緩い構造を持つデータに対する問合せ語間の関連を反映する検索方式の提案

坂口 良† 清光 英成‡ 大月 一弘‡ 森下 淳也‡ 依田 平‡

†神戸大学大学院総合人間科学研究科 〒657-8501 兵庫県神戸市灘区鶴甲 1-2-1

E-mail: †sakaguchi@mediafusion.co.jp

あらまし

Web 検索や XML 検索は一般的に全文検索技術などを用いて実現されている。これは、テキスト中に問合せ語が存在するかどうかを判別するだけであり、ユーザの検索意図を十分に反映した結果を取得できるとはいえない。問合せ語間の係り受け関係を容易にユーザが指定できる検索式記述の考案が必要であると共にその解釈を寛容にしておく必要もある。本論は上記の問題を解決する為の問合せモデルを明らかにする。また、XML 検索を例として試作したプロトタイプシステムの評価について議論する。

キーワード XML 情報検索、Web とインターネット、問合せ処理、XML データ管理

1. はじめに

Web 検索は、ディレクトリ型、ロボット型の二つに大別出来る。ディレクトリ型は、ディレクトリ分類された項目を辿って目的のデータを探す方式であり、ロボット型は入力されたキーワードを全て含む Web ページの一覧を表示する。ディレクトリを辿るだけ、キーワードを並べるだけの簡単さと平易さを持ち有用で利便性が高い。

Web と同様に緩い構造を持つデータとして XML（eXtensible Markup Language）がある。XML はスキーマレスで構造の変更に強いという特徴を持ち、データを表現する方法として有用である。元々HTML と XML は SGML から派生した言語である。HTML は表示をするためのマークアップ言語であり、XML はデータ交換をするためのマークアップ言語である。HTML と XML は表記方法が似ているが、検索方法は全く異なる。XML はデータ構造を把握していないと検索することが出来ない。

XML文書から必要な情報を取得する場合、XQuery（XML Query）、XPath（Xml Path language）などの検索言語を用いるのが一般的である。XQueryは、XMLデータ問合せのための関数型言語であり、SQLのSELECT、FROM、WHEREを、FLWR表現式のfor、let、where、returnで表現する言語である。XPathはルートノードから探したいデータを保持するノードへの経路を指定するという形で問合せをする言語であり木構造上を探索するものである。XQuery、XPath共にデータを更新することは出来ない。XPathは単一のXML文書の構造と値を検索するためのものであり、複数の構造が異なるXML文書を検索することを目的としていない。そのため複数のXMLを検索するためには、各XMLにXPathをそれぞれ用意する必要がある。

XML の木構造表現を択える方法として DOM (Document Object Model)や SAX（Simple API for XML）がある。DOM は、対象となる XML 文書全体を読み込み、オブジェクトツリーを生成する。SAX は、XML 文書の先頭から一行ずつ順番に処理を行う方式で、文書全体をメモリ展開せずに処理を行えるため高速に処理を行うことが出来る。SAX は上位ノードへの移動が出来ないため本研究ではノードウォークが出来る DOM を用いる。また XQuery は表現式が基本単位として、複数の組み合わせを記述する関数型言語であり、ノードの位置を特定するだけであれば XPath に比べ検索式が煩雑になるため、検索式は XPath を用いる。

複数の XML 文書に対する検索は、構造が固定化されているのが一般的であるが、XML 文書量の増加に伴い同一内容で複数の構造を持つ場合がある。本研究では上記の現状に着目し、複数の構造を持つ XML に対して有効な検索方式を考える。

本研究は簡単で平易な検索式で、予めデータ構造の知識がなくても、ユーザの検索意図を反映した検索結果が得られるような XML 検索方式を提案する。

構造の知識なしで、検索キーワードを列挙し、Web 検索のようにユーザ所望の XML 文書サブツリーを導き出す。XML 文書内の検索対象ノードをユーザが指定し、検索式（XPath）を自動的に生成する。同種の検索セマンティクスには述部のキーワードを変更するだけで対応する XPath として再利用が可能である。

2. 関連研究

XML 文書検索は XPath や XQuery などの構造に基づく問合せと、キーワードベースの全文検索に大別することができる。本研究でも課題とする構造を把握して

いない XML に対する検索する方法として下記の 3 つの研究があげられる。3 つの研究の共通点は、半構造の XML に対して容易な検索方法を提案している点である。

Amer-Yahia らは、構造走査とキーワード走査を融合させた XML 検索方法を提案している[1]。XPath による厳密な構造指定では、ユーザ所望のデータを検索結果に全て含ませることができない場合がある。この問題に対して段階的に制約を緩めることにより近似的アプローチで検索漏れを減らそうとしている。近似による検索結果が厳密な構造指定による検索結果を完全に含むような制約の緩和方法を提案しており、キーワードマッチを XML 文書の上位階層に広げる構造近似とキーワード間に指定される述語をより弱いものに置き換えたり、削除したりすることで検索結果が包含されるような近似を実現している。

軽量なセマンティクスのみでコンテンツの生成ができる XML に対して、値のマッチと構造のマッチを融合した検索を目指している点は本研究と同じである。本研究は、XML 上のどのエレメントがユーザの問合せセマンティクスを満足しているのか、検索結果はどのエレメントの値であるのかをユーザが明示的に指定でき、データ作成者が表現したデータ構造に関する知識と煩雑な XPath を記述することを必要としない。Web 検索のようにキーワードを列挙して構造上の位置と返されるべきデータの意味を XML サンプル上で指定するだけで適切な XML 文書の検索結果を得られるという特徴を持つ。

Yunyao Li らは NaLIX という XML データベースに対する双方向型の自然言語検索インタフェースを提案している[2]。NaLIX はユーザが入力した英文に対して形態素解析技術を用いて品詞を見分け、XML データベースに対する検索式（XQuery）を生成する。キーワードと XML 文書のノードを関連付け検索式を導き出す手法は本研究と同じであるが、自然言語検索を用いるため、より簡素化された検索方法を提案している。しかし取得した品詞が関連付け出来ない場合、問い合わせ語としては破棄されるため、取得した結果はユーザ所望データである可能性は低下する。

Sara Cohen らは XSEarch という XML の知識のないユーザでも容易に検索を行える専用検索言語を用いた検索方法を提案している[3]。指定されたキーワードが存在するか全文検索し一致したフラグメントを取得する。また複数の結果を取得した場合、独自の技術によりランク付けされ順序付けされた結果を返すことができる。複数の XML 文書全体に全文検索を行い対象となる XML 文書を導き出す手法は同じであるが、他の研究と同様、ユーザが明示的にノードを指定しないためユーザ所望データである可能性は低下する。

本研究と上記 3 研究との相違点は、問い合わせ語から候補を取得し再度ユーザに指定させる点である。ユーザの 2 フェーズセレクト方式を用いることにより所望するデータを取得できる可能性が向上する。

3. XML 問合せ

XML 文書から“長崎”に出張した人の名前を取得する例を図 2 に示す。図 1 のサンプル XML 1 － 1 では

/root/出張報告[出張先='長崎']/氏名

と XPath を指定し、1 － 2 では

/root/一覧/出張報告書[場所='長崎']/なまえ

と指定する必要がある。同一の内容を表現する XML 文書でもデータ構造が違えばそれぞれ対応する XPath を用意する必要がある。また複数の XML 文書が同一の構造を持ったとしても要素名が違うだけでそれに対応する XPath を用意する必要がある。

上記のように XML は任意の構造を定義できるため、ユーザにより同一内容のデータを異なる XML 表現で記述することができる。予め標準の構造を用意することも考えられるが、XML 本来の自由度を失い、利便性が損なわれる恐れがある。

XML 文書をデータ構造の知識なしで検索するために Web 検索のように、ユーザにキーワードを列挙させて全文検索技術を用いて検索する方法も考えられるが、列挙されたキーワードを含む XML 文書を収集することしかできない。そこで、収集した XML 文書からどのノードが検索結果なのかを特定できれば XPath を生成することが可能となり、ユーザが XPath を記述する必要がなくなる。例えば月単位で増加する定型的なデータは同じ XML 構造、要素名で作成されるのが一般的である。そのような XML 文書は、生成された XPath を利用できるため、再度 XPath を生成する必要はない。

本研究は、Web 検索のように簡単で平易でユーザが予めデータ構造の知識を必要としない XML 検索方法を提案する。また結果となる XML 文書のノードをユーザが指定することで意図した結果を取得出来る方法も提案する。

図１：サンプル XML（１－１）

```
<root>
  <出張報告>
    <管理者>坂口</管理者>
    <出張先>長崎</出張先>
    <日付>20070130</日付>
    <氏名>坂口</氏名>
  </出張報告>
  <出張報告>
    <出張先>高知</出張先>
    <日付>20070130</日付>
    <顧客名>田中</顧客名>
  </出張報告>
  <会社>
                :
</root>
```

図 1：XML サンプル（１－２）

```
<root>
  <一覧>
    <出張報告書>
      <場所>長崎</場所>
      <日時>20070130</日時>
      <なまえ>坂口</なまえ>
    </出張報告>
    <出張報告書>
      <場所>高知</場所>
      <日時>20070130</日時>
      <なまえ>田中</なまえ>
    </出張報告>
          :
  </一覧>
</root>
```

3.1. 構造の走査と問合せ語の走査

一般的に XML 文書を検索するには、特定のノードまでの経路を指定するが構造の知識がない場合、ノードの特定を行うことは難しい。そこでノードを特定するために下記の処理を前処理として行う必要がある。

- 検索対象となる XML 文書の絞込み
- 検索対象 XML の構造を把握

上記の処理を行うには全文検索技術を用いれば可能である。ただ全文検索技術だけを用いてユーザが意図する検索結果を XML 文書から取得するのは難しい。全文検索技術は、列挙された問合せ語が存在するかどうかの判別しかできないため、偶然問合せ語が存在した場合、ユーザの意図と異なるデータも検索結果となる。

XML 文書構造を木構造化するには、パース処理を行い DOM オブジェクトとする必要がある。処理が完了した段階で問合せ語がどのエレメント、またはアトリビュートに存在するかをノードウォークして該当するノードが XML 上のどの位置にいるかを特定する。

上記のように該当ノードが１つの場合、ユーザが求めているデータである可能性が高いが、複数のノードが検索結果として求められた場合、システムとしてはどのノードが本当にユーザの求めている情データを判断することはできない。例えば下記の図２のサンプル XML２－１では“坂口”は“管理者”と“氏名”に存在することがわかる。ユーザが目で見れば一目瞭然だが、システムとして判断することは簡単ではない。ユーザの意図するノードを決定することを支援できれば２つの問題は緩和できると考えられる。

まずと問合せ語として“坂口”が指定された場合、ルートノードから順にノードの値を確認していく。するとまず管理者で坂口が存在することがわかる。次に氏名も坂口が存在することがわかる。ここまでの情報で作成できる XPath は下記となる。

/root/出張報告/氏名
/root/出張報告/管理者

図２：XML サンプル（２－１）

```
<root>
  <出張報告>
    <管理者>坂口</管理者>
    <出張先>長崎</出張先>
    <日付>20070130</日付>
    <氏名>坂口</氏名>
  </出張報告>
  <出張報告>
    <出張先>高知</出張先>
    <日付>20070130</日付>
    <顧客名>田中</顧客名>
  </出張報告>
  <会社>
                :
</root>
```

ここでユーザ直接どちらがユーザの意図する問合せ語かを問合せ語の要素名から判断してもらう。XML 文書から特定のノードの位置を把握することができた。これでユーザの意図とシステムの意図が一致することになる。

次にノードが特定したことにより、その対象ノードを求めるパス式を生成することができる。特定のノードから順にルートノードまでの経路をたどることによ

り XPath が生成される。例えば“氏名”の“坂口”が選択された場合、下記の XPath が導き出される。

/root/出張情報[氏名='坂口']

先ほどの XPath と比較すると条件式が追加されたことがわかる。

3.2. 問合せ語間の関係

データの XML 文書表現は作成者により異なる。つまりデータ組の作り方が違うため、XML 表現されたデータのセマンティクスが一意に定まらない。データの表現方法が違えば XML 構造も違ったものとなる。またユーザが求めるデータも、ユーザにより表現が異なる。XML 文書作成者とユーザの間には、データ表現、問合せ表現のギャップが存在する。このギャップを埋めるための、ブリッジが必要である。

ユーザは XML 文書を検索する場合、複数の問合せ語の組み合わせで行うことを前提としている。通常問合せは、AND や OR を用いて問合せ語間の関係を指定するが、Web like な検索を行うのであれば用いることはできない。対象ノードへのパスの組み合わせは、問合せ語間の関係を表現すると考えられるので、パスの組み合わせがあれば関係を導き出せる。これをマッチさせれば検索は行える。

パスを求めるためにはシステム的に行うのは不可能である。何かしらユーザに対して簡単な作業をしてもらうことで複数の問合せ語間の関係を求める方法が必要である。ユーザが簡単で平易な検索を行うためには、負担となるような作業は要求できない。そこで問合せ語を元に指定する候補を絞り込み、ユーザに関係を指定してもらう。ユーザにノードを指定してもらうことで、問合せ語間の関係を把握ことができる。これにより XML 作成者の XML 表現とユーザの XML 表現のギャップをブリッジすることができる。

ユーザ所望のデータを結果として得るためのエレメントまたはアトリビュートを射影することが必要である。

条件に一致する特定のノードを取得する方法を図２：XML サンプルを例に考える。

“管理者が坂口である出張した場所はどこか”という条件で探す方法を考える。答えを導き出す為には表 1:射影ノード候補一覧からユーザが指定する必要がある。

まず“坂口”が XML 文書のどの位置に存在するか、ノードウォークを行い特定する。

すると下記の２つの XPath が求められる。

/root/出張報告/管理者

/root/出張報告/氏名

ユーザは“管理者の坂口”を探しているので“/root/出張報告/管理者”を選択する。また管理者までのパス式が求められたことにより、下記の条件を指定する XPath も合わせて求められる。

/root/出張報告[管理者='坂口']

表１：射影ノード候補一覧

要素名	値
root	
出張報告	
管理者	坂口
出張先	長崎
日付	20070130
出張先	高知
日付	20070130
顧客名	田中

結果として“出張した場所”がどこであるかという情報が取得したいので、要素名の一覧から“出張した場所”がどの要素が意味しているのかユーザに選択してもらう必要がある。今回のサンプルでは要素名一覧は表１：射影ノード候補一覧となる。

ユーザは射影ノード候補一覧を確認すると、“出張先”が“出張した場所”であることがわかる。この出張先が選択されたことにより下記のパス式が求められる。

/root/出張報告/出張先

このパス式を使うことにより同一の構造をもつ XML に対しての検索は構造を把握していなくても全文検索を行わずに特定のノードの情報を取得することができ、取得したい XPath の全体像が明らかになる。

3.3. XPath の生成

前節までに述べた、取得条件を指定し、ユーザ所望データを射影する為の準備である。この準備が完了すると XPath を生成するための情報が全て揃い、プログラムレベルで XPath を生成することが可能となる。

先に検証した、“管理者が坂口である出張した場所はどこか”という条件で求められた XPath は下記の２つとなる。

坂口　：/root/出張報告[管理者='坂口']

出張先：/root/出張報告/出張先

この式からユーザが意図する条件を導き出す方法を考える。２つのパス式を比較すると、

/root/出張報告

が一致することがわかる。違いは下記となる。

坂口　：[管理者='坂口']

出張先：出張先

この２つの違いを組み合わせることで XPath を生成することが出来る。

/root/出張報告＋[管理者=’坂口’]＋出張先

↓

/root/出張報告 [管理者=’坂口’]/出張先

上記はリーフノード以外のパスは同一であり、ツリーの階層も同一の場合であることから、文字列を連結するだけで求められる比較的容易なパターンである。次に考えるのは文字の連結だけでは求められないパターンである。図３XML サンプルを用いて考えてみる。

ユーザが取得したデータは“2007/1/1 に坂口が出張した場所”とした場合の方法を考える。

まず“20070101”が値であるパスの一覧を取得すると下記であることがわかる。

/root/出張報告書/出張者情報/申請日

/root/出張報告書/出張情報/出張日

ユーザは出張した日が欲しいので“/root/出張報告書/出張者情報/申請日”を選択することにより申請日が“20070101”であるノードを求める“/root/出張報告書/出張者情報[申請日=’20070101’]”という XPath が求められる。次に出張した“坂口”を探す必要がある。XML 文書からパスを求めると下記となる。

/root/出張者情報/氏名

/root/出張情報/顧客名

ユーザは出張した人が欲しいため、“/root/出張者情報/氏名”が選択され出張者が“坂口である”ノードを求める

/root/出張者情報[氏名=’坂口’]

という XPath が求められる。

次に出張した場所の情報が欲しいのでユーザに下記の要素名から出張先を選択してもらう。

root、出張報告書、管理者、出張者情報、氏名、部署、役職、申請日、出張情報、出張先、目的、費用、出張日、顧客名ユーザは一覧から“出張先”があることを確認し選択することにより下記のパス式を求めることができる。

/root/出張報告書/出張情報/出張先

次に取得した３つのパスを比較する。

/root/出張報告書/出張者情報[申請日=’20070101’]

/root/出張報告書/出張者情報[氏名=’坂口’]

/root/出張報告書/出張情報/出張先

“/root/出張報告書”まで３つとも同一であることがわかる。次に条件指定する為の“申請日”と“氏名”と取得ノードである“出張先”とのリーフノード以外の違いだが、“出張者情報”と“出張情報”となる。条件式を記述する場合、違いが発生したパスから記述しなければならない。その条件を当てはめると３つの検索式は下記のように二つに分断される。

/root/出張報告書+[出張者情報/申請日=’20070101’]

/root/出張報告書 + [出張者情報/氏名=’坂口’]

/root/出張報告書 + /出張情報/出張先

次にそれぞれを組み合わせると下記の XPath が生成できる。

/root/出張報告書[出張者情報/申請日=’20070101’][出張者情報/氏名=’坂口’]/出張情報/出張先

取得した XPath を用いて検索を行った場合、“東京”が取得できることがわかる。また同一構造の XML 文書であれば、出張者が“坂口”から“田中”に変更された場合でも、下記のように XPath を変更するだけで結果を求めることができる。

/root/出張報告書[出張者情報/申請日=’20070101’][出張者情報/氏名=’田中’]/出張情報/出張先

先に述べた通り、取得した XPath は XML 文書構造が同一であれば、再利用することができる。また欲しい条件が変更された場合でも、条件指定部分を変更することにより対応することができる。

本研究では作成された XPath に名前を付けて履歴保存することにより他の XML 文書に対しても再利用することが可能となる。

図３：XML サンプル（２－３）

```
<root>
  <出張報告書>
    <管理者>山田</管理者>
    <出張者情報>
      <氏名>坂口</氏名>
      <部署>開発</部署>
      <役職>リーダ</役職>
      <申請日>20070101</申請日>
    </出張者情報>
    <出張情報>
      <出張先>東京</出張先>
      <目的>打合せ</目的>
      <費用>10000 円</費用>
      <出張日>20070101</出張日>
      <顧客名>坂口</顧客名>
    </出張情報>
  </出張報告書>
</root>
```

4. 検索システム

4.1. 報告書検索システム

本方式のアプリケーションとして報告書等の比較的構造が簡単な定型データを例にシステムの構築を行った。ユーザはデータの種類（例えば出張に関するもの、備品購入に関するものなど）は把握しているが、XML の構造は把握していないものと仮定している。

XML 文書の登録は、DOM を用いて構造化しておく。これにより XML 文書をシステムから操作することが可能となる。

次に XML 文書への検索式であるが、前章で述べた問合せ語間の関係の表現方法を用いる。これによりユーザは平易に問合せ語間に関係を持たせることが可能となる。

報告書検索システムが有用であるかサンプル XML と複数の問合せ語を用いて検証した場合の画面イメージが図５：報告書検索システムイメージとなる。図５の画面イメージの遷移を元に本システムについて述べる。なお XML データは“図４XML サンプル”を用いる。下記の問合せ結果を求める。

“坂口が 2007/1/1 に出張した目的”

①問合せ入力

まずユーザは条件となる値を列挙すると共に求めたい値の属性名を指定する。この時属性名は XML 文書上のエレメント名と一致している必要はない。ユーザは

“坂口　20070101　＜目的＞”

と入力する。

ここでは戻り値のエレメントと条件に指定される値と区別するため、便宜上“＜＞”で閉じることにより戻り値の値を表現する。

②対象ノード候補一覧

次に“坂口”と“20070101”を含むデータをサンプルとしてユーザに表示する。ユーザはどのエレメントの“坂口”、“20070101”であるかをサンプル上で指定する。これにより生成すべき XPath の条件部分が構成される。

サンプルでは下記の項目が取得できる。

図４XML 報告書サンプル（３－１）の場合

坂口　　: /root/出張報告書/出張者情報/氏名
/root/出張報告書/出張情報/顧客名
20070101 : /root/出張報告書/出張者情報/日付

図４XML 報告書サンプル（３－２）の場合

坂口　　: /root/報告書/出張/名前
20070101 : /root/報告書/出張/日付

ユーザの選択結果から下記の条件式を生成

/root/出張報告書/出張者情報[氏名='坂口']
/root/出張報告書/出張者情報[日付='20070101']
/root/報告書/出張[名前='坂口']
/root/報告書/出張[日付='20070101']

③射影ノードの指定

つづいて“目的”に対応するエレメントであるかを指定する。

サンプルでは下記の項目が取得できる。

図４XML 報告書サンプル（３－１）の場合

root、出張報告書、承認者、出張者情報、氏名、部署、役職、日付、出張情報、出張先、目的、費用、顧客名

図４XML 報告書サンプル（３－２）の場合

root、報告書、管理者、出張、日付、出張先、目的、名前、部署、費用、出張費、仮払い

ユーザの選択結果から射影されたノードまでの XPath を生成。

/root/出張報告書/出張情報/目的
/root/報告書/出張/目的

④XPath の一覧を表示

ユーザ意図が反映された結果が取得できる XPath が表示される。XML 文書に対して XPath 問合せを行う。

サンプルでは下記の項目が取得できる。

図４XML 報告書サンプル（３－１）の場合

/root/出張報告書[出張者情報/氏名='坂口'][出張者情報/日付='20070101'] /出張情報/目的

図４XML 報告書サンプル（３－２）の場合

/root/ 報告書/出張[名前='坂口'][日付='20070101']/目的

⑤検索結果表示

④で実行した XPath 問合せの結果が表示される。構造の異なる XML に対して問合せを行う場合、②～③の作業を繰り返し行う。

⑥終了

図５：報告書検索システムイメージ

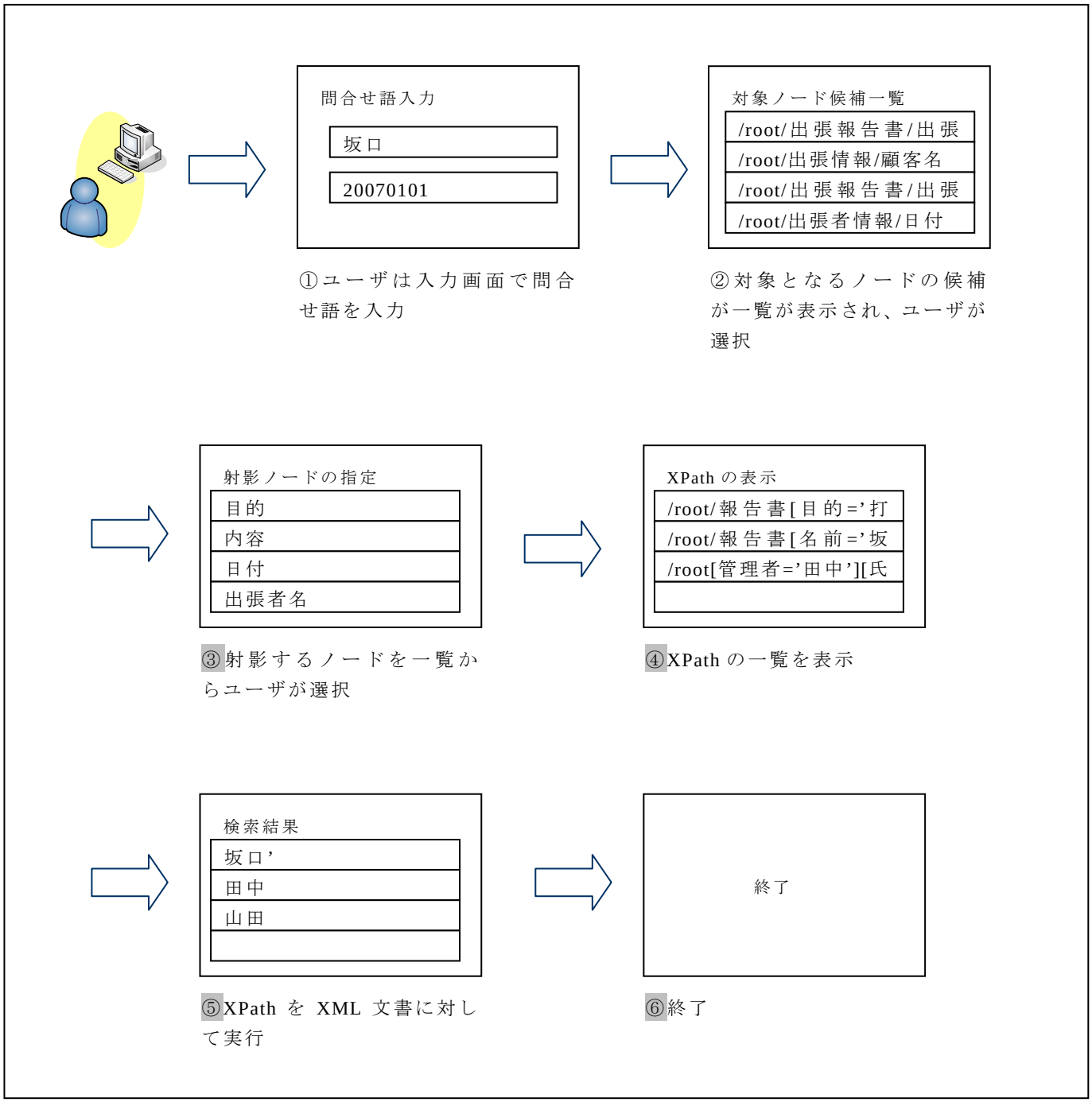

図４：XML 報告書サンプル（３－１）

```
<root>
  <出張報告書>
    <承認者>山田</承認者>
    <出張者情報>
      <氏名>坂口</氏名>
      <部署>開発</部署>
      <役職>リーダ</役職>
      <日付>20070101</日付>
    </出張者情報>
    <出張情報>
      <出張先>東京</出張先>
      <目的>打合せ</目的>
      <費用>10000 円</費用>
      <顧客名>坂口</顧客名>
    </出張情報>
  </出張報告書>
 </root>
```

図４：XML 報告書サンプル（３－２）

```
<root>
  <報告書>
    <管理者>山田</管理者>
    <出張>
      <日付>20070101</日付>
      <出張先>東京</出張先>
      <目的>納品</目的>
      <名前>坂口</名前>
      <部署>開発</部署>
      <費用>
        <出張費>50000 円</出張費>
        <仮払い>10000 円</仮払い>
      </費用>
    </出張>
  </報告書>
</root>
```

5. まとめ

本研究では、半構造データとして XML を用いて検証を行ったが、この技術は XML 文書に限らず HTML や Office 文書にも応用ができる。

ユーザのインタラクションを要求するため、構造が広く深い XML 文書には不向きかもしれないが、リレーショナルデータベースと違いデータ構造の定義を必要としないため、報告書や申請書などの複数の定型化されたフォーマットを持つ軽量なデータには有用な技術である。

文　献

[1] Sihem Amer-Yahia, Laks V. S. Lakshmanan, Shashank Pandit: FleXPath:

flexible structure and full-text querying for XML, Proceedings of the

2004 ACM SIGMOD international conference on Management of data, pp. 83-94, 2004.

[2] Yunyao Li, NaLIX: an Interactive Natural Language Interface for Querying XML

In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2005), Baltimore, MD, June 2005

[3] Sara Cohen, XSEarch: A Semantic Search Engine for XML:

School of Computer Science and Engineering The Hebrew University of Jerusalem Jerusalem 91904, Israel

The 29th International Conference on Very Large Databases (VLDB). September 2003.

[4] W3C , “ Extensible Markup Language(XML)1.0(Third Edition) ” , http://www.w3.org/TR/REC-xml/, February 2004

[5] W3C DOM IG , “ Document Object Model(DOM)”, http://www.w3.org/DOM/, January 2005

[6] David Brownell, SAX2, O’Reilly, January 2002

[7] Tomoharu Asami, “Relaxer Reference Manual”, http://www.relaxer.org/doc/refman/1.0/html/refman_en.html, December 2003

[8] W3C , “ XML Path Language(XPath) Version1.0 ” , http://www.w3.org/TR/xpath , November 1999

[9] W3C, “XSL Transformations Version1.0”, http://www.w3.org/TR/xslt, November 1999

[10] W3C , “ W3C XML Query(XQuery) ” , http://www.w3.org/XML/Query/, January 2006