意味的に拡張したXMLと その応用のためのデータベースエンジンの実現

的野晃整† 野宮一生† 板谷昌洋† 国島丈生‡ 横田一正‡

† 岡山県立大学情報系工学研究科 ‡ 岡山県立大学情報工学部

{matono, issei, itadani}@c.oka-pu.ac.jp {kunishi, yokota}@c.oka-pu.ac.jp

〒719-1197 総社市窪木 111 〒719-1197 総社市窪木 111

XML は電子化文書の交換フォーマットとしてデファクトスタンダードとなっているが、基本的なデータ構造は木である。これは複合オブジェクトなどのデータモデルの表現、個別化情報の表現、マルチメディア情報の統合の実現などを考えると、それらの要件を反映したモデルを XML 上に実現する必要がある。そこで、我々はこれまで XML の意味的な拡張を行い [7]、その記述言語と問合せ表現についての議論 [10] を行ってきた。このとき、XML のモデルも従来の木ではなくエッジにラベルの付いた有向グラフとして表現した。本稿ではこれらを再整理し、応用のためのデータベースシステムとして設計し実現した。

本システムの特徴は、1) 拡張 XML の意味論を反映した設計、2) 1 つのグラフとなるデータベース、3) パスの制約表現の集合による記述、4) パスの制約表現による検索、5) ルールによる出力データの成形、などがある。

キーワード: XML, 情報の統合, グラフモデル, XML データベース, 拡張項

Implemantation of a Database Engine for Applications Based on Semantically Extended XML

Akiyoshi Matono† Issei Nomiya† Masahiro Itadani†

Takeo Kunishima‡ Kazumasa Yokota‡

† Okayama Prefectural University, Graduate School of Systems Engineering

{matono, issei, itadani}@c.oka-pu.ac.jp

111 Kuboki, Soja, Okayama 719-1197

‡ Okayama Prefectural University, Faculty of Infomation Science and System Engineering

{kunishi, yokota}@c.oka-pu.ac.jp

111 Kuboki, Soja, Okayama 719-1197

Although XML has become a *de facto* standard for information exchange, its structure is basically tree, that is, its expressive power is very restricted. To cope with various requirements of advanced applications, expressions such as complex objects with identities, personalization of shared information, and modeling multimedia contents, we have taken an approach for semantic extension of XML[7] and proposed its query language[10].

In this paper, we reconsider the above extensions and describe an overview of our system called QuaserII. The characteristics are as follows: 1) consistent design with our extensions, 2) a database as set of directed graphs, 3) data and a query consisting of path expressions, and 4) transformation of results in the form of rules.

Key words: XML, infomation integration, graph model, XML database, extended term

1 はじめに

インターネット上で急速に増加する情報の交換プロトコルとして XML(Extensible Markup Language) がデファクトスタンダードになっている。XML は要素を自由に定義できることから、さまざまな応用に対応することができると考えられているが、通常の意味論は与えられておらず、すべて応用に任されている。しかし、表現能力は高くないことからわれわれは XML の拡張を提案している。特に、XML の基本的な構造は木であるがこれは応用面から考えると強すぎる制限である。その理由は、全体の構造として有向グラフを表現したり、ノードに関しても複数の要素をひとつの要素として扱ったり、枝をまたがって同一の要素にするといった要件があるためである。

そこで、XML を踏み越える意味論を与える必要があると考え、XML の拡張を提案している [7]。それを再整理すると、有向グラフの導入のための**識別子**と**等値制約**の導入と、情報の共有化のための**部分文字列**と**意味的な親**の導入、要素間の構成で集合を表現するための**構成子**の導入、XML では非整形式になるため表現できない属性を表現するための**拡張属性**の導入である。

コンテンツの部分を要素を用いて指定することはできるが、コンテンツの要素間の重ね合わせの可能性を考慮すると XML の整礎性を壊す危険がある。またコンテンツを持つ複数の要素を 1 つの論理単位として扱いたい場合、従来はそれらの要素を要素で覆う形をとる。これはコンテンツが孫の位置になるため意味的な親子関係の指定は必要な拡張であると言える。XML は一つの要素で同じ属性名を複数記述することは許されていない、属性名と属性値をイコールで対として扱わなければならないなどの制限がある。この制限を超えるものが拡張属性である。

また、それらの意味論を反映した高次な記述言語と問合せ表現に関しての議論を行った [10]。本稿ではこれまで研究してきた拡張の意味論とそれを反映したモデリング、問合せ能力をもった言語に関して再整理し、それらのためのエンジンとしてデータベースを持つシステムの設計、開発をしたので報告する。

2 節ではこれまで提案してきた拡張 XML を新たな意味論も含めて再整理し、3 節ではそれらの意味論をふまえたパスに基づく記述言語 *Qpit* に関して述べ、4 節では *Qpit* 言語の問合わせ表現について考察する。5 節では拡張 XML の意味論を反映したデータベースの設計を行い、6 節ではそれまでの内容を他のアプリケーションで応用するためのシステムについて触れる。

2 XML の拡張

前節であげたような拡張を意味的に行う。まず例を挙げ、それに沿って説明していく。

例 1 識別子によるサザエさん夫婦

```
1  <夫婦 xmlns:q="Qpit">
2    <人 q:id="sazae">
3      <名前 q:id="name">フグ田サザエ</名前>
4      <夫 q:idref="masuo"/>
5    </人>
6    <人 q:id="masuo">
7      <名前>
8        <q:PCDATA substring="name(0,3)"/>
9        マスオ
10     </名前>
11     <妻 q:idref="sazae"/>
12   </人>
13   <趣味 q:constructor="set"
14         q:parents="masuo">
15     <q:PCDATA>ゴルフ</q:PCDATA>
16     <q:PCDATA>テニス</q:PCDATA>
17     <q:PCDATA>麻雀</q:PCDATA>
18   </趣味>
19 </夫婦>
```

この例はサザエさんの夫婦の関係を識別子によって表現した例である。1 行目は q に対して名前空間の定義を行っている。2 行目から 5 行目まではサザエさん、6 行目から 12 行目まではマスオさんの人物説明を行っている。それぞれ 2 行目で “sazae”、6 行目で “masuo” と識別子を定義し、それらを 11 行目で妻、4 行目で夫として参照している。識別子を互い違いに参照している点に注目すると、サザエさんとマスオさんは夫婦であることを表していることがわかる。識別子参照した要素と定義した要素は同じ実体であり、意味的な違いはない。

8 行目マスオさんの名字であるが、サザエさんの名前に識別子を付け、それの 0 文字目から 3 文字目までを参照している。これにより、“フグ田” を値に持つ要素を明示的に単位を生成しなくとも PCDATA の部分文字列を参照できる。さらに、文書系の情報では、1 つの文書に対して複数の人がアンダーラインを書いた場合、オーバーラップする可能性は十分考えられる。このような時、この属性の導入により XML を整形式に保つことができる。

13 行目からは趣味を集合で表現している。少し作為的ではあるが意味的な親を “masuo” としているため、これらの趣味はマスオさんの趣味であることを表現している。

□

例 2　等値制約による相思相愛

```
1    <person q:constaint="X">
2      <lover><q:Reference conref="Y"/></lover>
3    </person>
4    <person q:constaint="Y">
5      <lover><q:Reference conref="X"/></lover>
6    </person>
```

これは等値制約による相思相愛の例である。1 行目と 4 行目で、ある人物の等値制約をそれぞれ “X” と “Y” に定義して、それらを 5 行目と 2 行目で参照している。

要素を確定できていない場合は識別子を使用することはできない。したがって、この例のように誰かと誰かが愛し合っているというモデルを表現するためには等値制約を用いて表現する。等値制約は 2 つの意味があり、一方は上記の説明のような意味、他方は変数としての等値制約である。

□

以下のような要素や属性を導入した。

- 属性 $id =$ “$\langle id \rangle$”
  要素の識別子の指定
- 属性 $idref =$ “$\langle id \rangle$”
  指定した識別子をもつ要素を参照
- 属性 $constraint =$ “$\langle g\text{-}variable \rangle$”
  要素の等値制約の指定
- 属性 $conref =$ “$\langle g\text{-}variable \rangle$”
  要素の等値制約の参照
- 属性 $constructor =$ “$set \mid sequence$”
  要素の構成の指定
- 属性 $condition =$ “$\langle e\text{-}attribute \rangle, \cdots$”
  拡張属性の指定
- 属性 $parents =$ “$\langle id \rangle, \cdots$”
  親要素の指定
- 属性 $substring =$ “$\langle id(s\text{-}digit, e\text{-}degit)) \rangle$”
  指定した PCDATA の部分文字列の参照
- 要素 $Reference$
  任意場所での参照の指定
- 要素 $PCDATA$
  原子値であるコンテンツを持つ要素

上記の id 属性と $idref$ 属性による識別子の導入は有向グラフの導入に対応している。$constraint$ 属性と $conref$ 属性による等値制約の導入によって同一の関係を表したり、不確定な情報を扱ったりすることができる。また、要素の子要素間の構成を集合あるいは列として扱うために $constructor$ 属性を導入した。既定値は列である。$condition$ 属性では拡張属性を複数指定することができ、$parents$ 属性によって意味的な親子関係を表現する。$substring$ 属性は指定した識別子を持つ $PCDATA$ 要素の $s\text{-}digit$ から $e\text{-}digit$ までの部分文字列を要素間の重ね合わせによる危険性を意識することなく参照できる。そのため $PCDATA$ 要素は id 属性を持つことができる。また、$Reference$ 要素によって、任意の場所で等値制約参照、識別子参照や部分文字列参照ができる。そのため属性に $conref$, $idref$, $substring$ 属性のいづれかを保持し、空要素でなければならない。なお、上記の属性はすべてグローバル属性である。

前述したように有向グラフを導入するために、拡張 XML は正式には以下のようにグラフの方程式で記述する。例 1 の記述はこのシンタックスシュガーと位置付け、構文的には XML の整形式を維持させる。

$$
\begin{aligned}
\langle graph \rangle &::= \{\langle equation \rangle, \cdots\} \\
\langle equation \rangle &::= \\
&\quad \langle g\text{-}variable \rangle = (\ \langle id \rangle, \\
&\qquad\qquad \langle e\text{-}label \rangle, \\
&\qquad\qquad \{\langle attribute \rangle, \cdots\}, \\
&\qquad\qquad \{\langle e\text{-}value \rangle, \cdots\}) \\
\langle e\text{-}value \rangle &::= \text{“}\langle PCDATA \rangle\text{”} \mid \langle g\text{-}variable \rangle \\
\langle e\text{-}label \rangle &::= PCDATA \mid Refrence \mid \langle Element \rangle \\
\langle attribute \rangle &::= idref = \text{“}\langle id \rangle\text{”} \\
&\qquad \mid conref = \text{“}\langle g\text{-}variable \rangle\text{”} \\
&\qquad \mid constructor = \text{“}set \mid sequence\text{”} \\
&\qquad \mid condition = \text{“}\langle e\text{-}attribute \rangle\text{”} \\
&\qquad \mid parents = \text{“}\langle id, \cdots \rangle\text{”} \\
&\qquad \mid substring = \text{“}\langle id(s\text{-}digit, e\text{-}degit) \rangle\text{”} \\
&\qquad \mid \langle a\text{-}label \rangle = \text{“}\langle a\text{-}value \rangle\text{”}
\end{aligned}
$$

$\langle g\text{-}variable \rangle$ は変数で $\langle equation \rangle$ 間の等値制約の役割を果たしている。$\langle id \rangle$ は $\langle equation \rangle$ の識別子であり、$\langle e\text{-}label \rangle$ は要素名、$\langle e\text{-}value \rangle$ は部分要素で、$\{\langle e\text{-}value \rangle, \cdots\}$ はその集合で要素値となる。$\langle a\text{-}label \rangle = \langle a\text{-}value \rangle$ は通常の属性で、$\langle e\text{-}attribute \rangle$ は拡張属性である。$\langle id \rangle, \langle g\text{-}variable \rangle$ 以外の特殊な属性は普通の属性と同様に記述する。$\langle PCDATA \rangle$ は原子値である。要素値が原子値である場合は要素名は必ず、$PCDATA$ とする。

このグラフの意味論は $\mathcal{QUIXOTE}$[6] 同様に超集合論上で定義する。つまりグラフの式で表現される拡張 XML

の意味はその式の制約解消した結果の式であり、解を求める手続きが超集合論上の制約解消系である。その可解性と合流性は保障されている。$g_1 = (id_1, l_1, \{A_1\}, \{E_1\})$ と $g_2 = (id_2, l_2, \{A_2\}, \{E_2\})$ が同一であるかは、g_1 と g_2 の bisimulation による。

また、上記のグラフで扱う拡張 XML のモデルはエッジに属性付きのラベルをつけた有向グラフである。

次に、先ほどの例 1 と例 2 をグラフで表現した。

例 3　グラフ表現

1. サザエ夫婦

$$
\begin{aligned}
g_0 &= (\_, 夫婦, \{xmlns{:}q = \text{“}\mathit{Qpit}\text{”}\}, \{g_1, g_5, g_{10}\})\\
g_1 &= (sazae, 人, \{\}, \{g_2, g_4\})\\
g_2 &= (name, 名前, \{\}, \{g_3\})\\
g_3 &= (\_, q{:}PCDATA, \{\}, \{\text{“フグ田サザエ”}\})\\
g_4 &= (\_, 夫, \{q{:}idref = \text{“}masuo\text{”}\}, \{\})\\
g_5 &= (masuo, 人, \{\}, \{g_6, g_9\})\\
g_6 &= (\_, 名前, \{\}, \{g_7, g_8\})\\
g_7 &= (\_, q{:}PCDATA,\\
&\qquad \{q{:}substring = \text{“}name(0,3)\text{”}\}, \{\})\\
g_8 &= (\_, q{:}PCDATA, \{\}, \{\text{“マスオ”}\})\\
g_9 &= (\_, 妻, \{q{:}idref = \text{“}sazae\text{”}\}, \{\})\\
g_{10} &= (\_, 趣味, \{q{:}constructor = \text{“}set\text{”},\\
&\qquad q{:}parents = \text{“}masuo\text{”}\}, \{g_{11}, g_{12}, g_{13}\})\\
g_{11} &= (\_, q{:}PCDATA, \{\}, \{\text{“ゴルフ”}\})\\
g_{12} &= (\_, q{:}PCDATA, \{\}, \{\text{“テニス”}\})\\
g_{13} &= (\_, q{:}PCDATA, \{\}, \{\text{“麻雀”}\})
\end{aligned}
$$

2. 相思相愛

$$
\begin{aligned}
X &= (, person, \{\}, \{g_0\})\\
g_0 &= (, lover, \{\}, \{Y\})\\
Y &= (, person, \{\}, \{g_1\})\\
g_1 &= (, lover, \{\}, \{X\})
\end{aligned}
$$

□

例 2 で述べたように、等値制約はデータベースの空値、ψ-項での「タグ」、部分情報の制約変数などの役割をもっている。しかし XML の表現単位は任意性があるために、タグ (グラフ変数) のスコープには注意が必要である。つまり $X = masuo, Y = sazae$ と $X =$ ミッキーマウス $, Y =$ ミニーマウス という 2 つの相思相愛関係を表現するためには同一の X と Y は使用できないために、グラフ変数は必要に応じて改名する必要がある。

3　拡張 XML のパス表現

2 節では XML の拡張を行って、その構文と意味論としてのグラフに関して述べた。しかし、これらの表現は冗長な部分を残していてあまり使いやすいとは言えない。そこで、我々は拡張項に基づく表現言語 $\mathcal{Q}pit$(Query by Path and Intelligent Term) を提案している。それは、XML の要素を項と対応させ、項の連結によるパスに基づいた制約による表現である。この表現はコンテンツの記述や操作を直接行うことを考慮している。

$\mathcal{Q}pit$ は情報を表現する記述と情報に対して操作をする記述からなっており、その特徴は項を連結させたパスの制約集合に基づいた高次な記述、拡張 XML の意味論を反映したモデル、アクセスパスによる情報のフィルタリング、正規表現や変数の導入、ルールによる出力構造の指定などがある。

ここでは情報を表現する記述の例を挙げる。例 1 と例 2 を $\mathcal{Q}pit$ で表現すると以下のようになる。

例 4　$\mathcal{Q}pit$ 表現

1. サザエ夫婦

$$
\begin{aligned}
&夫婦/[\ 人\#sazae/[名前@g_0/\text{“フグ田サザエ”}\#name,\\
&\qquad\qquad 夫\ (\#masuo)],\\
&\qquad 人\#masuo/[名前@g_1/[(\#name(0,3)),\\
&\qquad\qquad\qquad\qquad \text{“マスオ”}],\\
&\qquad\qquad\qquad 妻\ (\#sazae)],\\
&\qquad 趣味\ (+set\#\#masuo)/[\text{“ゴルフ”},\\
&\qquad\qquad\qquad\qquad \text{“テニス”},\\
&\qquad\qquad\qquad\qquad \text{“麻雀”}]]
\end{aligned}
$$

2. 相思相愛

$$
\begin{aligned}
&person@X/lover@g_2/(@Y)\\
&person@Y/lover@g_3/(@X)
\end{aligned}
$$

□

ここで、例 4.1 の $@g_0$ や $@g_1$、例 4.2 の $@g_2$ や $@g_3$ はそれぞれの項が異なった存在であることを表現するために用いている。

$\mathcal{Q}pit$ 表現を以下のように定義する。

$$
\begin{aligned}
\langle path\rangle ::= &\ \langle term\rangle\\
| &\ \langle path\rangle/\langle term\rangle
\end{aligned}
$$

$$
\begin{aligned}
&| \quad \langle path\rangle/[\langle path\rangle,\cdots]\\
\langle term\rangle ::= &\ \langle e\text{-}label\rangle\\
&| \quad \text{“}\langle PCDATA\rangle\text{”}\\
&\qquad @\langle variable\rangle?\\
&\qquad \#\langle id\rangle?\\
&\qquad\quad (\ +set \mid sequence\ ?\\
&\qquad\quad \#\langle variable\rangle?\\
&\qquad\quad \#\langle id\rangle?\\
&\qquad\quad \#\langle id\rangle(\langle start\rangle,\langle end\rangle)?\\
&\qquad\quad \#\#\langle id\rangle\cdots\\
&\qquad\quad \langle e\text{-}attribute\rangle,\cdots\\
&\qquad\quad \langle attribute\rangle,\cdots)?
\end{aligned}
$$

要素名 $\langle e\text{-}label\rangle$ は項のラベルである。等値制約 $\langle variable\rangle$ は @ に続けて記述し、() 外の等値制約はその定義で、内の等値制約はその参照を意味している。識別子 $\langle id\rangle$ は # に続けて記述し、等値制約と同様に () 外の識別子はその定義で、内の識別子は参照である。また () 内の識別子参照の直後に (⟨start⟩, ⟨end⟩) と記述された時は $\langle start\rangle$ から $\langle end\rangle$ までの部分文字列の参照となる。つまり、拡張 XML の *substring* 属性である。構成子は () 内の先頭に記述し、省略時は規定値の列となる。意味的な親の指定は () 内で ## に続けて記述する。これは拡張 XML の *parents* 属性に対応している。拡張属性や通常の属性は “,” で区切って複数記述できる。これらの記述は匿名であった場合はすべて省略できる。

ここでパス制約の連結を説明する。

```
<l1>
   <l2/>
</l1>
```

のように入れ子関係になっているとき

$$
\begin{aligned}
v_1 &= (\_, l_1, \{\}, \{v_2\})\\
v_2 &= (\_, l_2, \{\}, \{\})
\end{aligned}
$$

のようなグラフとなる。ここで、注意すべきは 1 行目の最後に 2 行目のグラフ変数が記述されている点である。これにより、前述した入れ子関係を表現する。さらにパスの制約表現では

$$l_1/l_2$$

と表現できる。このように項を連結することができる。これをパスと言い、l_1 の部分要素は l_2 であるという制約を持つ。

また、識別子によって同一な存在であることを表している場合、

```
1    <l0>
2       <l1 q:idref="id"/>
3    </l0>
4    <l2 q:id="id">
5       <l3/>
6    </l2>
```

のように 4 行目の識別子を 2 行目で参照している場合、グラフは以下のようになる。

$$
\begin{aligned}
v_0 &= (\_, l_0, \{\}, \{v_1\})\\
v_1 &= (\_, l_1, \{idref{=}\text{“}id\text{”}\}, \{\})\\
v_2 &= (id, l_2, \{\}, \{v_3\})\\
v_3 &= (\_, l_3, \{\}, \{\})
\end{aligned}
$$

このとき、l_1 と l_3 は存在が同じとみなし、以下のようなパスで記述できる。

$$
\begin{aligned}
&l_0/l_1(\#id)\\
&l_2\#id/l_3
\end{aligned}
$$

あるいは

$$l_0/l_2\#{=}l_1/l_3$$

といった表現をすることができる。後者の記述では = によってつながれた項は同一の存在であることを表している。前者の記述は構造情報をもっており、後者は構造情報ももっていないが、ここで構造情報は大きな意味をもたない。むしろ意味情報を一目で把握できる後者の記述の方が高レベルな表現である。

例 4 でも例 1 や例 2 で記述されている構造情報を残した記述になっているが、より意味的な関係を一目で把握できる記述の方が好ましいのであれば、例 4.2 などは次のようにも記述できる。

$$person@X/lover@g_1/person@Y/lover@g_3/(@X)$$

また、以下のように要素値が原子値である場合

$$
\begin{aligned}
v_1 &= (\_, l_1, \{\}, \{v_2\})\\
v_2 &= (id, PCDATA, \{\}, \{\text{“}pcdata\text{”}\})
\end{aligned}
$$

要素名は必ず $PCDATA$ であるが、$\mathcal{Q}pit$ 表現では

$$l_1/\text{“}pcdata\text{”}\#id$$

と記述する。このように $PCDATA$ の場合も識別子や等値制約、属性など記述できる。

複数のパス制約は以下のように省略して記述することができる。

$$\left.\begin{array}{l} label_0/label_1/\text{“}e_1\text{”} \\ label_0/label_2/\text{“}e_2\text{”} \\ \quad\vdots \\ label_0/label_n/\text{“}e_n\text{”} \end{array}\right\} label_0/[\, label_1/\text{“}e_1\text{”}, \; label_2/\text{“}e_2\text{”}, \; \vdots \; , \; label_n/\text{“}e_n\text{”}]$$

4 グラフ操作

Qpit は情報を表現する記述と問合せによって情報を操作する記述に大きく分けることができる。前節では前者の *Qpit* で情報を表現する記述についての説明をした。この後者は前者の記述を基本的に包含しており、その記述を拡張した記述である。本節では、後者の情報を操作を記述に関して説明をする。

グラフ構造をもつ情報をパスで表現することは難しいが、グラフの特定のノードに対しての経路を表現するにはパスは非常に便利な表現方法である。本研究でパスを導入した最も大きな理由はここにある。

尚、特定の部分要素へのアクセスを考えるとき、ルートからパスを記述する必要があるが対象の情報は木ではないため意味的にルートは存在しない。そのため、識別子や等値制約、要素名、属性によってすべての項を一意にする必要がある。

ここで例 1 の情報に対して原子値 “フグ田サザエ” を示す記述の 1 つで

$$人\#masuo/妻/名前/\text{“フグ田サザエ”}$$

のように表現することもできる。

このようにして情報に対して特定の部分情報を指し示したパスをアクセスパスあるいはフィルタリングパスと呼ぶ。アクセスパスによって抽出された情報に対して操作を行うことを可能にすることができる。この概念は *Qpit* の特徴の一つである。

ここで、正規表現を導入することでアクセスパスに幅もたせることができ、データの構造を完全に把握していなくとも、ある程度コンテンツの特定が可能となる。また、パス中で操作の対象となる項を特定するために変数を導入し、項の束縛も可能にする。変数にはアクセスパスのパタンとマッチしたパスの制約集合中の項群が束縛される。

まず、パス間の順序は $m \leq n$, $1 \leq \forall i \leq m$, $l_i \sqsubseteq l'_i$ のとき

$$l_1/l_2/\cdots/l_n \quad \sqsubseteq \quad l'_1/l'_2/\cdots/l'_m$$

が成立する。パス間に $p_1 \sqsubseteq p_2$ の関係があったとき、p_2 は p_1 のパタンである。$e_1/\ldots/e_l\$X/\ldots/e_m$ が $e'_1/\ldots/e'_l/\ldots/e'_m/e'_{m+1}/\ldots/e'_n$ のパタンであったとき、変数 X に束縛される項群は e'_l である。パタン中に正規表現がある場合、可能性のあるすべてのパス制約が対象となる。

このようなパスによる情報のフィルタリングは XPath[1] と類似しているが、最も大きな違いは XPath が木をデータモデルにしており、*Qpit* のアプローチはそれを包含することを意図している。つまり、XPath は IDREF 型の属性を持つ要素からの参照は、XPath の id 関数や XPath2.0[2] の dereference 演算子によって行うことはできるが、*Qpit* では識別子参照のアクセスと子要素のアクセスとに意味的な違いはほとんどなく、まったく同じ表現で記述できるという点が異なる。しかし、XPath には関数があり *Qpit* にはないため、特殊な操作をすることは難しい。しかし、特殊な操作に関して以下に示す問合せによって行うことを前提に考えている。

ここで操作について説明するが、現在操作は構造の変更しか検討しきれていない。今後、情報に対しての更新や追加など特殊な操作を含めて検討していきたい。フィルタリングによって得られた項は、検索の結果として出力する際に目的の構造であるとは限らない。そのため、出力結果の構造を適切な構造に変換するためにルールを使用して、そのヘッドに変換のための情報を記述する。つまり

出力結果の構造情報

<= 検索に使用するアクセスパスの制約集合

とする。出力結果の構造情報も *Qpit* のパスで表現する。アクセスパス中に記述された変数と同じ変数をヘッドにも記述することで束縛されている項群を出力結果のどの部分とするかを指定することができる。

ここで、例 1 に対して問合せを行う例を幾つか挙げる。

例 5 問合せの例

1. 変数束縛と構造変換

$$?-address/name/\$A$$
$$address/name/\$A <= 人/名前/\$A$$

この問合せは人の名前の一覧を生成している。右辺で変数 A に “フグ田サザエ” と “フグ田マスオ” という値を束縛している。右辺の “人” や “名前” の項には本来識別子や等値制約などの特殊な属性があるがこのように省略して記述することもできる。この問合せの結果を以下に示す。

```
<address>
  <name>フグ田サザエ</name>
  <name>フグ田マスオ</name>
</address>
```

このように左辺で記述された構造にしたがって解を出力している。ここで表記上 XML を用いたが、解はあくまで閉包性のある構造のみでありレイアウトは含んでいない。このように構造とレイアウトを分離することで出力レイアウトを状況に応じて変更することが可能である。

2. 少し複雑な例

$$
\begin{aligned}
&?- \$B/\text{“}\$D \text{ を好き”}\\
&\$B/\text{“}\$D \text{ を好き”}\\
&\qquad <= \$A/[\text{趣味/”ゴルフ”},\\
&\qquad\qquad !*\text{名}!*/\$B,\\
&\qquad\qquad \text{妻}/!*\text{名}!*/\$D]
\end{aligned}
$$

この問合せの大まかな意味は趣味がゴルフである人の妻の名前を求めている。正規表現によって “名前” でも “氏名” でもパタンにマッチするような記述になっている。!は任意の一文字で、*は直前の文字 (列) を 0 から無限回繰り返す。妻の名前を変数 $\$D$ に束縛している。出力結果は

```
<フグ田マスオ>
    フグ田サザエを好き
</フグ田マスオ>
```

となる。

□

現在、問合せには 1 つのルールしか記述できないが、今後より複雑な構造の変換を行うために複数のルールに対応することをふまえて考察したい。

また、この問合せによる構造の変換は W3C によって開発が進められている XQuery[4] や XML-QL[3] に比べ演算子や関数などまだ不十分な点を残しており、あまり強力な問合せ言語ではない。しかし基本的なデータモデルが異なるため、それらの言語にはない部分もある。構造の変換に関しても XQuery の construct に比べて出力結果が単純すぎるので、出力の変換は別の出力編集機能でやることを前提に今後検討したい。

5 データ管理

コンテンツのモデリングは応用に依存しており、メディア間の関連付けがシステム構築者のみの意図を反映した静的なものとなっている。そこで、ユーザの意図にしたがったモデリングを動的に行える必要があると考えられる。我々はこのような観点から、各観光地情報を様々なメディアでデジタル化したものを観光ポイントと呼び、そのモデリングを出発点とし、その一般化を考えた研究 [8] を行っている。そこで提案しているコンテンツベースを実現するには、少なくともこれまで拡張してきた XML の意味論を反映したデータベースである必要がある。しかし、ここで議論するデータベースは、以前から提案しているコンテンツベースの機能を完全に実現するものではない。それは、その基礎概念だけを取り入れているためで、観光ポイントのコンテンツデータベースとして扱うなら、観光ポイント用の API を作成し、それによってデータを扱う必要がある。

これまでの拡張は XML 自体、またはその要素を対象にしていたが、ファイルやディレクトリも基本的な構造は木であることに着目し、それらも要素として扱うことのできるデータベースを考えた。つまり、それ自体が大きな一つのグラフとなるのである。

OS のファイルシステム上で XML を扱う場合、一般的にファイル単位で扱うが我々の提案している XML データベースでは、それ自体が 1 つの拡張 XML となるものでファイルやディレクトリの概念が無く、すべて部分木の要素という単位で扱う。このためデータの処理には時間がかかるが、前節で説明した問合せによって元々はファイルやディレクトリであったものもすべて XML の要素と同様に扱うことができる。

しかし、すべての XML がこのデータベースに保存してあるとは限らない。従来 XML はテキストで記述されファイル単位で保存されている。そのため、データベースに保存されたデータとファイルとして保存してあるデータとでは、問合せの対象範囲が異なる。つまり、一方では検索対象を特に指定する必要がなく、他方では検索対象の XML ファイルを指定する必要があると言うことで

ある。

そこで、これまで議論してきたパスの有効範囲を広げファイルやディレクトリも問合せの対象として扱うように拡張する。つまり、パス表現で URI を扱うことができるようにすることである。しかし、将来的に問合せによるデータの更新を考えているため、ファイルとして保存されているデータに対して問合せを行うことは、ファイルシステムの構造を変えてしまうことにつながる。また、一般的なファイルシステムは木構造であるため *Qpit* の有向グラフを表現できない。そのためデータベースに保存されるデータとファイルシステムのデータとは多少異なった問合せを行う必要がある。

例 6　ファイルとデータベースの違い

- ファイルに対するルール

$$loves/[\$A, \$B] \\ <= file:///\backslash(!*/\backslash)*/DEWS\backslash[0\text{-}9\backslash]*/ \\ sample.xml/[\ \$A/妻\$B, \\ \$B/夫\$A]$$

ファイルへの問合せの場合、変数への束縛の対象は XML ファイルより下の要素になる。この例は *sample .xml* というファイル中の相違相愛の $\$A$ と $\$B$ を求めるアクセスパスである。\(\) 内の文字列を繰り返すことを指しているため、この記述は任意要素名で、任意階層を意味している。

- データベースに対するルール

$$\$C/loves/[\$A, \$B] \\ <= db:///Qpit/DEWS\backslash[0\text{-}9\backslash]*\$C/ \\ sample.xml/[\ \$A/妻\$B, \\ \$B/夫\$A]$$

データベースのの場合、データベースより以下であれば元々XML ファイルであった要素でも変数に束縛できる。この例ではディレクトリであったものに対して束縛している。

□

前述したが XML データの更新なども *Qpit* の問合せで行うことを考えている。これには更新対象の特定や更新情報などをどのように表現するか検討する必要があるが、現時点では考えきれていないため、今後の課題にしたい。

6　実装と評価

これまで拡張 XML とそれを表現、操作するための言語 *Qpit* に関して議論した。本節では、それらを他のアプリケーションで応用するために開発したプロトタイプシステムについての議論、評価を行う。

実装は Java2(jdk1.3.1) を用いて行った。本システムの特徴は、拡張 XML の意味論を反映した設計、1 つのグラフとなるデータベース、パスの制約表現の集合による情報記述、問合わせ機構、ルールによる出力データの成形などが挙げられる。

本システムは図 1 に示すように四つの処理系によって構成される。各処理系の働きについて説明する。パーサ

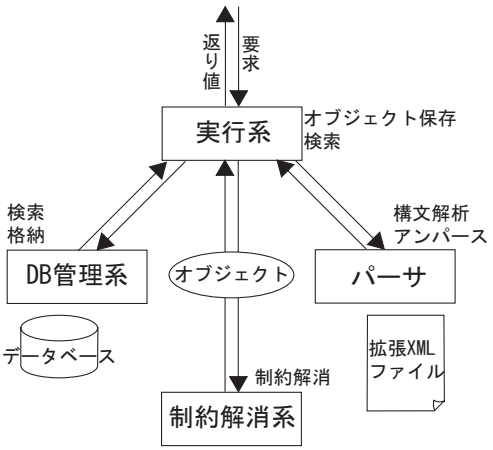

図 1: *Qpit* システムの概要図

は拡張 XML と *Qpit* の構文解析を行ってグラフをモデルとしたオブジェクト化を行う。制約解消系はグラフ間の同一性判定の規則にしたがって制約の矛盾の発見、解消を行う。データベース管理系はデータベースに保存されているデータの操作を行う。実行系は問合せ要求にしたがって各処理系に処理を分配する。また、データの検索、操作を行う。

本システムは応用アプリケーションからの要求にしたがって実行系が各処理系に命令を出して結果を返す機能とデータの API を提供している。要求は大きく分けて検索と保存の 2 つあると考えられる。

検索の場合、対象がデータベースと通常のファイルの 2 種類あり、前者の場合は保存する際にすでに XML をオブジェクト化してあるため再び構文解析する必要はない。後者の場合はテキストファイルであるため構文解析後、データのオブジェクト化を行って、制約の解消をする必要がある。整形化されたオブジェクトに対して検索を行う。

保存の場合も検索と同様に対象が 2 種類あり、データベースの場合はすでに保存してある情報との間で制約の解消を行う必要がある。また、ファイルの場合はオブジェクトをアンパースして保存する。また直列化してオブジェクトのまま保存することもできる。

本システムのデータベースは Oracle8i を用いて開発した。Oracle8i は RDB であるため、XML データの格納に適しているとは言えない。そのため、データモデルをグラフにして要素毎に格納している。5 節で説明したようにデータベース自体が一つの大きなグラフとなるため、一つのテーブルに次々と格納していく。これはパフォーマンスを下げる原因になるが、ファイル単位で格納した場合ファイルを含めた要素を部分木として扱うことから、情報のほとんどないテーブルが増加していく危険がある。

7 終りに

本稿では、現在検討している XML の意味的拡張とその問合せ言語 *Qpit* の要求仕様、またそれを他のアプリケーションで応用するための XML データベースエンジンについて述べた。

有向グラフや情報の共有化、集合などを表現するために XML の拡張を行い。拡張 XML の意味論としてグラフ表現導入し、その制約解消によって意味を導出できる。さらに、拡張 XML の記述が面倒である、問合せ表現がないといった問題点を解消するためにパスの制約に基づいた *Qpit* を導入した。*Qpit* は情報を表現するための記述と情報を操作するための記述に大きく分類できる。

今後の課題として変更以外の更新や削除などの機能の追加、複数のルールに対応して複雑な構造の変更、レイアウトに関して問合せ表現の強化をする必要がある。また、現時点で実装できている部分は内部データやパーサ、データベース管理の一部である。今後、早急に残りの部分に関して実装したい。それに沿って、正規表現と変数の導入に伴なうパタンの順序、項やパスの評価結果などに関してアルゴリズムを再検討する必要があるかも知れない。今回試作した *Qpit* システムのデータベースの API は観光ポイントのそれとは異なっているため、観光ポイントのコンテンツを扱うために、その概念を反映した API を作成する必要がある。

本研究の背景は演繹オブジェクト指向 $\mathcal{QUIXOTE}$[6] を分散環境へ拡張した QUIK[5]、マルチメディア情報を同期的対話的に制御した QUIK-II[9] の流れに沿っている。これらは、情報の統合を目的としており、その実現のためのさまざまなアプローチの一つである。本システムの応用として情報の統合という観点からコンテンツの再利用性、情報の個人化などを特徴とするマルチメディア情報提示システム GraphiX(GRAPHical presentation system by Individual data and extended Xml) の実装、情報の個人化のために XML 文書に対する情報の差分を記述することを目的とした、XML 操作スクリプト言語 xTrics(XML TRee Infomation Control Script) の研究などを行っている。

謝辞

最後に、さまざまな議論を頂く岡山県立大学横田研究室に携わる皆様に深謝します。

参考文献

[1] XML Path Language (XPath) Version 1.0. http://www.w3.org/TR/xpath, 16 Nov 1999. W3C Recommendation.

[2] XML Path Language (XPath) Version 2.0. http://www.w3.org/TR/xpath20, 20 Dec 2001. W3C Working Draft.

[3] XML-QL: A Query Language for XML. http://www.w3.org/TR/NOTE-xml-ql/, 19 August 1998. Submission to the World Wide Web Consortium.

[4] Query 1.0: An XML Query Language. http://www.w3.org/TR/xquery, 20 Dec 2001. W3C Working Draft.

[5] Bojiang Liu Kazumasa Yokota and Nobutaka Ogata. “Specific Features of the QUIK Mediator System,” IEICE Transactions on Information and Systems, vol.82, no.1 pp.180-188, Jan 1999.

[6] Kazumasa Yokota and Hideki Yasukawa. “Towards an Integrated Knowledge-Base Management System — Overview of R&D on Databases and Knowledge-Bases in the FGCS Project” Proc. Int. Conf. on Fifth Generation Computer Systems (FGCS’92) pp.89-112, June 1992. Tokyo.

[7] 横田一正, 國島丈生, and 劉渤江. “異種文書管理のための問合せ機構の考察”. 情報処理学会データベースシステム研究会＆電子情報通信学会データ工学研究会合同ワークショップ, Jul 2000. 花巻.

[8] 岡本愛子, 赤木絵里, 国島丈生, and 横田一正. “マルチメディアコンテンツの柔軟なモデリング方式の考察”. IEEE 広島支部学生シンポジウム, Jan 2001. 広島.

[9] 的野晃整, 藤野猛士, 平野尚孝, 板谷昌洋, and 横田一正. “対話的演出機能をもつマルチメディア提示システムの実現”. 夏のデータベースワークショップ 2001 (DBWS2001), July 2001.

[10] 的野晃整, 板谷昌洋, 横田一正, 國島丈生, and 劉渤江. “データベースの概念を組み込んだ XML のための問合せ言語” 電子情報通信学会技術研究報告, DE2001-13, June 2001. 奈良.