関する制限が少ない[1]。

- 簡易性。(X)HTML に関する情報は豊富に存在しており学習コストが低い。また、入力段階から HTML ブラウザを用いて確認しながら作業できるため心理的な障壁も低く、完成後すぐに Web ページとして公開できる。
- 堅牢性。(X)HTML の表書式には明確な規定があり、一貫した形式になる。また、記入ミスを比較的発見しやすい。
- 拡張性。(X)HTML の表形式は、それ自体で十分な表現力を持っている。それに加え、必要に応じて XML 関連規定と組み合わせて拡張を施すこともできる。
- 高橋 (2009) で提案したエチオピア言語 Web サイトのコンテンツとしてそのまま利用できる。

2 語彙データのマークアップ

2.1 XHTML Tables Module

HTML 3.2 (1997) で導入された表形式は、HTML 4.0 (1997, 1999) において、非視覚系媒体での利用のために、また HTML ブラウザでの表示を補助するために、表面には現れない様々な要素型や属性が追加された。XHTML 1.0 は HTML 4.0 を XML として再定義したものであり、表関連の要素や属性もそのまま引き継いでいる。そして、それらは XHTML Modularization (2001) において表モジュールとして分離され、必要に応じて他の XML 応用言語に組み込むことが可能になった。本稿は原則として XHTML Modularization による XHTML Tables Module のモデルおよび書式を用いる。とは言え、実際には HTML 4.0 のものと全く同じである[2]。

2.2 表モデル

XHTML の基本的な表モデルを図 1 に示す。

---

[1]アプリケーション側の制限で Unicode を扱えない場合はありうる。だが、それはあくまでアプリケーション側の問題であり、XML およびその応用言語としての XHTML では、デフォルトの文字符号化方式として UTF-8 と UTF-16 が指定されている。

[2]現在 World Wide Web Consortium および Web Hypertext Application Technology Working Group が策定している HTML5 では、本稿で用いる属性の一部が、現在ほとんど用いられていないという理由で削除される。だが本稿は、それらの属性にこそ有用性を見出している。

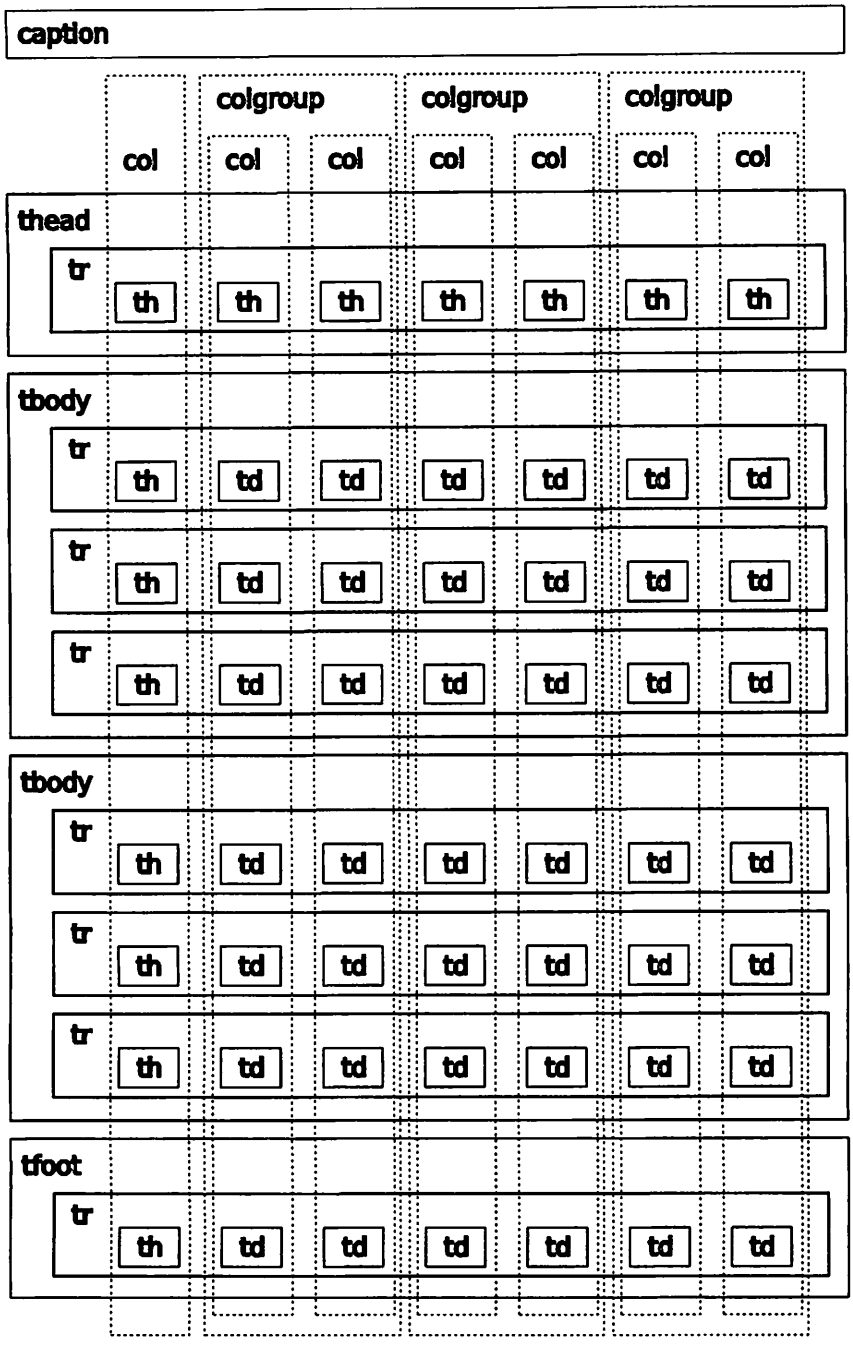

図 1: XHTML の表構造

2.3 表の基本構造

本節では、XHTML における表の基本的な書式と、それを利用した語彙データのマークアップ方法について概観する。

2.3.1 行とセル

最も単純な表は次のようなものである。

English	Amharic	Argobba	Zway
one	and	hand	had
head	ra:s	di'na	ohat

これは XHTML で次のように書かれる[3]。

```
<table xmlns="...">
  <tr>
    <th>English</th>
    <th>Amharic</th>
    <th>Argobba</th>
    <th>Zway</th>
  </tr>
  <tr>
    <th>one</th>
    <td>and</td>
    <td>hand</td>
    <td>had</td>
  </tr>
  <tr>
    <th>head</th>
    <td>ra:s</td>
    <td>di'na</td>
    <td>ohat</td>
  </tr>
</table>
```

まず、表（table 要素）の範囲を開始タグと終了タグで明示する。この表には 3 個の行（tr 要素、table row）があり、それぞれの範囲を開始タグと終了タグで明示する。各行には 4 個のセル（td 要素、table data cell）があり、それぞれ開始タグと終了タグでデータを囲む。

[3]以下、名前空間宣言 xmlns="http://www.w3.org/1999/xhtml" は煩雑さを避けるため省略する。

ただし、ここにはデータそのものではなく、行または列の見出しとなるセルもある。見出しセル（th 要素、table header cell）には専用のタグを用い、データそのもの（td 要素）とは区別する。また、人間の目には見出しセルがどの行、どの列のものであるか推測できるが、非視覚系のブラウザ、あるいは機械処理の観点からは、見出しの有効範囲を示しておくと良い。

```
<table xmlns="...">
  <tr>
    <th scope="col">English</th>
    <th scope="col">Amharic</th>
    <th scope="col">Argobba</th>
    <th>Zway</th>
  </tr>
  <tr>
    <th scope="row">one</th>
    <td>and</td>
    <td>hand</td>
    <td>had</td>
  </tr>
  <tr>
    <th scope="row">head</th>
    <td>ra:s</td>
    <td>di'na</td>
    <td>ohat</td>
  </tr>
</table>
```

th 要素の scope 属性により、その見出しセルの有効範囲が列方向（col）か行方向（row）かが明示される。例えば、“Amharic” が列方向の見出し、“one” が行方向の見出し、それらの交差地点には /and/ がある、ということが明確になるのである。

さらに、class 属性を用いて行やセルに固有の情報を持たせることができる。

```
....
  <tr class="vocab-basic">
    <th scope="row">one</th>
  ....
  <tr class="vocab-basic">
    <th scope="row">head</th>
....
```

上記は、“one” と “head” を含む行が基礎語彙（“vocab-basic”）であることを表している。class 属性値には任意の値を、空白区切りで何個でも置くことができる。例えば、この XHTML データを Web ページとして表示する際に次のようなスタイルシート（CSS）を適用すれば、基礎語彙を含まない全ての行が非表示となり、基礎語彙のみが表示される。

```
tr:not(.vocab-basic) {
  visibility: collapse;
}
```

2.3.2 行のグループ化

前節で触れたように、class 属性を用いることで行に固有の情報を持たせることができる。だが、多くの行が共通のグループに属する場合には、各行に class 属性を付与するより、まとめて指定できた方が便利である。

```
<table xmlns="...">

  <!-- 表ヘッダ -->
  <thead>
    <tr>
      <th scope="col">English</th>
      <th scope="col">Amharic</th>
      <th scope="col">Argobba</th>
      <th scope="col">Zway</th>
    </tr>
  </thead>

  <!-- 表フッタ -->
  <tfoot>
    <tr>
      <td>語彙数</td>
      <td axis="語彙数">2</td>
      <td>言語数</td>
      <td axis="言語数">3</td>
    </tr>
  </tfoot>

  <!-- 数詞 -->
  <tbody class="vocab-numeral">
    <tr>
      <th scope="row">one</th>
```

```
      <td>and</td>
      <td>hand</td>
      <td>had</td>
    </tr>
    ....
  </tbody>

  <!-- 身体 -->
  <tbody class="vocab-body">
    <tr>
      <th scope="row">head</th>
      <td>ra:s</td>
      <td>di'na</td>
      <td>ohat</td>
    </tr>
    ....
  </tbody>
</table>
```

ここではまず、見出しセルのみを含む行を表ヘッダ（thead 要素、table header）とし、語彙データ自体とは区別している。また、同様に語彙データとは直接関係のない参考情報を表フッタ（tfoot 要素、table footer）とし、語彙データ自体とは区別する（なお、axis 属性に関しては 2.4.2 節で詳述する）。

図 1 と異なり、tfoot 要素が thead 要素の直後に置かれていることに注意されたい。表モデルや HTML ブラウザでの表示上は、表フッタは表の最後に位置する。しかしながら、XHTML での書式上、tfoot 要素は表データ本体（tbody 要素、table body）よりも前に置かれねばならない[4]。これは、例えば表ヘッダと表フッタを固定し、データ本体のみをスクロールさせるときなどに、表フッタ情報を先に取得できた方が有利だからである。

表ヘッダと表フッタは 1 個の表につき 1 個ずつしか置けないが、表データ本体である tbody 要素は何個でも置くことができる。したがって、語彙データを「数詞」「身体」などのようにグループ化して各々を tbody 要素とし、さらに class 属性を付与することで、その中にある全ての行の語彙グループが分かるようになる。

[4]HTML5 では table 要素型の内容モデルが変更され、tfoot 要素を table 要素の最後に置いても良いことになっている。

2.3.3 列のグループ化

行と同じように、列もグループ化できる。ただし、行のグループ化は実際の行を囲む直感的な書式であるのに対し、列のグループ化は抽象的な書式で行われる。行（tr 要素）または行グループ（thead 要素、tfoot 要素、tbody 要素）が出現するよりも前に、列（col 要素）および列グループ（colgroup 要素）を並べるのである。

```
<table xmlns="...">

  <!-- 1 列目 -->
  <col class="heading English" />

  <!-- 2..4 列目 -->
  <colgroup class="Semitic">
    <col class="Amharic" />
    <col class="Tigre" />
    <col class="Tigrinya" />
  </colgroup>

  <!-- 5..7 列目 -->
  <colgroup class="Cushitic">
    <col class="Tsamai" />
    <col class="Arbore" />
    <col class="Konso" />
  </colgroup>

  <!-- 8..10 列目 -->
  <colgroup class="Omotic">
    <col class="Male" />
    <col class="Ari" />
    <col class="Basketo" />
  </colgroup>

  <!-- 表ヘッダ -->
  <thead>
  ....
</table>
```

col 要素と colgroup 要素は、HTML ブラウザでは表示されない。これらの後に続く各行が、どのような列に区切られているかを明示するのみである。ゆえに図 1 では実線ではなく点線によって表現している。

上記例では、1 列目は英語による見出し列であり、2 行目から 4 行目がセム諸語、5 行目から 7 行目がクシ諸語、8 行目から 10 行目がオモ諸語であることを明示する。列グループに属する各列が具体的に何を示すかは、その中に含まれる col 要素の class 属性によって示されている。上記のように、col 要素は単独で用いても良いし、colgroup 要素内に入れても良い。ただし、col 要素は空要素であるため内容を持つことはできない。

このように言語グループを定義すれば、例えば次のスタイルシート（CSS）を適用することでセム諸語の列のみを表示することができる。

```
colgroup:not(.Semitic) {
  visibility: collapse;
}
```

2.3.4 キャプション

表のキャプションは caption 要素で示される。caption 要素は、あらゆる表関連データの先頭に置かなければならない。

```
<table xmlns="...">
  <caption>Nouns</caption>
  ....
</table>
```

ただし、キャプションの表示位置はスタイルシート（CSS）で変更可能である。以下の例はキャプションを表の下部に表示する。

```
caption {
  caption-side: bottom;
}
```

表フッタ（tfoot 要素）と同じように、どの位置に表示されるかに関わらず、(X)HTML の書式上、caption 要素は表データの先頭でなければならない。

キャプションは、語彙データを検索する際、どの表から検索するかという最初の絞り込みに利用することができる。

2.4 多層的な表構造

本節では、XHTML の表を語彙データベースとして利用すべく、やや複雑な表の書式を利用する。

2.4.1 データから見出しの逆参照

見出しセルの有効範囲を scope 属性で示すことは、データが含まれているセルを識別しやすくする (2.3.1 節)。逆に、あるデータを含むセルが与えられたとき、行方向と列方向の見出しによって、そのデータの性質が分かる。例えば、次の表において /ohat/ が与えられたとき、それは言語 "Zway" において、"head" という意味を持つ語であることが分かる。

English	Amharic	Argobba	Zway
one	and	hand	had
head	ra:s	di'na	ohat

しかし、次のようなデータが存在する可能性がある。

English	Amharic	Argobba	Zway
yes	awo	ʔo:	
no	jəl:əm	ha'ku:nam	ʔil:o

この表では、/ʔo:/ は言語 "Argobba" と "Zway" の両方の列にまたがっている。これを XHTML で表現すると次のようになる。

```
<table xmlns="...">
  ....
  <thead>
    <tr>
      <th scope="col">English</th>
      <th scope="col">Amharic</th>
      <th scope="col">Argobba</th>
      <th scope="col">Zway</th>
    </tr>
  </thead>
  ....
  <tbody class="vocab-miscellaneous">
    <tr>
      <th>yes</th>
      <td>awo</td>
      <td colspan="2">ʔo:</td>
    </tr>
    <tr>
      <th>no</th>
      <td>jəl:əm</td>
      <td>ha'ku:nam</td>
```

```
        <td>?il:o</td>
      </tr>
    </tbody>
    ....
  </table>
```

/?o:/ を含むセル（td 要素）に colspan 属性が付けられ、セルが 2 列にまたがることを示した結果、その行における td 要素の総数が他の行に比べ足りなくなる。この表を HTML ブラウザで表示するときは問題ないが、データベースとして利用する場合、セルの列番号を数えて対応する見出しセルを参照するという単純な手法では意図通りにならなくなる。

列番号を算出する際に td 要素に付与されている colspan 属性値を加算するのも良いが、ここでは見出しセル（th 要素）に ID を付与し、さらに各セル（td 要素）が参照すべき見出しセルの ID を headers 属性で列挙する方法を考える。

```
<table>
  ....
  <thead>
    <tr>
      <th scope="col" id="lang-English">English</th>
      <th scope="col" id="lang-Amharic">Amharic</th>
      <th scope="col" id="lang-Argobba">Argobba</th>
      <th scope="col" id="lang-Zway">Zway</th>
    </tr>
  </thead>
  ....
  <tbody class="miscellaneous">
    <tr>
      <th id="vocab-yes">yes</th>
      <td headers="vocab-yes lang-Amharic">awo</td>
      <td headers="vocab-yes lang-Argobba lang-Zway" colspan="2">o:</td>
    </tr>
    <tr>
      <th id="vocab-no">no</th>
      <td headers="vocab-no lang-Amharic">yel:em</td>
      <td headers="vocab-no lang-Argobba">ha'ku:nam</td>
      <td headers="vocab-no lang-Zway">il:o</td>
    </tr>
  </tbody>
  ....
</table>
```

ここでは、/ʔoː/ を含むセルが 3 個の見出しセル（“vocab-no”、“lang-Argobba”, “lang-Zway” という ID を持つ各 th 要素）と関連していることが headers 属性によって示されている。つまり、この語が言語 “Argobba” および “Zway” において “yes” という意味を持つことが分かる。headers 属性によって、表がどんなに不規則な形になっても、データと見出しとを明確に結び付けることができるのである。

データが列をまたぐ場合だけでなく、行をまたぐ場合も同様である。

English	Amharic	Chara
brother (older)	wəndɪm	iʃi
brother (younger)		geʃa

これを XHTML で次のように表現する。/wəndɪm/ を含むセル（td 要素）に rowspan 属性が付与され、語データが次の行にまたがることが明示された結果、次の行（tr 要素）には “Amharic” を参照するセル（td 要素）がないことに注意されたい。

```
<table>
  ....
  <thead>
    <tr>
      <th scope="col" id="lang-English">English</th>
      <th scope="col" id="lang-Amharic">Amharic</th>
      <th scope="col" id="lang-Chara">Chara</th>
    </tr>
  </thead>
  ....
  <tbody class="family">
    <tr>
      <th id="vocab-brother-older">brother (older)</th>
      <td headers="lang-Amharic vocab-brother-older vocab-brother-younger"
          rowspan="2">w@ndim</td>
      <td headers="lang-Chara vocab-brother-older">wendim</td>
    </tr>
    <tr>
      <th id="vocab-brother-yunger">brother (younger)</th>
      <td headers="lang-Chara vocab-brother-yunger">geSa</td>
    </tr>
  </tbody>
  ....
</table>
```

より典型的な例として、語彙をグループ化したとき、語彙グループ名を見出しに用いる場合を考える。

English	Amharic	Argobba	Zway
	Numeral		
one	and	hand	had
two	hulət:	xəʔeːt	hojt
	Body		
head	raːs	diʼna	ohat
mouth	af	aːf	ʔafɪn

この表の語彙データは、数詞を示す行グループと身体名称を示す行グループとに分類され、各々 "Numeral" と "Body" という見出しで区切られている。すなわち、各セルは言語に関する見出しと、意味に関する見出しに加え、語彙グループに関する見出しにも関連付けられている。

```
<table xmlns="...">
  <thead>
    <tr>
      <th scope="col" id="lang-English">English</th>
      <th scope="col" id="lang-Amharic">Amharic</th>
      <th scope="col" id="lang-Argobba">Argobba</th>
      <th scope="col" id="lang-Zway">Zway</th>
    </tr>
  </thead>
  <tbody class="numeral">
    <tr>
      <th colspan="4" id="vocab-group-numeral">Numeral</th>
    </tr>
    <tr>
      <th id="vocab-one">one</th>
      <td headers="vocab-one vocab-group-numeral lang-Amharic">and</td>
      <td headers="vocab-one vocab-group-numeral lang-Argobba">hand</td>
      <td headers="vocab-one vocab-group-numeral lang-Zway">had</td>
    </tr>
    <tr>
      <th id="vocab-two">two</th>
      <td headers="vocab-two vocab-group-numeral lang-Amharic">hulet:</td>
      <td headers="vocab-two vocab-group-numeral lang-Argobba">xə'e:t</td>
      <td headers="vocab-two vocab-group-numeral lang-Zway">hojt</td>
```

```
    </tr>
  </tbody>
  <tbody class="body">
    <tr>
      <th colspan="4" id="vocab-group-body">Body</th>
    </tr>
    <tr>
      <th id="vocab-head">head</th>
      <td headers="vocab-head vocab-group-body lang-Amharic">ra:s</td>
      <td headers="vocab-head vocab-group-body lang-Argobba">di'na</td>
      <td headers="vocab-head vocab-group-body lang-Zway">ohat</td>
    </tr>
    <tr>
      <th id="vocab-mouth">mouth</th>
      <td headers="vocab-mouth vocab-group-body lang-Amharic">af</td>
      <td headers="vocab-mouth vocab-group-body lang-Argobba">a:f</td>
      <td headers="vocab-mouth vocab-group-body lang-Zway">?afin</td>
    </tr>
  </tbody>
</table>
```

実際には、各語彙グループは tbody 要素によって明示されており（cf. 2.3.2）、上記のような書き方は冗長である。だが、こうした冗長性はデータの検査に役立つ。例えば、XHTML の書式上、列を挿入（すなわち個別言語の語彙データを追加）したとき、全ての行において、列に対応する位置にセルを追加していくことになる。手作業での挿入はもちろん、機械的な操作によってでさえ、この作業は極めてミスを起こしやすい。結果として、語データが本来属すべき列（言語列）にない、ということになりがちである。そうした場合、もし headers 属性が存在していれば、セルが本来参照すべき見出しセルと、セルが実際に置かれている位置から参照できる見出しセルとを比較して、不整合が生じている部分を特定することができる。

2.4.2 表の多軸化

2.4.1 節の headers 属性は、セルが参照すべき見出し情報をセル自体に持たせることで、単純に横軸と縦軸を辿るだけでは見出しに辿り着けない複雑な表に対応させるものであった。

ここで言う「見出し」は、「軸」における各「点」と言い換えることができよう。例えば、「言語軸」には “Amharic”、“Argobba”、“Zway”、……といった「点」が並んでいる。一方、「語彙軸」には “one”、“two”、“head”、……などの

「点」が並んでいる。本稿で扱ってきた表データは、横軸に「言語軸」を、縦軸に「語彙軸」を選択したものである。したがって、交点となる語データは「言語軸」と「語彙軸」の両方の性質を持つ。しかし 2.4.1 節で述べたように、語データが持ちうる性質はこの 2 つだけではなく、例えば「語彙グループ軸」のような性質もありうる。つまり、1 つの語データは多数の軸の交点として表すことができる[5]。

語	→	言語軸	語彙軸	語彙グループ軸
and	→	Amharic	one	Numeral
hand	→	Argobba	one	Numeral
had	→	Zway	one	Numeral
hulət:	→	Amharic	two	Numeral
xəʔeːt	→	Argobba	two	Numeral
hojt	→	Zway	two	Numeral
raːs	→	Amharic	head	Body
di'na	→	Argobba	head	Body
ohat	→	Zway	head	Body

結局、表データとは、多数の軸の中から特に 2 つを取り出して横と縦に並べたものに過ぎない。その過程で失われた軸は、もし表データの中に（不規則な）見出しセルが残っていれば、headers 属性によって拾い出すことができる。

だが、表データ化の過程で見出しセルとしても残されず、見かけ上消滅してしまう軸もありうる。そのような軸を「復活」させるのが、axis 属性である。

ごく簡単な例を挙げる。次の表において、“Amharic” の /and/ と /raːs/ は、それぞれの音声データにリンクしている（下線は始点アンカーであることを示す）。それ以外の語は音声データを持っていない。

English	**Amharic**	**Argobba**	**Zway**
one	and	hand	had
head	raːs	di'na	ohat

この表には、音声データが存在するか否かの「音声データ軸」があると考えられる。

[5]意義素の考え方に近いかもしれない。だが、ここでは意味構造という複雑な問題には踏み込まず、あくまでデータとしての管理しやすさ・検索しやすさといった実用的な観点からの分類を行っている。

語	→	言語軸	語彙軸	語彙グループ軸	音声データ軸
and	→	Amharic	one	Numeral	あり
hand	→	Argobba	one	Numeral	なし
had	→	Zway	one	Numeral	なし
ra:s	→	Amharic	head	Body	あり
di'na	→	Argobba	head	Body	なし
ohat	→	Zway	head	Body	なし

「音声データ軸」を措定することで、表データの中から音声データを持つものだけを拾い出し、同一線上に並べることが可能になる。しかしながら、音声データを持つ語を含むセルは表データの中に点在しており、適当な位置に見出しセルを設けることができない。そこで、セル（td 要素）に axis 属性を付与し、セルが属する軸の種類を全て列挙する。axis 属性の値は任意の値のリスト（空白区切り）である。下記の例では “pronunciation-yes” と “pronunciation-no” という 2 つの点を示すことで、音声データ軸の存在を暗示している。

```
<table xmlns="...">
  <thead>
    <tr>
      <th scope="col">English</th>
      <th scope="col">Amharic</th>
      <th scope="col">Argobba</th>
      <th scope="col">Zway</th>
    </tr>
  </thead>
  <tbody class="numeral">
    <tr>
      <th colspan="4">Numeral</th>
    </tr>
    <tr>
      <th>one</th>
      <td axis="pronunciation-yes">and</td>
      <td axis="pronunciation-yes">hand</td>
      <td axis="pronunciation-no">had</td>
    </tr>
  </tbody>
  <tbody class="body">
    <tr>
      <th colspan="4">Body</th>
    </tr>
```

```
      <tr>
        <th axis="pronunciation-no">head</th>
        <td axis="pronunciation-no">ra:s</td>
        <td axis="pronunciation-no">di'na</td>
        <td axis="pronunciation-no">ohoat</td>
      </tr>
    </tbody>
  </table>
```

2.4.3 軸名と見出しの共通化の必要性

これまで説明してきたように、表の多軸化によって二次元以上のデータを表現できるのだが、その前に class 属性値、id 属性値、axis 属性値をある程度まで共通化する必要がある。これらは任意の値をとれるため、データの編集者は好きなように追加・変更できるが、そのことがデータの相互運用性に支障をきたす可能性があるからである。

そうした事態を避けるため、本稿では、言語名に lang-、語彙名に vocab-、語彙グループ名に vocab-group-、音声データが存在するものには pronunciation- という接頭辞を付けることで、「軸名-点名（見出し）」という一貫した名前付けを行ってきた。こうした名前付けには一層の議論が必要であろう。例えば乾 (2006) が論じている ISO 639 の言語タグを語彙データの言語軸にも用いることで、語彙データだけではなく様々なシステム上で言語名の扱いを統一できるであろう。

なお、今回作成した語彙データを含むエチオピア言語 Web サイトでは、言語名を含むファイル名（URI）および ID（フラグメント識別子）には、IETF 言語タグを採用した。これは、以下のサブタグの並びにより言語を識別する仕組みである。

サブタグ	:	language	-script	-region	-variant	-extension	-privateuse
典型的文字数	:	2~8	4	2~3	4~8	2~8	1~8

- language は言語コード（ISO 639）を示す。省略することはできない。
- script は書記体系コード（ISO 15924）を示す。省略可。
- region は国・地域コード（ISO 3166）を示す。省略可。
- variant は言語変種を示す（現在は標準規格なし）。省略可。
- extension はアルファベット 1 文字（x を除く）から始まる拡張コードを示す（現在は標準規定なし）。拡張コードは重複しない限り任意の数だけ並べて良い（e.g. a-myExt-b-another）。

- privateuse は x から始まる私用領域を示す。8 文字までのサブタグを任意の数だけ並べることができる。

例えば、言語タグ amh-ET-x-Amharic の構成は次のようになる。

language	=	amh（アムハラ語）
script	=	省略
region	=	ET（エチオピア）
variant	=	省略
extension	=	なし
privateuse	=	Amharic

私用領域には何を書いても良いが、ここでは言語の通称を書いておいた。例えば、言語コード amf は「Hamer-Banna」を指すが、両者を区別して amf-x-Hamer と amf-x-Banna にしたい場合もある。また、私用領域のサブタグが 8 文字以内で複数指定可能という制約を利用し、myf-ET-x-Mao-of-Bambeshi のように、名称が複数のサブタグにまたがっているものもある。なお、私用領域はあくまで私用の拡張であり、どのように処理されるかはアプリケーション依存であることに注意されたい。

また、言語コードを持たないもの、もしくは不明なものには undefined を付けた（e.g. undefined-ET-x-Chaha）。これは IETF 言語タグ形式の意図的な違反であり、意図を知らずに通常処理しようとするアプリケーションに敢えてエラーを起こさせることで、誤動作を防ぐためのものである。

3 おわりに

本稿は、XHTML Tables Module を用いて語彙データをマークアップし、さらに簡単なデータベースとしても利用するための手法を提案した。XHTML の表形式は行・列の 2 次元にとどまらず、多軸化することで n 次元のデータを表現できるからである。

また、表を多軸化するにあたり、名前付けの共通化の必要性を指摘し、実例として IETF 言語タグを利用する方法を挙げた。この点に関しては一層の議論が必要である。

【参照文献】

Austin, D. et al (eds.) 2008 *XHTML Modularization 1.1.*
http://www.w3.org/TR/2008/REC-xhtml-modularization-20081008

Davis, M. (ed.) 2009 *RFC 5646: Tags for Identifying Languages.*
http://www.rfc-editor.org/rfc/rfc5646.txt

Hickson, I. (ed.) 2010 *HTML5.*
http://www.whatwg.org/specs/web-apps/current-work/

Phillips, A. and M. Davis (eds.) 2006 *RFC 4647: Matching of Language Tags.*
http://www.ietf.org/rfc/rfc4647.txt

Raggett, D. 1997 *HTML 3.2 Reference Specification.*
www.w3.org/TR/REC-html32

Raggett, D. et al. (eds.) 1999 *HTML 4.01 Specification.*
http://www.w3.org/TR/html401

乾秀行 2006「地理情報システム（GIS）によるエチオピアのデジタル言語地図」乾秀行（編）『オモ・クシ系少数言語の調査研究及び地理情報システムを用いたデータベース構築（Cushitic-Omotic Studies 2006）』1-7.

高橋洋成 2008「XML を利用した文献データベースの構築」乾秀行（編）『オモ・クシ系少数言語の調査研究及び地理情報システムを用いたデータベース構築（Cushitic-Omotic Studies 2007）』13-29.

高橋洋成 2009「エチオピア言語の Web サイト構築」乾秀行（編）『オモ・クシ系少数言語の調査研究及び地理情報システムを用いたデータベース構築（Cushitic-Omotic Studies 2008）』5-10.

【Web サイト】

ISO 639.2: Codes for the Representation of Names of Languages.
http://www.loc.gov/standards/iso639-2/langcodes.html

ISO 639-3 Registration Authority.
http://www.sil.org/iso639-3/

ISO 3166 Maintenance agency.
http://www.iso.org/iso/country_codes.htm

ISO 15924 Registration Authority.
http://www.unicode.org/iso15924/

Language Subtag Registry.
http://www.iana.org/assignments/language-subtag-registry