XLink を利用した表組み中心の複合文書作成について

大坂　哲司
株式会社フジミック（ＸＭＬコンソーシアム基盤技術部会兼務）

概要：　業務で利用するイントラネットでは、多種多様の表が掲載されている。しかしこれら表の中のデータは、表作成の中で埋め込まれることがほとんどで、データベースと連携されているものは少ない。本稿では、ブラウザ上で表組みを簡単に操作できるシステムを提案する。このシステムは、表組み書式の作成、入力フォームの作成、表データの記述、表データの表示／印刷を一貫して行うことができる。また、１つの表は別のコンテンツから参照されることも多く、同じ表を個別に作成することは無駄である。本稿では XLink を利用した表の埋め込み方法についても提案する。

About the compound document creation using XLink constructed a table

Tetsushi Osaka　(Fujimic Inc.)

Abstract:　Various tables are displayed in the intranet used on business. However, it is most that the data which constitutes these tables is embedded in table creation, and there is little what is being interlocked with the database. In this paper, the system which can perform table operation simply by the browser is proposed. This system can perform consistently the formatting constructed a table, creation of input form, description of front data, and a display/printing of front data. Moreover, as for one table, it is useless for it to be referred to in many cases from another contents, and to create the same table individually. In this paper, it proposes also about how to embed a table of having used XLink.

１　はじめに

Web ページを見ると、至るところサイトで表組みを利用している。ページ表示に用いられる HTML は主に、テキストオブジェクト、イメージオブジェクト、テーブルオブジェクト、フォームオブジェクトから構成される。HTML において、表組みはテーブルオブジェクトによる表の表示に他ならないが、表組みはページのレイアウトのためにもしばしば利用している。

表組み XML 文書のレンダリングは次に述べる２つ方法が一般的である。１つは書式情報を持った要素のテキストデータはその書式に従ってレンダリングする方法、もう１つは、XML 文書をデータとし、別途用意した書式情報と合わせて変換処理を行いレンダリングする方法がある。前者として、電子公文書や電子出版フォーマット JepaX などの XML 文書において表組みは特別なボキャブラリを作成せず、HTML の TABLE 要素を流用している。これらの言語は HTML と同様に TABLE 要素中に表組みデータと書式情報を保持し、TABLE 要素を解釈するアプリケーションによってレンダリングされる。後者の方法は、XML 文書はデータの電子交換のために大いに利用されていることは周知の事実となっているところである。しかし XML 文書は可読性が高いと言っても、XML 文書をそのまま読む人はほとんどいない。通常 XML 文書は XSLT[XSL] によってきれいで読み易い書式に変換することが一般的である[森口ら]。XML 文書を表的に表現する場合、それらのデータは TABLE 要素に展開され、ブラウザ上にレンダリングされる。この両者において HTML に

変換されブラウザ上に表示する場合、通常 HTML の TABLE 要素を用いられる。この TABLE 要素は簡易的なテーブル表現として大変便利で、しかも簡単に表組みが作成できる。しかし TABLE 要素は、テキストの長さに応じてセルの大きさが変わる、ブランクセルにおいてセル枠が表示されない、などのレイアウトに関わる若干の問題を抱えている。また TABLE 要素で作成した表組みの印刷は、一般的な帳票印刷と異なり大きなレベル差がある。

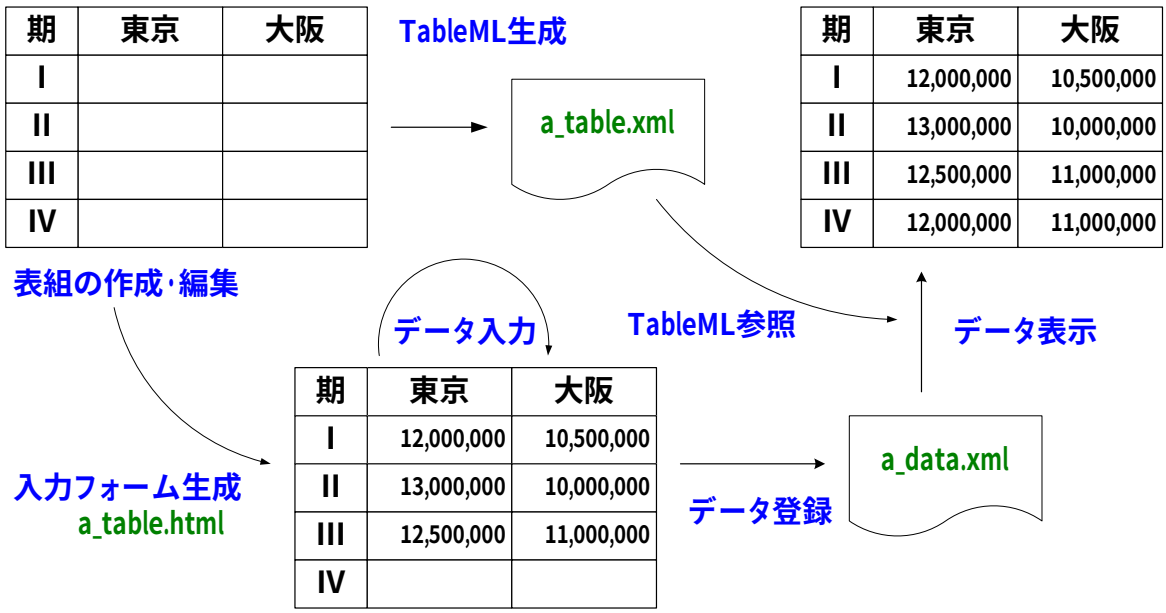
HTML の表、例えば、決算報告表、機器のスペック表などは他のコンテンツにおいて再利用される若しくは流用されることがある。このように同じ表をコンテンツ毎に作成した場合、作成時間のロス、数字の入力ミスなど思わぬコストが生じる。この状況を解決する方法として、何らかの方法でデータの一元管理が必要となる。一般的な方法として、対象となる表のみを表示する HTML ファイルを作成し、このファイルを表示するコンテンツ中で IFRAME 要素によって、その表をコンテンツ内に表示することができる。HTML ファイルを IFRAME 要素で表示する方法は簡便であるが、表用 HTML ファイルをどのように生成させるか、またその表中のデータを利用する場合どのようにするか、などの新たな問題が発生する。

表組みを含む HTML の作成は、ホームページ作成ツールによって、HTML の知識がなくても容易に作成ができる。しかしこのようなツールで作成した HTML は単なる HTML で上記に挙げた問題を解決できるものではない。表に関わる諸問題は、表書式の作成、表データの入力、表データの表示／印刷、表データの再利用など一連の操作が一貫性を持って行われていないために生じるものである。

本稿は上記の問題を解決するため表に関わる一連の操作ができるシステムの実装方法並びに表表現のための TableML について次の内容に従い報告する。第 2 章において、表書式と表データの分離、表書式と表データの合成表示、一連の操作を容易にする TableML の作成、入力フォームの自動生成、印刷に耐え得る表組の作成について、第 3 章では表の再利用を容易にする XLink による表の埋め込み表示[大坂 1]について、第 4 章では今後の展望を含めたまとめについて報告する。

2　表の生成、登録、表示について

図１　表作成の処理フロー



本稿で作成したシステムの概要は図１に示す通りである。ブラウザ上（インターネットエクスプローラーIE5.0 以上）で表組み中心の HTML を作成するアプリケーションは JavaScript で作成し、ブラウザ上でマウス操作により出来上がった表から表の書式情報を抽出して TableML を生成すると共に、表データを入力するための入力フォームを作成する。この入力フォームを用い、このフォームに記入したデータを TableML のデータ

記述に則り XML 文書として保存する。更にこの XML 文書から表の表示を行う。

２．１　表組みの作成と編集

表組みは印刷に耐え得るため、TABLE 要素を用いず、DIV 要素[CSS]を用いて枠の生成、位置編集、大きさ編集に関わる操作は次のように行った。

・生成

マウスが押下された時（onmousedown イベントが生じた時）、その座標(offsetX=event.clientX, offsetY=event.clientY)を取得する。これを開始座標として DIV 要素 dNode(id="rectframe_xxx"、dNode.style.posLeft=offsetX, dNode.style.posTop=offsetY)を生成する。次にマウスをドラッグした時（onmousemove イベントが生じた時）の移動座標から移動量(movetoX=event.clientX-offsetX, movetoY=event.clientY-offsetY)を求め、DIV 要素に大きさ(dNode.style.pixelWidth=movetoX, dNode.style.pixelHeight=movetoY)を与え、四角い枠が表示される。マウスをドラッグしている間、この操作を続けると、四角い枠が大きくなったり小さくなったりする。マウスボタンがアップした時（onmouseup イベントが生じた時)、四角い枠は固定される。なお、生成した DIV 要素は border 属性によって境界線が見えるようにする。またこのノードの id 属性は後の編集などに利用され、その値は rectframe_xxx とし、xxx は通し番号を示す。

・位置編集

マウスが枠内で押下された時、その枠のブラウザ内での表示開始座標(offsetX=event.clientX, offsetY=event.clientY)並びに HTML 中での表示座標(positionX=dNode.offsetLeft, positionY=dNode.offsetTop)を取得する。次にマウスをドラッグした時の移動座標から移動量(movetoX=event.clientX-offsetX, movetoY=event.clientY-offsetY)を求め、DIV 要素の表示位置(dNode. style.posLeft=positionX+movetoX, dNode.style.posTop=positionY+movetoY)を変えて、枠が移動する。マウスボタンがアップした時、枠の移動は停止する。

・大きさ編集

マウスが枠内で押下された時、図２に示す通り、枠編集のため枠４角にボール（編集位置を識別するための）が現れる。このボールをマウスドラッグすると枠の大きさが変更できる。枠右下のボールによる大きさの変更は枠生成の方法とほぼ同じで、枠編集前の幅、高さに移動量を加えたものが新しい幅、高さになる。その他のボールによる編集では、幅、高さのみならず、枠左上の開始座標を操作する必要がある（移動量はマイナスの値をとることはできないため)。マウスボタンがアップした時、枠の大きさは固定される。

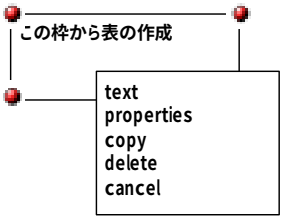


図２　表の編集画面

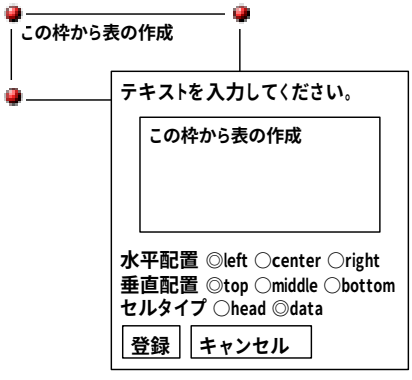


図３　テキストの編集

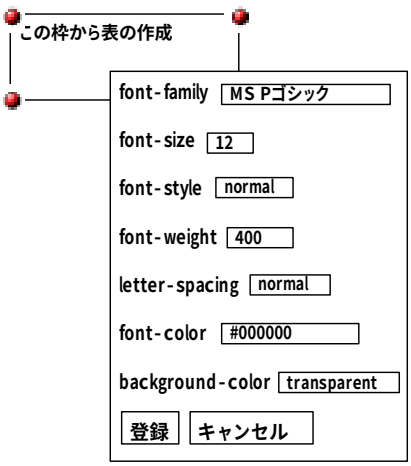


図４　プロパティの編集

・編集メニュー

表の枠は図２に示すメニューによって編集される。この編集メニューはリスト１に示すように XLink によ

って定義される。この XLink では、編集に必要な処理系をリンク対象としている。メニューに XLink を用いると、ダイナミックに処理系の変更が可能で、処理系の追加、削除が容易となる。開発途上において、種々の処理系の作動状況の確認など大変重宝する手段である。なお、XLink では xlink:show=”popup”と言う振る舞い属性は用意されていないが、XLink を利用すると言う本稿において拡張された属性である[大坂 1]。編集メニューとして用意したものは、枠の中に埋め込むテキストの編集、テキストに関わるプロパティ情報の設定、枠自身が有するプロパティとしてフォント情報や背景情報の設定、枠オブジェクトのコピー並びに削除などがある。

```
<?xml version="1.0" encoding="Shift_JIS"?>
<links xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="extended">
<resources>
        <resource xlink:type="locator" xlink:href="C:/tmp/wml/ms/mouseDrag.html#rectframe" xlink:label="A" group="A" xlink:title="menu"/>
        <resource xlink:type="resource" xlink:href="javascript:textEdit()" xlink:label="B" group="B" xlink:title="value"/>
        <resource xlink:type="resource" xlink:href="javascript:setProperties()" xlink:label="C" group="C" xlink:title="properties"/>
        <resource xlink:type="resource" xlink:href="javascript:copyrect()" xlink:label="D" group="D" xlink:title="copy"/>
        <resource xlink:type="resource" xlink:href="javascript:deleteRect()" xlink:label="E" group="E" xlink:title="delete"/>
        <resource xlink:type="resource" xlink:href="javascript:cancelmenu()" xlink:label="F" group="F" xlink:title="cancel"/>
</resources>
<arcs>
        <arc xlink:type="arc" xlink:from="A" xlink:to="B" xlink:show="popup"/>
        <arc xlink:type="arc" xlink:from="A" xlink:to="C" xlink:show="popup"/>
        <arc xlink:type="arc" xlink:from="A" xlink:to="D" xlink:show="popup"/>
        <arc xlink:type="arc" xlink:from="A" xlink:to="E" xlink:show="popup"/>
        <arc xlink:type="arc" xlink:from="A" xlink:to="F" xlink:show="popup"/>
</arcs>
</links>
```

リスト１　メニュー用 XLink

・テキストの編集

表の枠を上記の手順で作成した後、この枠の中にテキストの入力・編集ができる。またテキストに対する属性には、枠内におけるテキストの水平（左、中央、右）・垂直配置（上、中央、下）を指定する属性、テキストを表示するセルの意味合い（ヘッダ、データ、その他）を示す属性がある。テキストの水平配置はスタイル情報の text-align によって、左寄せ、中央、右寄せが指定できる。しかし DIV 要素では垂直配置を指定する方法が用意されていない（TABLE 要素は VALIGN 属性によってセル内の垂直配置を指定することができる）。DIV 要素で垂直配置を行うため、フォントメトリックスの計算によって、テキストの垂直配置を実現した。

DIV 要素内のテキストの垂直配置は、DIV 要素の高さ H に対して、テキストの表示高ｈとすると、”top”配置の場合は paddingTop=0、”bottom”配置では paddingTop=H-h、”middle”配置では paddingTop=H-h/2 とすることによって、テキストの垂直配置を行った。また paddingTop の値からテキストの垂直配置状況が逆算することも可能となった。ここで用いたフォントメトリックスは次のクラスによって実現した。

fontMetrics はフォントメトリックスのクラスで、テキストとスタイル情報若しくはノード情報を与えてフォントメトリックスの計算を行う。このクラスには、setText で別のテキストを設定したり、setNode で別のノードを設定したり、setProperty で別のスタイル情報を設定するメソッドがある。これら設定メソッドによってクラスオブジェクトの状態を変えた場合、calculate メソッドによりフォントメトリックスの再計算を行う。クラスオブジェクトのインスタンス情報は、getText メソッドでテキスト、getProperty でスタイル情報、getFontWidth でフォントサイズ、getFontHeight でフォント高、getTextWidth でテキストの表示長、getTexHeight でテキストの表示高、getTextLines でテキストの表示行数が取得できる。尚、テキストの表示長は、テキストが確実に表示される幅の DIV 要素内にテキストを格納した SPAN 要素 tNode を設け、tNode.offsetWidth により算出した[大坂 2]。テキストの表示高は、指定された幅を有する DIV 要素内にテキストを格納した SPAN 要素 tNode を設け、tNode.offsetHeight により算出した。

・プロパティの編集

表組みの枠内に表示されるテキストのスタイル情報はフォントメトリックスに関わる情報（font-family、font-size、font-style、font-weight、letter-spacing）と表示色に関わる情報（font-color、background-color）とし、これらの値を編集した。表組みを構成する DIV 要素 node に対して、f=new fontMetrics(node)とすると、クラスオブジェクト内で node.currentStyle.XXXX によって各プロパティの初期値が求まり、この値を画面に表示した。編集した値は、setProperty によってクラスオブジェクト内に設定され、表示される。

・その他の編集メニュー

GUI を通して表を作成する場合、コピーや削除は便利な機能である。本稿においてもこれらの機能をサポートしている。コピーは、マウスが押下した枠 pN に対して、そのクローン var cn=pN.cloneNode(true)を生成し､できたクローンが元の枠に重ならないようにクローンの表示位置を cn.style.pixelLeft=event.clientX、cn.style.pixelTop= event.clientY で変更した。

２．２　TableML について

上述のように本稿では GUI を用いて表組みの作成を行っている。策定した TableML は、この表組み作成過程並びに表組みに埋め込むデータ若しくはデータベースとの関係を考慮したものとなっている。

```
<table>
<tableForm id="osaka">
        <form>
                <defs>
                        <rect id="r0" width="150" height="30"/>
                        <rect id="r1" width="150" height="20" style="fill:pink;"/>
                        <rect id="r2" width="150" height="30"/>
                </defs>
                <textBox id="t0" use="#r0" transform="translate(10 0)" style="stroke:green; text-align:center"/>
                <textBox id="t1" use="#r1" transform="translate(10 40)" style="stroke:blue; text-align:center"/>
                <textBox id="t2" use="#r1" transform="translate(160 40)" style="stroke:blue; text-align:center"/>
                <textBox id="t3" use="#r1" transform="translate(310 40)" style="stroke:blue; text-align:center"/>
                <textBox id="t4" use="#r2" transform="translate(10 60)" style="strokel:blue; text-align:right"/>
                <textBox id="t5" use="#r2" transform="translate(160 60)" style="stroke:blue; text-align:right"/>
                <textBox id="t6" use="#r2" transform="translate(310 60)" style="stroke:blue; text-align:right"/>
        </form>
        <initData usedForm="#">
                <caption use="#t0 align="center" valign="top">比較表</caption>
                <dispData use="#t1">東京</dispData>
                <dispData use="#t2">名古屋</dispData>
                <dispData use="#t3">大阪</dispData>
        </initData>
</tableForm>
<tableData id="data1" usedForm="#osaka">
        <caption use="#t0" align="center" valign="top">2001年度○○価格表</caption>
        <dispData use="#t4">1200</dispData>
        <dispData use="#t5">1100</dispData>
        <dispData use="#t6">1300</dispData>
</tableData>
</table>
```

リスト２　TableML リスト

TableML（リスト２）は、表組みの書式を記述する tableForm 要素と表の中に埋め込むテキストを記述する tableData 要素から成る。書式構成は書式を示す form 要素と初期データを記述する initData 要素から成る。form 要素は表中で用いられる枠を記述（枠の大きさ、背景色など）する rect 要素と、枠をテキストボッ

クスとして捕らえ、そのテキストボックスの表示位置やテキストへのスタイル情報を記述する textBox 要素から成る。ここには記述していないが、form 要素は枠やテキストへのスタイル情報を付与する style 要素もある。デフォルトの style 要素の情報は、"font-family:'MS PGothic'; fill:none; stroke:black; font-size:12; font-weight:400; font-style:normal; letter-spacing:normal; text-decoration:none; line-height:normal; text-align:left; vertical-align:middle;"としている。

textBox 要素において、use 属性は rect 要素の id 属性と対応して枠を参照し、transform 属性（SVG に準拠）でその枠の表示位置を設定する。他の要素においても use 属性と id 属性との関係は同じである。表にはヘッダと呼ばれるセルが表の先頭にあり、列に関する情報を与えている。TableML ではこれらヘッダ情報は表の初期値として、書式の一部として initData 要素で定義する。この要素の usedForm 属性は参照する書式は自分自身の中にある書式を利用する。

表へのデータ記述を行う tableData 要素は、usedForm 属性によって参照すべき書式を URI で示し、データは dispData 要素中に記述する。データ記述は書式記述と分離可能で、それぞれ別の XML 文書として管理することができる。

2001年度○○価格表

東京	名古屋	大阪
1200	1100	1300

図５　リスト２に対応する表

リスト２中で、initData 要素中にある表ヘッダは id 属性が t1、t2、t3 で示される textBox 要素を参照し、それらの枠は id=”r1” の rect 要素から成る。同様に tableData 要素中のデータは id 属性が t4、t5、t6 で示される textBox 要素を参照し、それらの枠は id=”r2”の rect 要素から成る。

２．３　書式情報と入力フォームの生成、データ登録並びにデータ表示

図１で作成・編集した表組みはブラウザ上に表示されていて、ブラウザの HTML-DOM の中から表組みを構成するテキストボックス情報を抽出する。DIV 要素の大きさ、表示位置、スタイル情報、テキストを抽出する。まず DIV 要素の大きさでグループ化して rect 要素を生成する。次に位置情報を transform 属性に変換し、大きさに対応する rect 要素から use 属性を求めて textBox 要素を生成し、すべて DIV 要素について textBox 要素を求める。ここで、textBox 要素の id 属性値は、2.1.1 の生成の項で述べたマウスによって作成した DIV 要素の id 属性値と対応するものである。DIV 要素がテキストを有する場合、表の初期値データとして initData 要素中にそのテキストを持つ dispData 要素を生成し、対応する textBox 要素から use 属性を求め要素に追加される。これら一連の処理によって表組み用の書式情報がブラウザ上で作成され、サーバーに登録される。

また、テキストを有しない DIV 要素は入力フィードを解釈し、DIV 要素内に入力フィードを設ける。設ける入力フィードは、DIV 要素の表示高が１行分の時 INPUT 要素により、2 行以上の場合 TEXTAREA 要素によるテキストボックスの生成を行う。このように表示状態の変わった HTML を HTML-DOM から抽出し、その HTML 情報を入力フォームファイルとして生成し、サーバーに登録される。尚、この HTML ファイルは書式情報とペアになるよう、hidden 文で書式ファイル名が埋め込まれている。

表組みの入力フォームは GUI で作成し、テキストデータがない DIV 要素は入力フィードとなった HTML が表示される。入力フィードにデータを入力しデータ登録を行うと、前節の initData 要素と同様に tableData 要素中に対応する dispData 要素が生成し追加される。登録処理による入力データは TableML に則った XML 文書（リスト３）が作成され、サーバーに登録される。

```
<table>
        <tableData id="data1" usedForm="a_form.xml#osaka" >
                <caption use="#t0" align="left" valign="top">2001年度○○価格表</caption>
                <dispData use="#t4">1200</dispData>
                <dispData use="#t5">1100</dispData>
                <dispData use="#t6">1300</dispData >
        </tableData>
</table>
```

リスト３　表データ

リスト３は登録した表データで、tableData 要素の usedForm 属性に参照すべき書式情報が埋め込まれている。この書式情報に従い tableData 要素中にある表データはブラウザ上の JavaScript によって図５のように画面に表示される。ここで表示された表は、ブラウザに印刷コマンドによってきれいな印刷が提供される。

3　XLink による表の埋め込み表示

HTML コンテンツは IFRAME 要素によって現在表示している HTML 中に埋め込み表示できる。SVG コンテンツは EMBED 要素によって埋め込み表示ができ、しかも SVG 空間と EMBED 空間の大きさや縦横比が異なっても、SVG コンテンツをすべて表示できるように縮尺を変えて表示を行う。しかし IFRAME 要素では IFRAME 空間の大きさと埋め込まれるコンテンツの大きさが対応しているか、IFRAME 空間の方が大きくなければ、埋め込まれるコンテンツをすべて表示することはできない。IFRAME 空間の方が小さい場合、コンテンツがこの空間の大きさでクリップされた状態で表示され、また IFRAME 空間が大き過ぎると間の抜けたレイアウトとなったりし、見苦しいものとなる。

XLink で記述された表組みコンテンツをダイナミックに埋め込み表示するため、ブラウザ上の JavaScript による処理を行った。埋め込まれる表の大きさを表組みの tableForm 要素から求め、その大きさを持つ IFRAME 空間を生成するため、HTML の指定箇所に addTextNode 関数[大坂、野村]によって IFRAME 要素 iNode を追加する。このノードの id 属性を iNode.id=”osaka”とする。この操作によって生成した IFRAME 空間へのアクセスは、IFRAME 中のドキュメントオブジェクトが c=document.frames["osaka"].document（IFRAME 要素は HTML-DOM の frames コレクション中にある）によって参照され、IFRAME 空間内のドキュメントルートが p=c.documentElement によって求まる。このノード p の要素名は HTML である。なお、IFRAME 空間の生成にはそれなりの時間がかかる。そのため IFRAME 空間生成完了（document.readyState）をタイマー駆動によって監視しなければならない。このように生成したノードの中に TableML で記述された表組みデータを指定の書式に従い展開して、IFRAME 空間内に表組みを表示することができる。

```
リソースの記述
<resource xlink:type="locator" xlink:label="A" group="A" xlink:href="a.html" xlink:title="Aのページ"/>
<resource xlink:type="locator" xlink:label="B" group="A" xlink:href="a.html#abc" xlink:title="データ表示"/>
<resource xlink:type="locator" xlink:label="C" group="C" xlink:href="uriage.td" xlink:title="売上表"/>
<resource xlink:type="locator" xlink:label="D" group="D" xlink:href="uriage.jpg" xlink:title="売上画像"/>
<resource xlink:type="resource" xlink:label="E" group="E" xlink:href="javascript:deleteEmbed()" xlink:title="削除"/>

アークの記述
<arc xlink:type="arc" xlink:from="B" xlink:to="C" xlink:show="embed" xlink:actuate="onRequest"/>
<arc xlink:type="arc" xlink:from="B" xlink:to="D" xlink:show="embed" xlink:actuate="onRequest"/>
<arc xlink:type="arc" xlink:from="C" xlink:to="E" xlink:show="delete" xlink:actuate="onRequest"/>
<arc xlink:type="arc" xlink:from="D" xlink:to="E" xlink:show="delete" xlink:actuate="onRequest"/>
```

リスト４　XLink データ

表を埋め込み表示するための XLink ソースはリスト４の通りで、a.html の id=”abc”を XLink のリンクポイントとして、売上表と売上画像へのマルチリンクを記述している。このリンクポイントをクリックすると、この売上に関する２つのコンテンツのメニューが表示され、いずれかを選択するとこのリンクポイント直前の位置に埋め込み表示される。売上表は上述の IFRAME 要素によって表データを埋め込み表示し、画像は IMG 要素によって画像の埋め込み表示を行った。

４　まとめ

本稿では図１に示す表組みの生成編集、入力フォームの自動生成し表データの登録、表データから表の表示に関わる一連の操作が一貫して行えるシステムが開発できた。これらの処理の一貫性は TableML によって実施されている。また表組みを他のコンテンツで参照できるように、XLink を用いてその埋め込み表示を実現した。これらの機能を通して利用者に対して、表組みを簡単に作成できる環境、他のコンテンツで表組みを引用するための環境を提供すると共に、事務処理で利用される定型伝票への応用によって Web 環境でも質の高い印刷を可能とし、記入データのデータベース連動の容易にするなどのメリットを提供するものである。

本稿で利用した TableML は表作成の GUI 並びに表に埋め込まれるデータ若しくはデータを提供するデータベースを考慮し、現在のバージョンは最もシンプルな仕様となっている。よく見られる表の形態は、表ヘッダ行に倣い、データを記入する行が並び連なる。金額や数量を対象とした表では、最終行は合計欄となっている。これらの表の形態から、1 行分を作成すると、その行をコピーしていくことで表が出来上がる。今回開発した GUI は表を構成するセル単位での操作を中心としている。しかし表作成の利便性を考慮すると、行単位で操作できる機能が必要となる。これをサポートできる TableML として、複数の textBox 要素を束ねる block 要素を設けて行を定義し、表の繰り返し構造に対応する機能拡張を行っている。TableML に合わせ、表作成 GUI を用いた記述方法並びにデータリンク方法が今後の課題となる。また数値を伴う表では、例えば個数と単価を計算し金額を算出する、合計を算出するなどの簡単な計算機能も必要となる。これに対応するため、データタイプを記述すると共に、計算スクリプトの記述方法も今後必要となろう。

XML 文書で記述された表組みを他のコンテンツに埋め込んだことによって、複合文書は更に多彩な表現力を持つようになる。例えば複合文書中の表をクリックすると、その表のデータから SVG を用いてグラフを表示することが可能となる。このようなインタラクティブで多彩な複合文書を実現するために、XLink によるリンク管理は大変有効である。

参考文献

[大坂、野村] ”SVG-DOM によるアニメーションと XHTML 中心複合文書の可能性”、
情報処理学会研究会報告、FI-66、DD-32、3/2002

[大坂 1] ”JavaScript を用いた XLink の実装方法について”、情報処理学会研究会報告、DD-35、9/2002

[大坂 2] ”XLink データの登録について”、情報処理学会研究会報告、DD-36、11/2002

[CSS]“Cascading Style Sheets”、http://www.w3.org/Style/CSS/

[XSL]“Extensible Stylesheet Language (XSL) Version 1.0”、http://www.w3.org/ Style/XSL/

[森口ら]“XML 文書のスタイルシート生成方式”、情報処理学会研究会報告、DD-17、3/1998