データベースシステム　122－59
(2000. 7. 28)

SuperSQLによるXMLデータドキュメントの自動生成

赤堀 正剛† 田中 隆一‡ 遠山 元道#

†慶應義塾大学大学院 理工学研究科 管理工学専攻　‡慶應義塾大学 理工学部 情報工学科
慶應義塾大学 理工学部 情報工学科／JST さきがけ研究21
神奈川県横浜市港北区日吉 3-14-1 慶應義塾大学情報工学科遠山研究室
045-563-1141 内線 43244
† aka@db.ics.keio.ac.jp, ‡ ryu@db.ics.keio.ac.jp, # toyama@ics.keio.ac.jp

あらまし

本研究では、関係データベースの既存のデータをXMLデータドキュメントとして取り出すためにSuperSQLを利用することを提案する。SuperSQL質問文のGenerate句にあるTFEが複雑な木構造を表現可能であることを利用し、複雑な構造のXMLデータドキュメントを生成する。更に、このデータを視覚化して表示するXSL・構造を示すDTD等も同様に生成し、応用データとしての利便性を高めデータ利用者の支援を行う。

キーワード　SuperSQL 、 XML 、 データベース 、 データベース出版 、 EDI

XML data document generation using SuperSQL

Masatake Akahori † Ryuichi Tanaka ‡ Motomichi Toyama #

†Department of Administration Engineering, Faculty of Science and Technology, Keio University.
‡Department of Information and Computer Science, Faculty of Science and Technology, Keio University.,
Department of Information and Computer Science,
Faculty of Science and Technology, Keio University/JST PRESTO
3-14-1, Hiyoshi Kouhoku-ku Yokohama-shi, Kanagawa-ken
+81-45-563-1141 ext.43244
†aka@db.ics.keio.ac.jp ‡toyama@ics.keio.ac.jp # toyama@ics.keio.ac.jp

Abstract

In this paper, we propose the extension of SuperSQL to extract data from a relational database in the form of XML. TFE, a part of Generate clause in SuperSQL, can express complex tree structure so that we can generate complex XML data documents. Additionally, it generates XSL, which can convert the XML document into HTML, and DTD, which express the structure of the XML documents, to support users of the generated XML data document.

key words　**SuperSQL , XML , Database , Database Publishing , EDI**

1. はじめに

近年、電子商取引や部署間・企業間の連携業務の拡大により電子データ交換 (EDI) の必要性が増大している。既存のデータベースは各々異なったスキーマを持ちデータ互換性は低い為データ交換のための共通規格が必要であり、専用の規格に加え更に多様な構造のデータ交換の必要性が増してきた。そこで自己記述的な XML が注目され、構造を統一してデータ交換を行いつつある。

一方、従来より関係データベースには膨大な量のデータが蓄積されており、これらは交換したい構造で格納されているとは限らなく、適切な構造の XML データに変換する必要がある。そこで自由な構造記述能力等を持ち高機能で汎用的な変換機構が必要となり、本研究では SQL を拡張してデータベース出版機能を持たせた SuperSQL を用いて関係データベースから様々な構造の XML データを生成する。その柔軟な構造指定能力により、同じ関係データベース中のデータから様々な構造の XML データを出力可能であり、逆に異なるスキーマを持つ関係データベースから統一した構造のXMLデータ☆が生成しデータ統合を図ることが可能である。

1.1 関連研究

データベース中のデータを XML 文書として出力する機能は異種情報源統合システム等に存在する。例えば YAT[4] は OODB を情報源に持ち、wrapper を通じてデータを抽出している。InfoWeaver[5] は統合データモデル WebNR/SD に従った入れ子型データモデルに構造化文書やリンク構造を扱う抽象データ型を導入し、情報源の一つとして関係データベースを wrapper を介して使用している。Araneus は ADM(ARANEUS DATA MODEL) に従った特殊な型に格納したデータを XML 文書として出力できる[6][7][8]。一方、奈良先端大で開発中のシステム[9] は関係データベース中にエッジの集合に分解し格納した XML 文書を再度 XML 文書として出力できる。これらに対し、本研究は通常の関係データベースに既存のデータを XML データとして出力する点が異なる。

関係データベース Oracle[10] でも格納されたデータを XML データして取り出せるがその構造は平坦で単純なスキーマであり、SuperSQL の様に複雑な構造を直接取り出せない。Swift 社[11] の XML ServerWare の一コンポーネントである DB-X は質問文の入れ子で複雑な構造の出力を可能としている。一方、SuperSQL は入れ子でない単一の質問文で複雑な構造の指定が可能である。

1.2 本論文の構成

2章では簡単に XML 文書に関して述べる。3章では SuperSQL に関して述べ、それを XML 出力のために拡張した結果を 4章にて述べる。最後に 5章にてまとめる。

2. XML 文書について

XML 文書の構成は大きく、「XML 宣言」「文書型定義 (DTD:Document Type Definition)」「XML インスタンス」の 3 つとなる☆☆。XML 宣言は XML のバージョンと符号化方式を指定する。文書型定義はどの様な構造の XML インスタンスを認めるかという定義であり、文脈自由文法にほぼ相当する。

XML インスタンスは要素が階層構造化されており、各要素は開始タグ (要素型を < と > で囲んだ文字列) と終了タグ (要素型を < / と > で囲んだ文字列) の対に囲まれ区切られている。開始タグと終了タグの間の文字列がその要素の内容である。開始タグと終了タグの対は入れ子とすることが可能であり、これにより階層構造を表現する。又、開始タグでは要素型と > の間に属性 (属性名と属性値を「=」で繋いだもの) を記述する事が可能である。尚、XML の属性とデータベースの属性の意味するところは異なる為、本論文では XML の属性を単に「属性」、データベースの属性を「データベース属性」と呼ぶ。

3. SuperSQL について

データベースで管理されているデータを利用するには、データを取り出し必要な構造へ変換し、応用データを構築する出力媒体・メディア☆☆☆に変換する必要がある。SQL は関係データモデルに基づいており、出力は平坦な表である。応用データへ変換するにはその構造や出力媒体へ変換する為の専用の手続き・アプリケーションが必要となる。

一方、SuperSQL はデータ・構造及びレイアウト・出力媒体の 3 つの要素から成り、SQL のデータ操作機能に加え、出力構造やレイアウト、出力媒体の指定が可能である。(図 1参照)

SuperSQL の質問文は、SQL の SELECT 句を拡張し「 GENERATE < *medium* >< *TFE* >」とい

☆ XML で記述したデータを本論文では XML データと呼ぶ。

☆☆ XML 宣言と DTD は必須ではない。

☆☆☆ HTML、LaTeX、Excel、XML 等

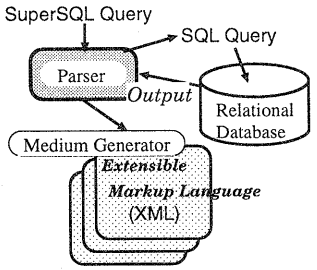

図1　システムの実装

う構文を持つ GENERATE 句で置き換えたものである。ここで $<medium>$ は出力媒体を示し、HTML、Java、Excel、LaTeX などの指定が出来る [2),3)]。$<TFE>$ はターゲットリストの拡張である Target Form Expression (TFE) を表し、データの構造化と結合子、反復子などのレイアウト指定演算子を持つ一種の式である。即ち、反復子 ([ ] の対の後に結合子が続くもの) に囲まれた内容のグループ化と { } によるまとまりの指定が可能であり、これにより図2の様な木構造を表現可能である。

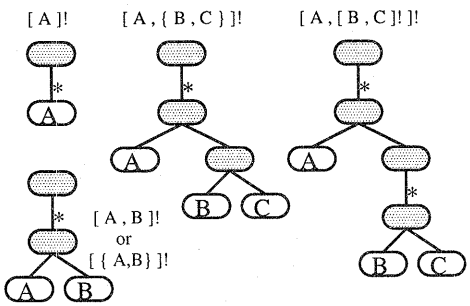

図2　TFE による木構造の表現

これらを用い、SuperSQL は関係データベースから得られる平坦な構造のデータを階層構造を持つデータに変換することが可能である。この階層構造を持つデータを入れ子を持つ表として出力する事より、検索された情報の意味を反映した表示が可能になる。

3.1 次　　元

独立に意味付けされた方向、即ち意味的に直交な成分を表し、構造化されたデータを整列させる際に用いる概念である。例えば水平方向、垂直方向、深度方向、時間軸方向等がある。この次元に対応する「,」「!」「%」「#」等の結合子を使用する。どの次元にどの様な意味を割り当てるかは出力媒体毎に異なり、同じ結合子でも出力媒体により効果が異なることもある。

3.2 結合子と反復子

出力結果のレイアウトを指定する際に使用し、次元に対応した出力を生成する。以下にその例を挙げる。

(1) 結合子 … 前後にあるオペランド、即ちデータベース属性や構造化されたデータベース属性を結合し、そのデータを対応する次元に従い並べて表示する。

- 水平結合子 (,)
  例 : Name, Tel → | $name_1$ | tel_1 |
- 垂直結合子 (!)
  例 : Name! Tel →

$name_1$
tel_1

(2) 反復子 ([ ] の対の後に結合子が続く) … [ ] 内にあるオペランド、即ちデータベース属性や構造化されたデータベース属性をグループ化し、そのデータを存在する限り対応する次元の方向に反復して表示する。

- 水平反復子 ([ ],)
  例 : [Name],
  → | $name_1$ | $name_2$ | … | $name_{10}$ |
- 垂直反復子 ([ ]!)
  例 : [Name, Tel]! →

$name_1$	tel_1
$name_2$	tel_2
…	…
$name_{10}$	tel_{10}

3.3 装　飾　子

各データベース属性や「{ }」、反復子に対するオプションを指定する。「{ }」、反復子に対してオプションを指定すると、原則的にその内側のデータベース属性や「{ }」、反復子に対してそのオプションが適用される☆。オプションの指定は、データベース属性もしくは「{ }」、反復子の後に続く「@{」と「}」の間に記述し、オプションと引数の対を「=」で結合して指定する。複数のオプションを指定する場合は「,」を挟んで並べて記述する。以下にその例を示す。

☆ この様なスコープルールが適用されない一時的オプションもある。

装飾子の指定例

(1) データベース属性 A に対し name オプション (引数は「alpha」) size(引数は「large」) を、[ ]! に対し tag オプション (引数は「beta」) を指定する。

```
[ A@{name=alpha, size=large}
]!@{tag=beta}
```

(2) 以下の指定は共に等しい。

```
[ A@{size=large}, B@{size=large}]!
[ { A, B }@{size=large} ]!
[ A, B ]!@{size=large}
```

3.4 関　　数

TFE 中のデータベース属性、もしくは「{ }」や反復子等で構造化されたデータベース属性に対し適用され、これらを引数として処理した結果を出力する。

4. SuperSQL の XML データ出力について

SuperSQL はデータベースへの問い合わせ結果を構造化し、木構造データへ変換することができる。これを利用しデータベース中に既存のデータから XML データを生成するために表 1の機能を付加する。

表 1　XML データ出力の為の拡張

拡張事項	実現方法	参照
要素の連結	結合子・反復子の「,」「!」	4.1.1
データベース属性の結合	データベースの concatenate 機能「||」	4.1.1
属性	装飾子によるオプション「att」	4.1.3
要素名の命名	装飾子によるオプション「name」「tag」	4.1.3
DTD の「+」の実現	[ ]	4.2.2
DTD の「*」の実現	[ ] + null を「不在」と扱う	4.2.2
DTD の「?」の実現	null を「不在」と扱う	4.2.3
DTD の「|」の実現	結合子「|」	4.2.4
DTD の生成・利用	TFE ↔ DTD の変換	4.4
XML データの視覚化	XSL ファイルの生成 (→ HTML/LaTeX ファイルの生成)	4.5

4.1 拡　　張

4.1.1 結合子 …　「,」「!」「\|」「\|\|」

(1) 「,」「!」 … オペランドとなるデータベース属性もしくは構造化されたデータベース属性を結合する。他の出力媒体では「,」は水平方向の、「!」は垂直方向の次元を表すが、XML データを出力媒体とした場合、水平・垂直に要素を並べて記述しても差異は無く同じ構造のデータを示すために「,」「!」に区別は無い。しかし、後述する出力 XML データの視覚化に於いては水平・垂直方向の次元は区別される。

(2) 「|」 … 「A | B」と TFE に記述した場合、オペランドであるデータベース属性 A 又は B の内、値が存在する、つまり null 値でない方のオペランドを出力とする。両方のオペランドの値が存在した場合、左側に記述されたオペランド、つまりデータベース属性 A の値を優先して出力とする。(詳細は 4.2.4参照)

(3) 「||」 … 正確には SQL の「連結 (concatenate 機能、||)」であり、結合子ではない。前後のオペランドを結合して一つのデータベース属性の様に扱う。ただし属性名が無くなってしまうため、装飾子によるオプション「name」による名前付が必要である。例えばデータベース中に性と名が別々の格納されているが出力としては場合に一つの氏名ととしてまとめて得たい場合に使用する。

4.1.2 反復子 …　「[ ],」「[ ]!」

[ ] の後の結合子は、グループ化されたデータを出力する際に適用する次元、つまりどの様に反復して出力するかを指定している。他の出力媒体では「,」は水平方向、「!」は垂直方向を示すことがある。しかし XML データが出力の場合には水平・垂直方向の区別は意味を持たない。ただし、この区別は 4.5に後述する出力 XML データの視覚化の際に必要である。

4.1.3 装飾子 …　「@ {~}」

データベース属性や { }、反復子に対しその直後の装飾子「@{ }」内にてオプションを指定できる。例えば以下の指定が可能である☆。4.3に例を示す。

(1) name(名前付) … 対象 (データベース属性) の要素名を名付ける。

(2) tag(名前付) … 対象 ({ }、[ ]) の要素名を名付ける。

(3) att(属性付加) … 対象の名前を属性として、対象の値を属性値として引数となる要素へ付加する。

(4) notag(要素型の表示・非表示指定) … 対象の要素型 (タグの対) の表示 (off)・非表示 (on) を指定する。

(5) null … 対象のデータベース属性が null 値をとる場合の動作を不在 (ne) もしくは未知 (unk) として指定する。4.2.1参照。

☆ (1) ~ (2) は一時的なオプションである。

4.1.4 関数 … 「NULL関数」

そのデータベース属性を用いてグループ化等を行いたいが、出力に値が現れて欲しく無い場合に使用する。オペランドを「null(」と「)」で囲んで記述する。図3に例を示す。

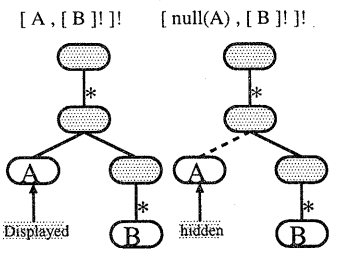

図3 null関数

4.2 Null値について

4.2.1 空値の扱い … 未知と不在

null値にはいくつかの種類がある。例えば「値が不明(未知:unknown)」「属性自体存在しない(不在:non-existent)」等である。XMLデータに対しては、

(1) 「未知」 … 「要素は存在するが値が不明もしくは無い(例: < A/ >)」

(2) 「不在」 … 「要素自体存在しない(タグ自体出力されない)」

という対応になるだろう。要素が存在すれば、そこには何らかの意義が存在する。外結合等の結果データベースからの出力にnull値が存在した場合は元々そのタプルにはその属性自体が存在していなかった事になる為、XMLデータに変換する際には後者の「不在」のタグ自体が出力されない形式の方が自然であろう。よってnull値に対してはデフォルトではタグごと出力しないものとする。ただし、装飾子を用いたオプション指定で前者の「未知」の形式で出力する(例: A@{null=unk}) ことも可能である。

4.2.2 DTDにおける「*」の実現

反復子により、図2の様に木構造を示すことができる。反復子の中のオペランドはデータがある限り反復して出力され、null値を含まない表や通常の結合を利用した場合はオペランドのデータは一つ以上存在するためDTDの「+」に相当する動作を行う。それに対し、外結合などの結果null値を含む表を元にした場合はオペランドのデータは0個以上存在することとなり、null値を「不在」と扱うとデータが0個の場合はにタグごと表示されずDTDの「*」に相当する動作を行う。図6に例を示す。

4.2.3 DTDにおける「?」の実現

DTD中の「?」は「その直前に書かれた要素がたかだか一つ存在する」という意味である。TFEでそのような意味を実現するには、データベース属性が値を持てば表示しnull値であれば「不在」として処理すれば良い。これによりたかだか1つのデータが出力されることとなり、DTDの「?」に相当する動作を行う。図6に例を示す。

4.2.4 DTDにおける「|」の実現

DTDにおける「|」の機能を、null値に対し特殊な解釈を行う特殊次元「|」にて実現する。「A|B」とTFEに記述した場合、オペランドであるデータベース属性A又はBの内、値が存在する、つまりnull値でない方のオペランドを出力とする。両方のオペランドの値が存在した場合、左側に記述されたオペランド、つまりデータベース属性Aの値を優先して出力とする双方の値が共にnull値の場合、左優先であるので左のオペランドのnullオプションに従う。図6に例を示す。

4.3 サンプルデータと生成例

サンプルデータとして電話会社の顧客データを想定した図4と図5のデータを使用する。このデータを使用した生成例を図6と図7と図8に示す。

図4 サンプルデータ: 電話会社

4.4 DTDとTFEについて

XMLデータを利用する際、XMLインスタンスの構造を表す文書型定義(DTD)を必要とする場合がある。そこでSuperSQLにDTDを生成する機能を付加する。尚、TFEにおいて異なるデータベース属性に対し同じ名前付を行っていると正しい変換が行う事ができない。

又、既にDTDが存在し、それに従ったXMLデータを生成する場合にはDTDからTFEのテンプレートを生成し、それをデータベース属性名に編集してTFEおよびSuperSQL質問文を生成する。ただしTFEは木構造に限定されるため、DTDの示す構造は有向非巡回グラフ(DAG:Directed Acyclic Graph)である必要がある☆。

☆ TFEの示す構造はルートが一つのみであるが、ルートが複数のXMLデータを表現可能である(最外の[ ]に名前を付けなければ良い)。

Tel(契約電話番号)

Type	TelNo	Phone	CID
installed	03-XXX-XXXX	–	1
portable	090-XXX-XXXX	N207S	1
installed	044-XXX-XXXX	–	2
portable	090-XXX-XXXX	P601	3
portable	090-XXX-XXXX	F209i	3

Customer(顧客)

ID	Name	Method	CardNo	Account
1	M.A.	Card	012345	–
2	T.O.	Card	234567	–
3	A.M.	Account	–	9876543

Charge(通話料合計)

ID	TEL	Y	M	Payment
1	03-XXX-XXXX	2000	06	3750
1	090-XXX-XXXX	2000	06	7250
2	044-XXX-XXXX	2000	06	7500
3	090-XXX-XXXX	2000	06	4310
3	090-XXX-XXXX	2000	06	3100

図5　サンプルデータ: 電話会社

4.5 XSL ファイルの生成

出力したXMLデータをそのままアプリケーション等で使用する以外に、HTML ファイル等に変換・視覚化して閲覧する場合がある。そこで、XMLデータの視覚化の為に HTML、LaTeX ファイルへ変換するXSL ファイルを生成する機能を付加する。

前述のようにSuperSQL には次元という概念があり、水平方向、垂直方向等を割り当てられた各次元を利用してデータのレイアウトを指定することが可能である。本来ならば出力メディアをXMLとした場合に水平・垂直方向という概念は無用であるが、出力したXMLデータの視覚化においてこの概念は有用である。

SuperSQL では XML 以外にも直接 HTML や LaTeX 等の出力が可能である。XML を変換してこれらの他の媒体の出力を得ることには、

- 元となるデータベースが無くとも変換が可能
- XSL ファイルの生成・変更はSuperSQL 質問文をそのまま利用
- レイアウトを変更する際のコストはXSL ファイルを更新のみで出力データの更新は不要
- SuperSQL による出力生成時とXMLデータの変換時にコストを分散
- データ構造は等しいがレイアウトのみが異なる表示を複数必要とする場合などは、元となるXMLデータと複数のXSL ファイルを用意すれば良い。
- 同じ構造で複数の出力媒体のデータを必要とし、元のデータベースと応用データの利用場所が異なる場合、移動コストを省略可能

等の特徴がある。この為、ローカルで応用データを利用する場合や巨大な応用データを使用する場合、又は同じデータ構造のデータを異なるレイアウトで表示する場合などに有効である。

例えば、図7の生成例に対しHTML の表へ変換するXSL ファイルとその結果を図10に示す。

SuperSQL の質問文

```
GENERATE XML
[ { C.Name,
    { C.CardNo | C.Account },
    [ T.Phone ]!
  }@\{tag=Customer}
]!@{tag=Customers}
FROM Customer C, Tel T
WHERE C.ID = T.CID
```

データベースからの出力

Name	CardNo	Account	Phone
M.A.	012345	–	–
M.A.	012345	–	N207S
T.O.	234567	–	–
A.M.	–	9876543	P601
A.M.	–	9876543	F209i

DTD

```
<!ELEMENT Customers (Customer)+>
<!ELEMENT Customer
  (Name, (CardNo|Account), ()Phone)*)>
```

出力 XML データ

```
<Customers>
 <Customer>
  <Name>M.A.</Name>
  <CardNo>012345</CardNo>
  <Phone>N207S</Phone>
 </Customer>
 <Customer>
  <Name>T.O.</Name>
  <CardNo>234567</CardNo>
 </Customer>
 <Customer>
  <Name>A.M.</Name>
  <Account>9876543</Account>
  <Phone>P601</Phone>
  <Phone>F209i</Phone>
 </Customer>
</Customers>
```

図6　生成例 A

5. まとめと今後の課題

SQL を拡張しデータベース出版機能を持たせたSuperSQL を用い、関係データベースから構造化された

SuperSQL の質問文

```
GENERATE XML
   [ {C.Name,
     [ {T.Type, C.Tel, CH.Payment
       }@{tag=Phone}
     ]!@{tag=Phones}
   }@{tag=Customer} ]!@{tag=Customers}
FROM Customer C, Tel T, Charge CH
WHERE C.ID=T.CID and T.TelNo=CH.TEL
```

DTD

```
<!ELEMENT Customers (Customer)+>
<!ELEMENT Customer  (Name, Phones)>
<!ELEMENT Phones    (Phone)+>
<!ELEMENT Phone   (Type, Tel, Payment)>
```

出力 XML データ

```
<?xml version="1.0"?>
<Customers>
 <Customer> <Name>M.A.</Name>
  <Phones> <Phone>
    <Type>  installed</Type>
    <Tel>03-XXX-XXXX</Tel>
    <Payment>3750</Payment>
  </Phone> <Phone>
    <Type>  portable</Type>
    <Tel>090-XXX-XXXX</Tel>
    <Payment>7250</Payment>
  </Phone> </Phones>
 </Customer>
 <Customer> <Name>T.O.</Name>
  <Phones> <Phone>
    <Type>  installed</Type>
    <Tel>044-XXX-XXXX</Tel>
    <Payment>7500</Payment>
  </Phone> </Phones>
 </Customer>
     .....
</Customers>
```

図 7　生成例 B

SuperSQL の質問文

```
GENERATE XML
   [{T.Type,
     [C.Tel@{name=Phone},
        C.Name@{name=owner,att=Phone}
     ]!@{tag=Phones}
     }@{tag=Telephone}
   ]!@{tag=Telephones}
FROM Customer C, Tel T
WHERE C.ID=T.CID
```

DTD

```
<!ELEMENT Telephones (Telephone)+>
<!ELEMENT Telephone  (Type, Phones)>
<!ELEMENT Phones     (Phone)+>
<!ELEMENT Phone      (Tel)>
<!ATTLIST Phone  owner CDATA #REQUIRED>
```

出力 XML データ

```
<?xml version="1.0"?>
<Telephones>
 <Telephone>
  <Type>  installed</Type>
  <Phones>
   <Phone owner="M.A.">
   03-XXX-XXXX</Phone>
   <Phone owner="T.O.">
   044-XXX-XXXX</Phone>
  </Phones>
 </Telephone> <Telephone>
  <Type>  portable</Type>
  <Phones>
   <Phone owner="M.A.">
   090-XXX-XXXX</Phone>
     .....
  </Phones>
 </Telephone>
</Telephones>
```

図 8　生成例 C

XML データを生成した。今後は、構造に関する指定の充実やユーザ定義関数などを用いてより自由な出力を得られる工夫を行う。また、大量の出力構造化データに対する差分更新等の課題がある。

謝辞　本研究の一部は情報処理振興事業協会 (IPA) の高度情報化支援ソフトウェアシーズ育成事業の補助による。

参 考 文 献

1) T.Bray, J.Paoli, C.M.Sperberg-McQueen. Extensible markup Language (XML) 1.0 W3C Recommendation, Feb, 1998. http://www.w3.org/TR/REC-xml
2) SuperSQL http://ssql.db.ics.keio.ac.jp
3) M.Toyama, SuperSQL: An Extended SQL for Database Publishing and Presentation, in *Proc.SIGMOD'98*, ACM(1998), 584-586
4) Vassilis Christophides, Sophie Cluet, JeromeSimeon, On Wrapping Query Languages and

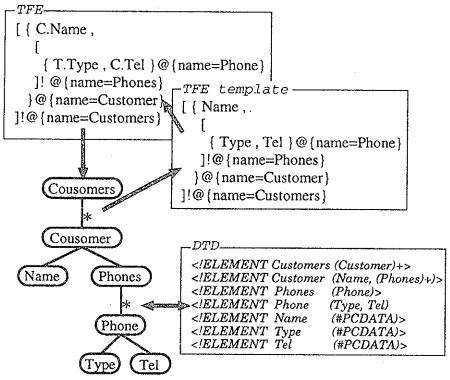

図9　TFE と DTD の変換

Efficient XML Integration, SIGMOD/PODS 2000, Dallas, Texas, May 14-19, 2000

5) "InfoWeaver: Dynamic and Tailor-Made Integration of Structured Documents, Web, and Databases," Proc. ACM Digital Libraries '99, August 1999, pp. 235-236.

6) Araneus Project, http://www.dia.uniroma3.it/Araneus

7) Giansalvatore Mecca, Paolo Merialdo, Paolo Atzeni, Araneus in the Era of XML, IEEE Data Engineering Bullettin, Special Issue on XML, September, 1999

8) ACM SIGMOD Record:XML version http://www.acm.org/sigs/sigmod/record/xml

9) 吉川 正俊, 志村 壮是, 植村 俊亮, オブジェクト関係データベースを用いたXML文書の格納と検索, 情報処理学会論文誌, データベース, Vol.40, No.SIG6(TOD3), pp115-131, 1999年8月

10) Oracle, http://www.oracle.com

11) Swift 社, http://www.swiftinc.co.jp/

12) 田島 敬史, 半構造データのためのデータモデルと操作言語, 情報処理学会論文誌 Vol.40 No.SIG3(TOD 1), pp.152-170, 1998年2月

XSL ファイル

```
<?xml version="1.0"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/TR/WD-xsl">
 <xsl:template match="/">
  <HTML>
   <HEAD><TITLE></TITLE></HEAD>
   <BODY> <xsl:apply-templates/> </BODY>
  </HTML>
 </xsl:template>
 <xsl:template match="Customers">
  <TABLE BORDER="2">
   <xsl:apply-templates/>
  </TABLE>
 </xsl:template>
 <xsl:template match="Customer">
  <TR> <xsl:apply-templates/> </TR>
 </xsl:template>
 <xsl:template match="Name">
  <TD> <xsl:value-of select="." /> </TD>
 </xsl:template>
 <xsl:template match="Phones">
  <TD> <TABLE BORDER="2">
       <xsl:apply-templates/>
  </TABLE> </TD>
 </xsl:template>
 <xsl:template match="Phone">
  <TR> <xsl:apply-templates/> </TR>
 </xsl:template>
 <xsl:template match="Type">
  <TD> <xsl:value-of select="." /> </TD>
 </xsl:template>
 <xsl:template match="Tel">
  <TD> <xsl:value-of select="." /> </TD>
 </xsl:template>
 <xsl:template match="Payment">
  <TD> <xsl:value-of select="." /> </TD>
 </xsl:template>
</xsl:stylesheet>
```

結果

M.A.	installed	03-XXX-XXXX	3750
	portable	090-XXX-XXXX	7250
T.O.	installed	044-XXX-XXXX	7500
...	...		

図10　視覚化:XML から HTML への変換