社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2005－DBS－137 (II) (68)
2005／7／14

XMLデータの整形出力処理系PPXにおけるDTDの利用

金 哲† 慎 祥揆† 有澤 達也‡ 遠山 元道

†慶應義塾大学大学院理工学研究科開放環境科学専攻
‡慶應義塾大学デジタルメディア・コンテンツ統合研究機構
慶應義塾大学 理工学部 情報工学科
E-mail: †{tetsu, shin}@db.ics.keio.ac.jp, ‡ari@db.ics.keio.ac.jp, toyama@ics.keio.ac.jp

XMLデータをHTMLに変換する既存の方法は汎用プログラム言語JAVA,PERL,PHP,C++などか、或いはスタイルシート言語XSLなどがある。本論文では、PPX処理系においてDTDを参考にレギュラーXMLデータをHTMLに変換する方法に基づき、イレギュラーXMLデータもルール演算子によってHTMLに変換するルールを提案する。また、実験により提案するルールのその有効性を示す。

キーワード：XML, レギュラーXMLデータ, イレギュラーXMLデータ

Application of DTD to the Formatting Output Processing System PPX of XML Data

Zhe JIN† Sang-Gyu SHIN† Tatsuya ARISAWA‡ Motomichi TOYAMA

†School of Science for OPEN and Environmental Systems, Faculty of Science and Technology,
Keio University.
Research Institute for Digital Midia and content,
Keio University.
Department of Information and Computer Science, Faculty of Science and Technology,
Keio University.
E-mail : †{tetsu, shin}@db.ics.keio.ac.jp, ‡ari@db.ics.keio.ac.jp, toyama@ics.keio.ac.jp

With the advent of XML as a standard for data representation and exchange on the Internet, the storying and querying of XML data become more and more important, as does the transformation of XML data into HTML. Several methods of transforming XML data into HTML have been proposed, e.g., XSL, JAVA, C++, PHP, and many more. In this paper, we propose a new method that formats irregular data into HTML in the PPX processing system referring to DTD automatically, and verify this proposed method by simulation.

keyword :XML, regular XML data, irregular XML data

1 はじめに

現在、XMLで記述されたデータが大量に存在し、半構造データを表現、データの交換、情報の記述・処理などに利用されている。これに従って、XMLデータをHTMLに変換する技術も必要になる。本論文では、XMLデータ文書をHTMLに変換する方法を主に三つに分けられる。

1.1 汎用プログラム言語を利用

XMLデータをHTMLに変換する汎用プログラム言語としてJAVA,PERL,PHP,C++などを挙げられる。汎用プログラム言語を利用する場合、XMLパーサは、XMLデータ文書をメモリ上に要素のツリーを作成するか、あるいは新しいドキュメントオブジェクトを作ってDOM,SAXなどのインターフェスを使ってXMLデータをHTMLに変換する。

1.2 スタイルシート言語を利用

スタイルシート言語XSLを利用する場合、XSLTプロセッサはスタイルシートの記述に従ってXMLデータをHTMLなどの変換を実現することができる。Oracle社のXSQL[1]はXSQLサーブレットがJDBC接続でデータベースに問い合わせをし、データベースからデータをXMLデータとして取得し、その結果をXSQLサーブレットに返す。XSQLサーブレットがデータベースから出力されたXML文書をXSLで表記されたスタイルシートによってHTMLなどに変換する。インフォテリア社のiXSLT[2]は変換元となるXML文書と、XSLT規格に則ったXSLTスタイルシートを読み込み、XSLTスタイルシートに記述されたルールに従ってHTMLなどに変換する。そのほかにもXSLを利用したW3C組織のXQuery、Vector社のXtoH[3]など、geneva大学のJAVAに基づいたXML変換言語、XMLTrans[4]などがある。

1.3 TFEを利用

上で述べたようにXSL,JAVAなどは様々なところで利用されているが、このような方法は一般ユーザに対して難しいし、作成する負担も大きい。それらに対してXSLやJAVAなどを利用せずに、TFEを利用してXMLデータをHTMLに変換するPPX処理系[6]を提案した。TFEはターゲットリストの拡張であるTarget Form Expressionを表し，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。TFEを利用するSuperSQL[5]は関係データベースの出力結果を構造化しHTML，XMLなどの媒体への出力を可能とするSQLの拡張言語であり，慶應義塾大学遠山研究室で開発されている。しかし、半構造のXMLデータを探索することもできないし、レイアウトすることもできない。また、PPX処理系はレギュラーXMLデータに限り、HTMLにレイアウトすることができた。

本研究では、PPX処理系において、レギュラーデータの固定フォーマットに基づいて、DTDを参考にイレギュラーデータも自動フォーマットするルールを提案する。

2 XMLデータの定義

2.1 PPX処理系概要

PPXは、直接にXMLデータをHTMLに変換できるクエリ言語であり、媒体指定、レイアウト表現式を指定するGenerate句とXPath式を指定するFor句から構成される。

2.1.1 GENERATE句

Generate句では二種類のレイアウト表現がある。(1)、レギュラーデータのレイアウト: Generate句の表現式では、レイアウトする属性を完全な絶対パスで一つ一つ指定して、固定フォーマットする。(2)、イレギュラーデータのレイアウト: Generate句の表現式では、レイアウトするある属性に対してルール演算子を利用して自動的にフォーマットする。

2.1.2 For句

For句では、Xpath式によってXMLデータを探索する。(1)、完全な絶対パスを利用した探索、即ちレギュラーデータの探索

For papers/paper/authors/author/name/text() (2)、不完全な絶対パスを利用した探索、即ち、イレギュラーデータの探索　For /papers

2.2 レギュラーXMLデータとイレギュラーXMLデータ

XMLのDTDとそれに対する一部分のXML文書によるパス集合関係は図1、図2のとおりである。ルールを述べる前に、レギュラーデータとイレギュラーデータを定義する。次のクエリから幾つのパス集合が生成する。

GENERATE html　[journalname, & paper],
FOR　　/papers/journalname/text()
　　/paper

パス集合には、DTDの絶対パス集合: P(D)、XML文書の絶対パス集合: P(X)、そしてクエリ絶対パス集合: P(Q)がある。P(Q)には完全な絶対パス集合: P(Qc)と不完全な絶対パス集合: P(Qi)の集合がある。このようなパスの集合から次のような定義がある。

定義1: レギュラーパス全集:　XMLデータ文書パス集合とクエリパス集合の共通部分である。

$$R = P(X) \cap P(Q)$$

イレギュラーパス全集:　XMLデータ文書パス集合からクエリパス集合を除いた部分である。

$$I = P(X) - P(Q)$$

妥当なXMLデータ文書:　XMLデータ文書パス集合がXMLのスキマパス集合に完全に含まれた部分である。

$$P(X) \subseteq P(D)$$

非妥当なXMLデータ文書(整形式XMLデータ文書):　XMLデータ文書パス集合がXMLのスキマパス集合に完全に含まれてない部分である。

$$P(X) \subset P(D)$$

定義2:tはP(Qc)に存在、かつtはPの完全なパスPがXMLデータ文書パス集合とクエリパス集合の共通部分にある時、

$$\{P | P \in P(X) \cap P(Q) \wedge \exists t \in P(Qc),$$

t is complete path of P, complete(t, p)}

Pの集まりが完全なレギュラーパス集合であり、レギュラーデータを持っている。例えば、/papers/journalnameようなパスは完全なレギュラーパスである。

定義3:tはP(Qi)に存在、かつtはPの不完全なパスPがXMLデータ文書パス集合とクエリパス集合の共通部分にある時、

$$\{P | P \in P(X) \cap P(Q) \wedge \exists t \in P(Qi),$$

t is incomplete path of P, incomplete(t, p)}

Pの集まりが不完全なレギュラーパス集合であり、データは持っていない。即ち、イレギュラーパス集合を持っているし、イレギュラーデータを持っている。例えば、/papersようにパスは不完全なレギュラーパスである。

定義4:tはP(Qi)に存在、かつtはPのprefixであるPがXMLデータ文書パス集合からクエリパス集合を除いた部分にある時、

$$\{P | P \in (P(X) - P(Q)) \wedge \exists t \in P(Qi),$$

t is prefix of P, prefix(t, p)}

Pの集まりがイレギュラーパス集合であり、イレギュラーデータを持っている。例えば、/papersようなパスはイレギュラーパスである。

定義5:tはP(Qi)に存在、かつtはPのprefixであるパスPがXMLデータ文書のパス集合からクエリパス集合を除いた部分とXMLのスキマパス集合の共通部分にある時、

$$\{P | P \in (P(X) - P(Q)) \cap P(D) \wedge \exists t \in P(Qi),$$

t is prefix of P, prefix(t, p)}

Pの集まりが想定内イレギュラーパス集合であり、想定内イレギュラーデータを持っている。例えば、/papersと同じprefixを持つし、DTDの中に含むパスである。

定義6:tはP(Qi)に存在、かつtはPのprefixであるパスPがXMLデータ文書のパス集合からクエリパス集合P(Q)除いた部分からXMLのスキマパス集合を除いた部分にある時、

$$\{P | P \in (P(X) - P(Q)) - P(D) \wedge \exists t \in P(Qi),$$

t is prefix of P, prefix(t, p)}

Pの集まりが想定外イレギュラーパス集合であり、想定外イレギュラーデータを持っている。例えば、/papersと同じprefixを持つが、DTDの中に含まないパスである。

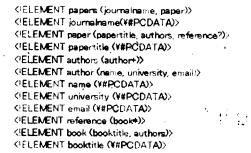


図 1: サンプル DTD

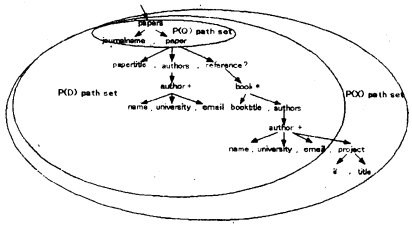


図 2: パス集合関係図

3 提案するルール

3.1 ルールセット

PPX 処理系は、&_演算子などのルール演算子があり、演算子には幾つのルールセットを持つ。ルールセットには、DTD の接続演算子セット、DTD の接尾演算子セットがある。

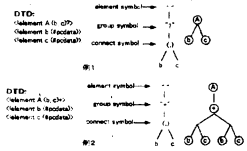
3.1.1 DTD の接続演算子セット

DTD の接続演算子セットには、","、"&"、"|" があり、図3は、接続演算子セット表である。結合するノードの間が","の時は、必ず順序を守りながら結合する。結合するノードの間が" "の時は、任意の順序で結合しても良い。結合するノードの間が"|"の時は、何れかを結合によって二種類の結合方法がある、即ち、縦棒の左側のノードと結合、或いは縦棒の右側のノードと結合など二つである。それらの結合優先順位は","が一番強し、次に"&"であり、一番弱いのが"
|"である。

3.1.2 DTD の接尾演算子セット

DTD の接尾演算子セットには、""、"？"、"＊"、"＋"などのシンボルがある。""は要素が一回だけ現れる。"？"は要素が現れないか、一回現れる。"＊"は現れないか、任意回出現である。"＋"は要素が一回以上出現である。このような規則から、""は、"?"より優先的に処理する。また、"＋"も"＊"より優先的に処理する。次は四種類のシンボルのお互いの結合する時について説明する。

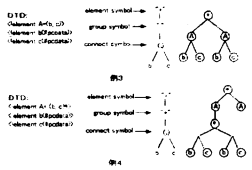
例 1：ある DTD のエレメントシンボルとグループシンボルが両方とも""或いは"？"の場合は、



一つの親ノード A の子供ノード b と c はグループで一回現れるので、それらが持つデータは一回結合される。パーズされた DTD の部分は (,)""""であり、(,)""ようにシンボルは省略できる。そのデータの部分は (b c) である。例 2：ある DTD のエレメントシンボルが""或いは"？"、グループシンボルが"＊"或いは"＋"の場合、一つの親ノード A に子供ノード b と c はグループで一回以上現れるので、それらが持つデータは二回結合される。パーズされた DTD の部分は (,)"+""""であり、(,)"+"ようにシンボルは省略できる。そのデータの部分は ((b1 c1)(b2 c2)…) である。

例 3：ある DTD のエレメントシンボルが"＊"或いは"＋"、グループシンボルが""或いは"？"の場合、親ノード A は一回以上現れるし、一つの親ノード A の子供ノード b と c はグループで一回現れるので、三回結合される。パーズされた DTD の部分は ((,)"")"+"であり、シンボルは省略できない。そのデータの部分は (((b1 c1)) ((b2 c2)) …) …) である。

例 4：ある DTD のエレメントシンボルとグループシンボルが両方とも"＊"或いは"＋"の場合、親ノード A は一回以上現れるし、一つの親ノード A の子供ノード b と c はグループで一回以上現れるので、三回結合される。パーズされた DTD の部分



は((,)"+")"+"であり、シンボルは省略できない。そのデータの部分は((((b1 c1)(b2 c2)(b3 c3)…))(((b1 c1)(b2 c2)(b3 c3)…))…)である。図3は、接尾演算子セット表である。

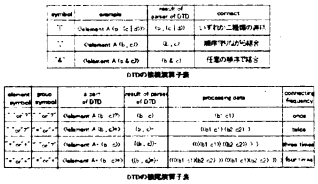


図 3: DTD の演算子セット表

3.2 ルール

本研究で提案するルールには、DTD を参考にイレギュラーデータを自動的にレイアウトする&_演算子を利用する。&_演算子はVHRC(Vertical−Horizontal Repeat Connection) ルールと HVRC(Horizontal−*VerticalRepeatConnection*) ルールの二つのルールを持っている。

3.2.1 VHRC ルール

VHRC ルールは、DTD の情報からイレギュラーデータを持つ個々の階層に対して縦連結、横連結を繰り返しながら反復する TFE を生成する。このようなデータを並べる方式は三角形を作り出すような形である。VHRC ルールを使う基準は、Generate 句の反復演算子とパーズされた DTD の一番外側の接尾演算子によって決まる。一つは、Generate 句の水平反復演算子が"[],"で、パーズされた DTD の一番外側の接尾演算子が""或いは"?"の場合、もう一つは Generate 句の垂直反復演算子が"[]!"で、パーズされた DTD の一番外側の接尾演算子が"*"或いは"+"の場合である。しかし、"*"或いは"+"によって、反復するノード数が予測できないぐらい発生するので、連結方法もそれによって変えなければならない。例えば、

最初、Generate 句の水平反復演算子が"[],"で、""或いは"?"だけあるパーズされた DTD「(,(,))」場合、データを横に並べすぎを避けるため、先に縦に結合、後は横に結合する縦横連結を繰り返しながら反復する VHRC ルールを適用する。

もし、パーズされた DTD のなかに"|"があれば、その度に、二種類の $TFE*$ を生成しなければならない。例えば、パーズされた DTD「(,((,)|(,,)))」の場合は、「(,(,))」に対する $TFE*$ と「(,(,,))」に対する $TFE*$ を個別に生成する。

しかし、"*"或いは"+"などもパーズされた DTD「(,(,)+)」にある場合は、単純に縦横連結を繰り返しながら反復する VHRC ルールで生成した TFE*でレイアウトすることが出来ないので、最適化する必要がある。即ち、横、或いは縦に並べすぎのを避けるし、元の構造の意味を失わないように連結方法を換わる。

Generate 句の演算子"[]!"が上のようなパーズされた DTD に当たる場合についてはここで説明は省略する。

3.2.2 HVRC ルール

HVRC ルールは、VHRC ルールとは逆に個々の階層に対しして横連結、縦連結を繰り返しながら反復する TFE*を生成する。このようなデータを並べる方式は VHRC とは逆の三角形を作り出すような形である。HVRC ルールを使う基準も VHRC ルールように、一つは Generate 句の水平反復演算子"[],"とパーズされた DTD の一番外側の接尾演算子""或いは"?"が結合される場合、もう一つは Generate 句の垂直反復演算子"[]!"とパーズされた DTD の一番外側の接尾演算子"*"或いは"+"が結合される場合、HVRC ルールでレイアウトする TFE*を生成する。HVRC ルールは VHRC ルールと同じルールセットを持っているので説明を省略する。

ルールでは、また様々な場合の DTD 演算子を対

処するために次のような最適化が必要である。

4 TFE*の最適化

最適化規則

入力：目的DTD式　　**出力**：TFE*

規則1：クエリの反復演算子とパーズされたDTDの一番外側の接尾演算子によってVHRCルールか、或いはHVRCルールを適用する。そして、目的式であるパーズされたDTDに対して規則2、規則3、規則4を適用して最適化する。

規則2：目的式に"|"がある時は、同じ階層の"|"対して二種類の$TFE*$を生成する。

規則3：目的式に“　”、或いは“？”だけある場合は、VHRCルールか、或いはHVRCルールそのまま適用する。

規則4：目的式のグループのシンボルに“＊“、或いは“＋”がある場合は、グループが持つ属性の連結の方法はクエリの反復演算子によって横、或いは縦に並べすぎを避けるため、結合方法を変換しなければならない。

規則5：目的式の属性のシンボルに“＊“、或いは“＋”がある場合は、個々の属性が持つvalueの数とストリングのサイズによって結合方法を変わらなければならない。

4.1 ルールによるTFE*の生成

上に述べた最適化の規則によってクエリ(2章の2．2節のクエリ)で、paper属性が持つルール演算子はどのようにイレギュラーデータを自動的に処理するTFE*の生成を図9で示す。ここでは、Generate句の反復演算子が","、パーズされたDTDの一番外側の接尾演算子がつ"なので、VHRCルール適用する。先ず、最初の階層は縦連結するから始まる。そして、次に階層には横連結になるけど、"+"と"*"があるので、連結方法を換わらなければならない。この階層では、現れないか、任意回現れる"*"より一回以上現れる"+"が優先なので"+"によって、横に並べ過ぎのを避けるため、縦横連結が行う。もし、同じ階層に"*"が一つしかない時は連結方法を換わらなくても結構である。しかし、"*"が二つがある時は、縦に並べ過ぎのを避けるため連結方法を換わる必要がある。最後の階層ではまた"+"があるので縦横連結ではなくて、横縦連結にする。

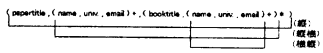


図4: TFE*生成

このように生成したTFE*:「縦縦横横縦」を利用してイレギュラーデータを自動的に処理する。

4.2 データ処理ステップ

第一歩：先ず、パス/papers/jounalnameと/papers/paperをマッチする。この二つパスはP(Q)の集合にあるのでレギュラーパス集合である。即ち、レギュラーデータを持つ。

レイアウト1:

クエリQが完全なレギュラーパス集合を持つ。即ち、レギュラーデータは、TFEによって固定フォーマットする。

第二歩：パス/papers/paperは不完全なレギュラーパスなのでイレギュラーパス集合を持つ。イレギュラーパス集合が持つデータを自動的にレイアウトするためDTDから構造の情報を得る。

レイアウト2:

クエリQがイレギュラーパス集合を持つ。即ち、イレギュラーデータはルールによって自動フォーマット(TFE)によってレイアウトする。

第三歩：XML文書はDTDに合う部分もあるし、DTDに合わない部分もある。それによってP(D)の集合にあるパスはDTDから得た情報によってレイアウトする。

レイアウト3:

クエリQが想定内イレギュラーパス集合を持つ。即ち、想定内イレギュラーデータは、TFE*によってフォーマットする。

第四歩：P(D)の集合にないパスはDTDからレイアウトする情報を得られない。ここで、パス/papers/.../author/project/idとパス/papers/.../author/project/titleは想定外イレギュラーパスである。

レイアウト4：

クエリQが想定外イレギュラーパス集合を持つ。

即ち、想定外イレギュラーデータは、XListによってフォーマットする。 データの定義とデータのレイアウト方法による、クエリの書き方とデータ性質の関係表は図１０のとおりである。

データ性質 / クエリ書き方	regular data part R(x) ∩ R(Q)	irregular data part R(x) − R(Q)	
	valid	valid (想定内)	invalid (想定外)
complete path exp in query (Qc)	固定(TFE) フォーマット	X	X
incomplete path exp in query (Qi)	X	自動(TFE*) フォーマット	自動(XLis) フォーマット

図 5: クエリ書き方とデータ性質

5 実験・評価・検討

5.1 実験

本章では、実験によって本ルールでレイアウトする結果の評価を行う。実験用データはDBLP[7], Shakespeare[8],IMDB[9] などのデータを利用した。実験するXMLデータでは、主にXMLドキュメントのサイズ、階層の深さ、データの数とストリングのサイズなどを考えた。また、クエリパス式では、短い不完全な絶対パス (SI)、長い不完全な絶対パス (LI)、複合な不完全な絶対パス (CI)、そして完全な絶対パス (CC) などを利用した。DBLP データはデータの平均サイズが短いが、深い階層構造を持つ。Shakespeare データは平均サイズも大きく、深い階層構造を持つ。また、IMDB データは平均サイズも短く、階層構造も深くないなどの特徴を持っている。ルールによってレイアウトされたHTMLを (1:見にくい、2:普通、3:見やすい、4:非常に見やすい)4段評価基準とする。

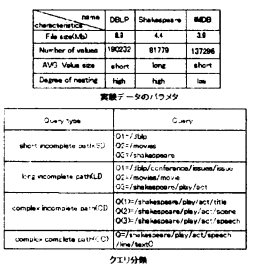


図 6: 実験データとクエリ

5.2 評価

実験では、XMLデータに対するPPXクエリ文の作成難易さ、全データの表示数、レイアウトの程度などについて考察した。図１３ (1) では、SIとLIを平均サイズが短く、深い階層構造のDBLPデータに使った場合、パスが長い方が短いほうよりそのレイアウトが良い。しかし、図１３ (2) では、平均サイズも短く、階層構造も深くないIMDBデータではレイアウトがほぼ同じになる。また、図１３ (3) では、平均サイズも大きく、深い階層構造を持つShakespeareデータではLI(Q3)はSI(Q3)より少し良いのが分かる。複数のパス式を増やしてルールを使うCIの方はレイアウトがもっと見やすくなるのが分かる。

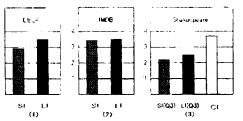


図 7: レイアウト結果

図１４では、不完全なパスを利用する場合は階層が深ければ深いほどそのパスの表示できるデータは少ない。一番深いの階層 (CC) では、表示できるデータ数は一個しかない。また、第一階層ではパスSI(Q3)によって表示されたデータ数は81779個を超える。この超える部分はDTDに定義されてないノードが持つデータであり、XLisで表示する。

良く使われている既存の技術のXSLとは次のような点で異なることが分かる。1．構造が複雑： 構造が複雑なXMLデータに対してXSL,JAVAなどは作成する言語が長くなり、複雑である。しかし、PPXの場合は数行で完成することができるので、一般ユーザでも簡単に利用できる。2．デザインHTML表： HTMLをブラウザに表示するため、XSLはスタイル言語を利用し、HTMLをCSSによって装飾する。しかし、PPXはクエリ言語を利

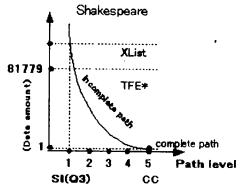


図 8: レイアウト可能データ数

用し、装飾演算子で HTML を装飾する。3．多様なレイアウト表現： XSL はスタイル言語を書き直さなければならない。しかし、PPX は属性位置を変えるたけで、多様なレイアウトの表現が出来る。4．イレギュラーデータの表示： XSL はレイアウトできないイレギュラーデータをそのまま出力する。しかし、PPX はルールによってイレギュラーデータもレギュラーデータに基づいてレイアウトされる。

5.3 検討

PPX クエリは次のようなことを検討にする必要がある。1．完全な絶対パス式で探索：完全なパス式の位置変換によって様々なレイアウトできる、しかし、出力するデータはパスによって一つ一つ指定しなければならないのでパス式の冗長になる。2．不完全な絶対パス式で探索：不完全なパス式は書き方は簡単である。しかし、それが持つサブ木の構造は分からないので、DTD から対応する少しの情報を得てから自動的にレイアウトするので、レイアウトが単純である。

6 終わりに

6.1 まとめ

本論文では、PPX 処理系において&_演算子を利用して DTD を参考にイレギュラーデータをレイアウトする TFE*を生成し、レギュラーデータの固定フォーマットに基づいてイレギュラーデータも自動的にフォーマットするルールを提案した。また、様々なデータに対する実験をすることにより、ルールで自動フォーマットできることを確認した。

6.2 今後の課題

本研究で利用した&_演算子は DTD の演算子だけを利用し、XML ドキュメントのサイズ、構造の複雑さ、およびデータの量、データのストリングのサイズなどの情報はまったく考慮利してなかったので、それらも考えた新たな演算子を利用して TFE* を生成することについても考えていく予定である。また、HTML のレイアウト生成する速度をアップするため相対パスの利用を検討中である。

参考文献

[1] XSQL:http://xsql.sourceforge.net/index.php

[2] iXSLT:http://www.infotcria.com/jp/ product/ixslt/index.jsp

[3] XtoH:http://www.vector.co.jp/soft/win95/net/sc197205.html

[4] Derek Walker and Dominique Petitpierre and Susan Armstrong:"XMLTrans: a Java-based XML Transformation Language for Structured Data"

[5] Motomichi Toyama: "Super SQL: An extended SQL for Database Publishing and Presentation", ACM SIGMOD '98, pp.584-586, 1998.

[6] 金哲、慎 祥揆、有澤達也、遠山元道: "XML データの整形出力処理系"、IEICE Technical Report, Vol.103 No.190, pp.37-41, Jul. 2003.

[7] Michael Ley. DBLP database web site. http://www.informatik.uni-trier.de/ ley/db. 2000.

[8] Jon Bosak. Complete Plays of Shakespeare in XML. http://www.ibiblio.org/xml/examples/shakespeare. 1999.

[9] The Internet Movie Database. http://www.imdb.com. 2000.