データベースシステム 117－6
（1999．1．22）

半構造化データモデルに基づく XML 文書の格納と検索及びその実装方法

北野 拓哉　波内 みさ

NEC C&C メディア研究所

〒216-8555 川崎市宮前区宮崎 4-1-1

{kitano, nami}@ccm.cl.nec.co.jp

概要

XML 文書の文書要素をツリーモデルに基づいて管理する方法を提案する．タグによって識別される XML 文書中の各要素は，1 つの親要素と 0 以上の子要素を持つツリーの関係を構成する．本稿ではこれら要素間の関係を表すインデックスを作成して各 XML 文書ごとに保持させ，要素単位での文書内容の取得・検索を可能にする．本稿の問合せモデルでは General Path Expression を採用する．DTD の違いにより要素の論理構造が異なる複数の XML 文書に対しても，General Path Expression を用いて要素間の関係に関する検索条件を曖昧に指定することにより，各 XML 文書の構造の差異を吸収した検索を実現する．曖昧性を持たせることにより劣化する検索性能も，要素探索経路の最適化によって改善する．

An XML Storage and Retrieval Method based on a Semistructured Data Model

Takuya Kitano　Misa Namiuchi

C&C Media Research Laboratories, NEC Corporation

1-1, Miyazaki 4-chome, Miyamae-ku, Kawasaki 216-8555, Japan

Abstract

We propose a method of XML document management based on a tree model. Each XML element in a document marked by a tag has at most one parent element and more than zero elements. A document structure is made up of its elements like a tree. In our method, an index is created, based on the tree structure for each XML document, and each index is stored with the document. Users can retrieve elements in XML documents by the index. We adopt a general path expression in our query model. Users give an ambiguous retrieval condition for the document structure with the general path expression. The general path expression absorbs the difference of the structure, even though DTDs for the documents are various. Generally, the performance of retrieval including such an ambiguous query expression is not so high. In this paper, the performance decline is improved by an optimization for element search path.

1 はじめに

WWW 上の新しい文書データ交換形式として XML が注目されている．XML はポスト HTML として WWW 上を流通する電子ドキュメントとして利用されるだけでなく，SGML の DTD の利点を引き継ぎ，独自のタグ付けとその意味解釈によってデータベースやアプリケーション間のデータ交換を行う共通のデータ交換形式としても有効に利用される[1]．よって XML 文書は，文書中のタグによって識別される文書要素の扱いが重要と言える．

本稿では，ユーザ独自の DTD を伴って WWW 上を流通する XML 文書を管理するための必須の機能要件は，以下の 2 点と捉えている．

1. 文書要素単位での文書の管理が可能なこと
2. 任意の文書構造を持つ XML 文書を管理可能なこと

1. に関しては，W3C の XML に関するいくつかの仕様書でも要求されている．XML のリンク機能の仕様の一部で，アンカーのリソースの指定方法を定める XPointer[2]では，そのリソースの指定を要素単位で行うことが可能である．また XML 文書とデータ交換を行うプログラムやスクリプトのための標準 API として公開されている DOM[3]では，要素単位の操作を行うための IDL や Java のインタフェースを定めている．さらに XML 文書に対する問合せ言語の一つとして提案されている XML-QL[4]においても，問合せ条件の指定や検索結果のビューの構成などを要素単位で指定することが可能な仕様となっている．

2. に関しては，システムが扱うべき XML 文書の DTD が複数ある場合，また現行使用している DTD がバージョンアップしたり，別の DTD を作成して使用する見込みがある場合などに必要とされる機能要件である．このような場合，DTD の特定の要素に注目したスキーマを個別に作成しり，追加・変更したりする方法では管理効率が悪い．この XML 文書管理の問題を解決する方法として，半構造化データのモデル[5],[6]をベースにする方法が注目されている[7]．

筆者らも XML 文書の管理には，構造情報をデータの一つとして取り扱う半構造化データモデルを利用し，各 XML 文書ごとに要素の構造情報を保持させる方法が適すると考える．本稿では，半構造化データモデルを基に，XML 文書の管理に特化・改良したデータモデルとその問合せモデルを提案する．本問合せモデルでは General Path Expression (GPE)[7]を採用する．DTD の違いにより要素の論理構造が異なる複数の XML 文書に対しても，GPE を用いて要素間の関係に関する検索条件を曖昧に指定することにより，各 XML 文書の構造の差異を吸収した検索を実現する．また，提案するデータモデルをオブジェクト指向データベース(OODB)を用いて実装した．この実装方法で要素の検索を高速に行うために，各 XML 文書ごとに要素のインデックスを作成してそれを検索時に利用する．また要素の探索経路の最適化も行う．これにより，半構造化データの問合せで指摘される検索性能面での不利[7]を改善する．

以下，第 2 章で XML 文書管理の方法を提案する．第 3 章で本提案に基づいた，OODB による実装方法を説明する．第 4 章では本方法に基づいた OODB 上のアプリケーションの試作例を示し，第 5 章で今後の課題を示す．

2 XML 文書管理方法の提案

半構造化データとして XML 文書を管理する方法を説明する．以下，XML 文書格納のためのデータモデルと，検索のための問合せモデルについて説明する．

2.1 データモデル

本稿では OEM[5]や P. Buneman[6]らの半構造化データモデルを参考に，XML や SGML などの構造化文書管理用のデータモデルとして設計した．本稿のデータモデルは汎用性のある OEM や P. Buneman らのモデルとは以下の点で異なる．

- ツリーモデルをベースとし，有向辺(エッジ)は双方向とする．
- 文書要素をツリーのノードと対応させ，ラベルはエッジではなくノードに付ける．
- 親ノードと子ノードの間の関係を順序付きリストとして管理する．
- 要素の属性は要素のノードの子ノードとし

て表す．複数の属性が存在する場合，その属性の順序(すなわち，子ノードの順序)は管理しない．

例えば，医療カルテをXML文書で表現する場合を考える．このとき，図1のDTDに従う図2のXML文書があるとする．この図2の文書を本稿のツリーモデルで表現すると，図3となる．ここで楕円のノードは要素を表し，要素名(タグ名)をラベルに持つ(要素ノード)．角丸長方形のノードは属性を表し，属性名をラベルに持つ(属性ノード)．#PCDATA型の文字列のデータは，該当する要素ノードの子ノードに記述する．また属性値は，該当する属性ノードの子ノードに記述する．各要素の中

```
<!ELEMENT カルテ     (患者,(疾患,臨床記録+)*)>
<!ELEMENT 患者       (氏名)>
<!ELEMENT 氏名       (#PCDATA)>
<!ELEMENT 疾患       (#PCDATA|疾患名|開始日|終了日)*>
<!ELEMENT 疾患名     (#PCDATA)>
<!ELEMENT 開始日     (#PCDATA)>
<!ELEMENT 終了日     (#PCDATA)>
<!ELEMENT 臨床記録 (#PCDATA)>
<!ATTLIST 患者       番号  NMTOKEN  #REQUIRED)>
<!ATTLIST 臨床記録 診断病名  CDATA  #IMPLIED)>
```

図1　医療カルテのDTDの例

```
<カルテ>
  <患者 番号="00038">
    <氏名>T.Kitano</氏名>
  </患者>
  <疾患>
    <疾患名>肺結核</疾患名>
    <開始日>1996/4/1</開始日>
    <終了日>1996/9/30</終了日>
  </疾患>
  <臨床記録 診断病名="肺結核">
  </臨床記録>
</カルテ>
```

図2　図1のDTDに従うXML文書の例

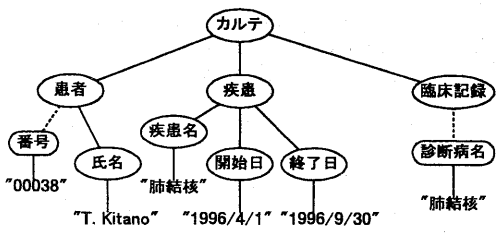

図3　図2のXML文書のツリーモデルによる表現

身，すなわち開始タグと終了タグで囲まれた部分は，対応する要素ノードの子孫ノードすべてとなる．図2の文書における要素の開始タグの出現順序に関しては，図3のツリーでは要素ノードの左深さ優先順序で表される．

2.2　問合せモデル

2.1節で説明したデータモデルで管理されるXML文書に対する検索式の記述方法を説明する．本稿で用いる検索式のシンタックスはOEMのLorel[8]と同じであり，GPEを加えて拡張したOQLの範疇に属する．ただしLorelとは前述の通り基礎となるデータモデルが異なるため，検索式のセマンティクスは以下の点で異なる(ただし，本稿に関する一部のみ)．

- メンバ選択演算子「.」の右側には子ノードのラベル，すなわち要素名か属性名が入る．または要素のメンバ関数(後述)が入る．
- GPEのワイルドカード「%」は，子ノードのラベルの中の0以上の文字列を表す．
- GPEのワイルドカード「#」は，左側のラベルのノードから，右側のラベルのノードにたどり着くまでの経路上の任意のノードを表す．右側のラベルが指定されていない場合は，左側のラベルのノード自身を含む子孫ノードすべてを表す．
- where句で用いる「=」は，要素の中身または属性値の文字列の完全一致を条件判定する等値演算子で，「like」は文字列の部分一致を条件判定する組み込み述語である．
- select句で指定されたノード以下のすべての子孫ノードが返却される．

複数の文書に対する以下の検索要求を満足させ本稿の検索式の一例を図4に示す．

「患者"00038"の疾患"肺結核"に関する"1996/4/1"から"1996/9/30"までの臨床記録を返せ」

```
select CR
from   カルテ.#.患者 P, カルテ.#.疾患 D,
       カルテ.#.臨床記録 CR
where  P.# = "00038"
and    D.%名 = "肺結核"
and    D.Overlap("1996/4/1", "1996/9/30")
and    CR.# like "肺結核"
```

図4　検索式の例

図 4 の from 句で，要素名「カルテ」から任意にたどることのできる要素名「患者」，「疾患」，「臨床記録」を持つ要素以下のサブツリー全体を，それぞれ変数 P，D，CR でバインドする．

where 句では from 句でバインドされた変数で指定されるサブツリーに対する条件判定を行う．where 句の最初の式では，要素名「患者」以下の任意のノードの中に，要素の中身または属性値として文字列"00038"と一致するノードが存在するかを判定する．二番目の式では，要素名「疾患」の子ノードに，後方一致で"名"を含むラベルを探し，そのノードで表される要素の中身または属性値が文字列"肺結核"と一致しているかを判定する．三番目の式では，要素名「疾患」のメンバ関数「Overlap()」を用いて条件判定する．メンバ関数の定義については 3.5 節で述べるが，ここでは「疾患」の「開始日」と「終了日」が 1996/4/1 から 1996/9/30 の期間と重なっているかどうかを判定するためのメンバ関数とする．where 句の最後の式では，要素名「臨床記録」以下の任意のノードの中に，要素の中身または属性値として文字列"肺結核"を含むノードが存在するかを判定する．

以上の処理の結果，select 句で指定されたサブツリーで，where 句の条件を満たした要素や属性が返却される．結果は複数の文書の中から図 2 で示すような文書が検索式の条件と一致し，その文書の中で条件を満たす要素，すなわち図 2 の 10 行目から 11 行目の要素「臨床記録」が返却される．

2.3 本問合せモデルの効果

本稿の問合せモデルでは GPE を採用している．GPE を用いることにより検索式における要素名や属性名の指定，要素間の関係などの条件を文字通り一般化することができる．すなわち，GPE を用いた検索式は，構造が異なる文書に対しても適用可能となる．特にインターネットなどの環境では，不特定多数のユーザが独自の DTD を使用して，各 DTD に従った構造の異なる複数の XML 文書を発信する可能性がある．本稿の検索式は，それら複数の文書における構造の差異を GPE の範囲で吸収して検索する方法を与えている．

図 1 から図 3 で示した XML 文書カルテと同じ医療カルテとしての意味を持つがその構造が異なる例として，DTD，XML 文書，XML 文書のツリー表現の例をそれぞれ図 5，図 6，図 7 に示す．

```
<!ELEMENT カルテ     (患者, (疾患, 臨床記録+)*)>
<!ELEMENT 患者       (氏名)>
<!ELEMENT 氏名       (#PCDATA)>
<!ELEMENT 疾患       (#PCDATA|病名|開始日|終了日)*>
<!ELEMENT 病名       (#PCDATA)>
<!ELEMENT 開始日     (#PCDATA)>
<!ELEMENT 終了日     (#PCDATA)>
<!ELEMENT 臨床記録   (主訴, 所見, 診断, 治療計画)>
<!ELEMENT 主訴       (#PCDATA)>
<!ELEMENT 所見       (#PCDATA)>
<!ELEMENT 診断       (#PCDATA)>
<!ELEMENT 治療計画   (#PCDATA)>
<!ATTLIST 患者       番号  NMTOKEN  #REQUIRED)>
```

図 5　図 1 と異なる医療カルテの DTD の例

```
<カルテ>
  <患者 番号="00038">
    <氏名>T. Kitano</氏名>
  </患者>
  <疾患>
    <病名>肺結核</病名>
    <開始日>1996/4/1</開始日>
    <終了日>1996/9/30</終了日>
  </疾患>
  <臨床記録>
    <主訴></主訴>
    <所見></所見>
    <診断>肺結核</診断>
    <治療計画></治療計画>
  </臨床記録>
</カルテ>
```

図 6　図 5 の DTD に従う XML 文書の例

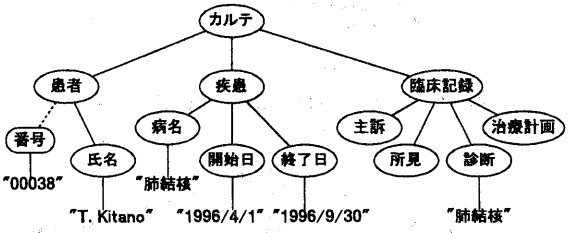

図 7　図 6 の XML 文書のツリーモデルによる表現

図 1 の DTD 対して図 5 の DTD の異なる点は以下である．

1. 要素「疾患名」という要素名が，要素「病

名」という別の要素名で定義されている．

2. 要素「臨床記録」に属性「診断病名」はなく，代わりに子要素「主訴」，「所見」，「診断」，「治療計画」を持つ．

これらの DTD の違いは，それぞれ以下に示す GPE による表現で吸収される(図 4 の検索式も参照)．

1. where 句の「D.%名」により，要素「疾患」の子ノードで，要素名の最後に"名"を持つ要素「病名」も評価の対象となる．
2. where 句の「CR.#」により，要素「臨床記録」の属性「診断病名」の代わりに，子孫要素「主訴」，「所見」，「診断」，「治療計画」が評価の対象となる．
3. select 句の「CR」により，要素「臨床記録」とそのすべての子孫要素，すなわち要素「主訴」，「所見」，「診断」，「治療計画」も返却の対象となる．

よって，図 4 に示した検索式を図 5 の DTD に従う図 6 の XML 文書に対して評価しても，結果として図 6 の XML 文書は検索式の条件と一致する．そして，図 6 の 10 行目から 15 行目の要素「臨床記録」とその中のすべての子孫要素「主訴」，「所見」，「診断」，「治療計画」が返却される．

上記の性質から，提案する検索式は，

- ✓ 基となる DTD が異なる複数の XML 文書に対する検索を行う場合
- ✓ 検索の際の要素名や属性名の指定，要素間の関係などの条件指定を簡略化したい場合
- ✓ そもそも DTD の詳細を知らなくて，厳密な名前の指定や要素間の関係指定ができない場合

などに有効に機能する．ただし GPE を多用すると，その曖昧性のために，ユーザが検索結果として意図しない文書や要素まで返却される可能性を広げることになる．よって GPE の利用は，文書の全文検索の場合と同様，大量の文書から目的の文書を絞り込むような目的で利用する方が良いと思われる．

3 実装方法

第 2 章で述べたデータモデルと問合せモデルに基づき，OODB のクラスライブラリとして実装する方法を説明する．OODB を選択する主な理由は，1)各インスタンス間を静的なオブジェクトリンクで関係づけて管理可能な点，2)継承によってクラスを追加可能な点，3)各クラス固有のメンバ関数(API)を定義可能な点，などである．その OODB のクラスライブラリとして実装し，XML 文書を管理するアプリケーション開発に利用する．

本クラスライブラリのインタフェースの一部を図 8 に示す．以下の節では，各クラスとそのメンバ関数について説明する．

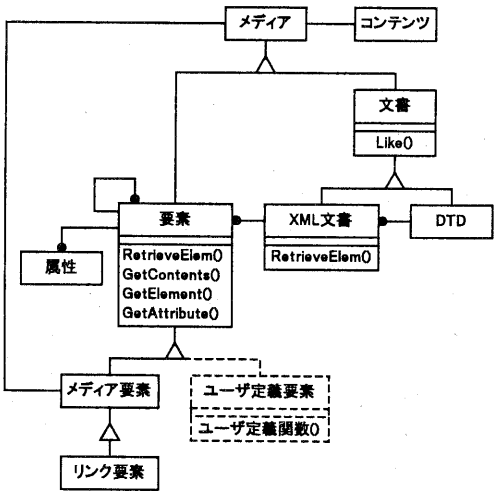

図 8 クラスライブラリのクラス図
(OMT 記法)

3.1 文書の格納

クラス「メディア」は本システムで格納対象となる全メディアの基本クラスである．メディアの実際のデータはクラス「コンテンツ」のインスタンスとして格納する．

クラス「文書」は全文検索機能を持ち，メンバ関数 Like()を呼び出して文書のコンテンツを全文検索することができる．

本稿で扱う XML 文書は，クラス「XML 文書」を通して OODB に格納する．クラス「XML 文書」は図 2，図 6 などで示す文書を管理する他に，図 3，図 7 などで示す各要素間の関係も管理する．XML 文書の文書構造を定義する DTD もクラス「DTD」を通して格納する．クラス「DTD」も図 1，図 4 などで示す DTD そのものを管理するだけでなく，DTD の要素型

宣言や属性リスト宣言などの内容，そしてそれら宣言の間の関係も管理する(図 9 が各宣言を管理するデータ構造の一例となっている).

3.2 要素インデックス

一般に要素の数は文書の数に比べて数十倍，数百倍も数が多い．すべての文書中のすべての要素をオブジェクトとして OODB に格納することは，管理すべきオブジェクトの数を莫大に増やすだけで現実的でない．よって本稿では，図 3 や図 7 で示すような要素同士のツリー構造のみを管理する「要素インデックス」を生成し，それをクラス「XML 文書」の各インスタンスに保持させて管理する方法を採用する．3.4 節で説明する要素単位での取出しや検索を行うメンバ関数は，すべてこの要素インデックスを用いたツリー上の計算に基づき，XML 文書中の該当部分の取出しを行う．

要素インデックスの生成は，1)XML 文書のインスタンス生成時，2)検索実行前，3)検索実行時，のどのタイミングでも可能である．要素単位での検索要求が予想される XML 文書に対しては，1)，2)のタイミングで生成しておいた方が性能の面では有利である．

3.3 要素の格納

要素インデックスを用いた計算によって XML 文書中の要素を取出す方法に加えて，クラス「要素」のインスタンスとしてあらかじめ OODB に格納する手段も用意する．この場合，格納する要素のインスタンスは，元文書となる XML 文書インスタンスと 1 対多のオブジェクトリンクの関係が保持される．このオブジェクトリンクをたどることにより，特定の XML 文書から特定の要素を取出すことができる．例えばマルチメディアデータとリンクする要素「メディア要素」や，他の文書とリンクする要素「リンク要素」などは，アプリケーションにとってはあらかじめ永続化して管理する方が効率的な場合もあるだろう．

3.4 要素の取出し・検索

XML 文書における要素の検索は，図 8 に示すクラス「XML 文書」のメンバ関数 RetrieveElem()を用いる．RetrieveElem()の引数に以下の 4 つの検索条件を指定し，これらの組み合わせによってさまざまな条件で要素を検索する．

- 要素名(タグ名)
- 要素の中身に含まれる文字列
- 属性名
- 属性値

RetrieveElem()の実行結果は，クラス「要素」のインスタンスの集合で返される．これらインスタンスに対してさらに以下のメンバ関数を呼び出すことも可能である．

- RetrieveElem()
  当該要素以下の要素(すなわち子孫要素)に対して RetrieveElem()を適用する．
- GetContents()
  当該要素の中身を取得する．
- GetElement()
  当該要素からのツリー上の相対位置(子/子孫/祖先/前/後/兄/弟など)[1]に対応する要素を取得する．
- GetAttribute()
  当該要素の属性を取得する．

図 4 の検索式と RetrieveElem()との関係を説明する．例えば図 4 で「from カルテ.#.患者 P」で記述される変数 P は，要素「カルテ」以下のすべての要素「患者」である．この要素「患者」のインスタンスを取得するには，要素「カルテ」のインスタンスを取得した後，

カルテ.RetrieveElem(“患者”); ・・・・・・・ (1)

として RetrieveElem を呼び出す．結果，要素「カルテ」以下で，要素名「患者」の要素インスタンスの集合が返却される．

さらに「where P.# = “00038”」で記述される条件を満たす要素「患者」を得るには，(1)の結果として得られた要素インスタンスすべてに対して，

患者.RetrieveElem(NULL, “00038”); ・・・・・ (2)
患者.RetrieveElem(NULL, NULL, “00038”);・ (3)

の 2 関数を実行する．関数(2)では，第一引数の NULL で要素名の条件指定を任意としてお

[1] XPointer[4]の相対ロケーション項の定義を参照

り，第二引数の文字列指定で要素「患者」以下の任意の要素の中身に対する文字列"00038"の完全一致を条件としている[2]．関数(3)では，第一引数の NULL で要素名の条件指定を任意とし，さらに第二引数の NULL で属性名の条件指定も任意としている．そして第三引数の文字列指定で，要素「患者」以下の任意の要素の中身に対してでなく，属性値に対しての完全一致を条件に指定している．結果はそれぞれの条件を満たす「患者」以下の要素インスタンスの集合が返却される．このとき，一つでも該当要素が返却されれば，where 句の要素「患者」の条件は満たされることとなる．

3.5 ユーザ定義要素

図 8 の「ユーザ定義要素」に示すように，ユーザはクラス「要素」の派生クラスを定義して，特定の要素に対する特別な属性をメンバ変数として与えたり，特別な処理をメンバ関数として実装しておくことが可能となる．本稿では，2.2 節で説明した要素「疾患」とそのメンバ関数「Overlap()」がその例に相当する．図 3 や図 7 の要素のツリー構造に基づき，要素「疾患」の子要素「開始日」と「終了日」で与えられる期間と，Overlap()の引数で与えられた期間と重なるかどうかを判定するメンバ関数をあらかじめ「ユーザ定義関数」として与えておけば，図 4 のような検索式からの呼び出しに応えることができるようになる．

3.6 要素探索経路の最適化

本クラスライブラリにおける要素を検索するメンバ関数 RetrieveElem()などの実装方法では，当該 XML 文書の文書構造を定義した DTD の各宣言の情報から，目的の要素を検索するための必要最小限の探索経路を計算する．そしてその求めた探索経路に従って，要素インデックスを用いた要素の検索を実行する．

図 6 の XML 文書に対して RetrieveElem("診断")を呼び出し，要素「診断」を検索する例で説明する．まず RetrieveElem()の呼び出しに応じて，当該文書の文書構造を定義した DTD オブジェクト(図 5)を取出す．クラス「DTD」では，各 DTD インスタンスごとに，概念的に図 9 で示すような要素や属性の宣言のツリー関係を構成したデータ構造を管理する．この DTD のツリー上の計算により，検索対象の要素にたどり着くための候補となる探索経路を計算する．計算の結果，検索対象である要素「診断」は要素「患者」，「疾患」，「主訴」，「所見」，「治療計画」(図 9 の×をつけた要素)以下には出現しないことが分かり，必要最小限の探索経路は「カルテ」→「臨床記録」→「診断」(図 9 の○をつけた要素)となる．

この探索経路の計算結果を用いて，図 7 の要素インデックスにおけるツリー上の計算では，最短の処理ステップで目的の要素「診断」にたどりつくことができるようになる．

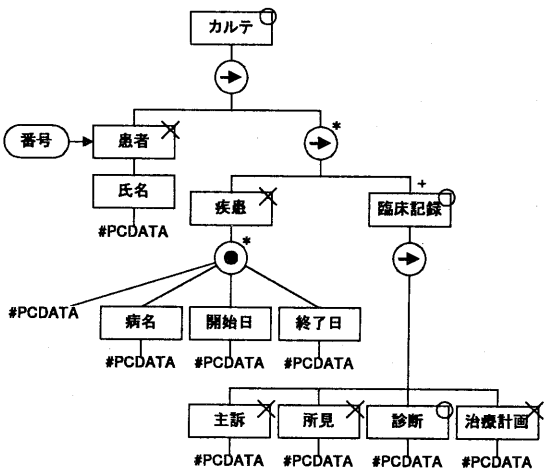

図 9 クラス「DTD」における各宣言の関係[3]
(図 5 の DTD を格納した例)

[2] 関数(2)の第二引数及び関数(3)の第三引数の文字列指定には，0 個以上の任意の文字列を表すワイルドカード「*」と，任意の 1 文字を表すワイルドカード「?」が利用可能である．各引数指定で"*00038*"とすれば，文字列"00038"の部分一致が条件となる．

[3] 図 9 の表記は，基本的に文献[9]の DTD の表記法に準ずる．長方形は要素型宣言を表し，角丸長方形は属性リスト宣言を表す．子要素の出現は，○に矢印で左から右に順番に出現，○に黒丸で選択的に出現することを表す．要素の出現回数は，*で 0 回以上，+で 1 回以上，その他は一回のみの出現を表す．

4 アプリケーションの試作

第3章で説明したXML文書管理のクラスライブラリは，OODB PERCIO[10]のマルチメディア文書クラスライブラリ(MMDCL)[11]の一部として開発した．そしてそのMMDCLを用いたPERCIOのアプリケーションとして，電子カルテ[12]をXML文書として管理する「XML電子カルテシステム」を試作した(図10)．本システムは，図4で示した検索式のようなカルテの構造に基づく複雑な検索が可能であること，また各病院間で用いるカルテのDTDが異なっていても，柔軟にその差異を吸収した検索が可能であることなどが特長である．

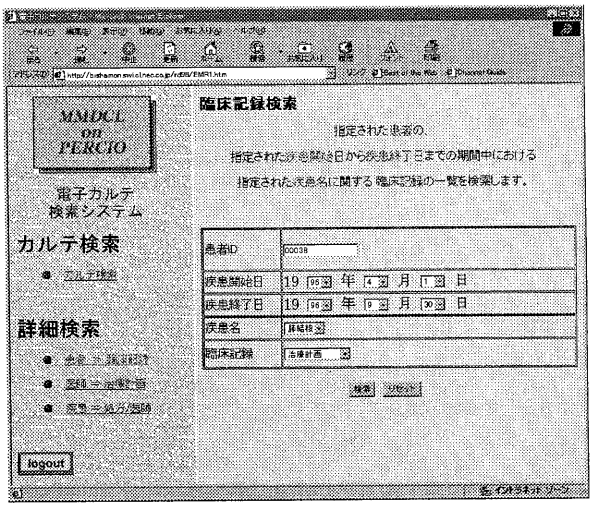

図 10 XML電子カルテシステム(検索画面)

5 今後の課題

一般に半構造化データに対する検索は，データ構造の条件指定に曖昧性を含むことにより，性能面での犠牲を払う．論文[13]発表の時点では，3.2節の要素インデックスを導入していなかったため，さらに性能面で問題があった．要素インデックスの実装による性能向上を評価し，性能面での改善を図る予定である．その上で，さらに検索性能の高速化を図ることが課題となる．

機能的な面では，XMLのリンクをたどる検索の実装などが今後の課題である．

参考文献

[1] Jon Bosak, "XML, Java, and the future of the Web", http://metalab.unc.edu/pub/sun-info/standards/xml/ why/xmlapps.htm, 1997.

[2] Eve Maler and Steve DeRose, editor, "XML Pointer Language (XPointer)", W3C Working Draft, WD-xptr-19980303, http://www.w3.org/TR/WD-xptr, 1998.

[3] Vidur Apparao, Steve Byrne et al., "Document Object Model (DOM) Level 1 Specification Version 1.0", W3C Recommendation, REC-DOM-Level-1-19981001, http://www.w3.org/TR/REC-DOM-Level-1, 1998.

[4] Alin Deutsch, Mary Fernandez et al., "XML-QL: A Query Language for XML", Submission to the W3C, NOTE-xml-ql-19980819, http://www.w3.org/ TR/NOTE-xml-ql, 1998.

[5] Serge Abiteboul, "Querying Semi-Structured Data", Proceedings of the Sixth International Conference on Database Theory, pp.1-18, Jan. 1997.

[6] Perter Buneman, "Semistructured Data", Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp.117-121, May 1997.

[7] Dan Suciu, "Semistructured Data and XML", Proceedings of the Fifth International Conference on Foundations of Data Organization, pp.1-12, Nov. 1998.

[8] Serge Abiteboul et al., "The Lorel Query Language for Semistructured Data", International Journal on Digital Libraries, Vol.1, No.1, pp.68-88, 1997.

[9] 日本ユニテックSGMLサロン 編著, "はじめてのSGML", 技術評論社, 1995.

[10] NEC Corporation, "PERCIO(ペルシオ)", http://www.ace.comp.nec.co.jp/product/db/percio/ p_main.htm, 1996-1998.

[11] Takayuki Saeki, Misa Namiuchi, "Extensible Multi-media Class Library for Object-Oriented Database Systems", Proceedings of the Ninth International Workshop on Database and Expert Systems Applications, pp.893-900, Aug. 1998.

[12] 羽澄 典宏 他, "電子カルテシステムの全体管理を行う"Medical Manager"の概念設計", 情報処理学会第57回全国大会講演論文集, Vol.4, pp.247-248, 1998.10.

[13] 北野 拓哉, 波内 みさ, "半構造化データモデルを利用したXML文書管理システムの試作", 情報処理学会第57回全国大会講演論文集, Vol.3, pp.283-284, 1998.10.