NAIST-IS-DT9761205

博士論文

XML 文書の部分文書検索に関する研究

絹谷 弘子

2002 年 2 月 5 日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である.

論文番号： NAIST-IS-DT9761205

提出者： 絹谷 弘子

審査委員： 植村 俊亮 教授
松本 裕治 教授
増永 良文 教授
吉川 正俊 助教授

提出日： 2002 年 2 月 5 日

XML 文書の部分文書検索に関する研究*

絹谷 弘子

内容梗概

構造化文書の一種である XML(Extensible Markup Language) は，ネットワーク上のデータや文書の交換手段として利用されるだけではなく，ワープロ文書，電子新聞記事や電子政府の各種文書などの公開文書に利用されるなど，その利用範囲は急速に拡大している．利用範囲の拡大に伴いネットワーク上に流通する XML 文書が急増しているため，必要な情報を効率良く高速に検索できる XML 文書を対象とした検索エンジンの必要性も高まっている．

現在の検索エンジンは，いくつかのキーワードを問合せとして入力し，関連性の高い文書に関する情報を検索結果として表示する．しかし，問合せ結果は，ネットワーク上の流通単位であるファイルであり，たとえ問合せとの関連性が高い部分がファイル中の一部分であったとしても，その部分を対象とした検索や結果の表示を行なえない．従って，XML 文書の一部を抽出し，再利用するためにも文書を部分文書の集合とみなし，部分文書単位の検索が必要である．

本研究の目的は,「利用者に負担のない問合せの入力による適切な部分文書の検索と表示」である．本研究で求める部分文書は，元文書の論理木構造を保持した形の文書とする．各部分木の根ノードと部分文書が一対一に対応する．部分文書検索において，部分文書の文書内容と同様にこの部分文書までの元文書中における論理構造上の位置を示す経路式が表す文脈情報を得ることが重要と考える．XML 問合せ言語では，この経路式を利用者が指定することを前提としているが，文書中の特定の場所を示す経路式を利用者が得る方法がない．利用者が検索結果

* 奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 博士論文, NAIST-IS-DT9761205, 2002 年 2 月 5 日.

の絞り込みや検索結果の再利用をするためにも部分文書の位置を表す経路式と部分文書の内容を利用者に提示することが重要である． さらに，従来の KWIC による表示では，検索結果として入力キーワードの周辺の文を表示することによって利用者がキーワードが使われている文脈を知ることができたが，XML 部分文書検索においても，検索結果として入力キーワードの周辺の部分文書と，その部分の構造上の位置を経路式として表示することが利用者により多くの文脈情報をもたらすはずである．

本論文では，利用者が検索対象となる XML 文書集合の文書構造についての知識を（1）標準語彙 (マーク付け言語）の構造知識を持つ場合,（2）全く持たない場合に分け，それぞれに対応した問合せを（1）構造とキーワードの組,（2）キーワードのみ，としてモデル化し，問合せ条件を満たす最適な部分文書を求めるアルゴリズムを提案し，その有効性を W3C の仕様書を対象とした実験によって検証した．本研究によって利用者が文書構造についての知識がなく，問合せ言語の複雑な構文を知らなくても，利用者にとって必要な文書の一部とその文脈情報を取り出し，表示することが可能であることを示した．

キーワード

XML 文書情報検索, 文脈検索，XML 問合せ言語，構造化文書, KWIC

Studies on Partial Document Retrieval of XML Documents*

Hiroko Kinutani

Abstract

XML (Extensible Markup Language), a kind of structured document, is not only utilized as an exchange format for multimedia data and documents on the network, but is also becoming widely used in word processor documents, e-paper articles, various other kinds of official documents. Due to the rapid increase in the number of XML documents circulating on networks, the necessity has arisen for a search engine specifically for XML documents, which can retrieve the required information efficiently and at high speed. Current search engines require the input of keywords as a query, and then documents of highest relevance are displayed as search results.

However, the query only shows results at the level of single file units, which means that sections of XML documents within those files are not displayed, even if they have a high relevance to the query. In this thesis, the research on the partial document retrieval of XML documents is reported. In order to retrieve a part of an XML document to reuse the information, it is necessary to have a search method that can retrieve query results as units with partial documents regarded as partial document collection originating from an XML document.

The purpose of this research is " Effortless search and display of an appropriate partial document to a user following the input of simple keywords". An XML

* Doctor's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9761205, February 5, 2002.

document is modelled as a labelled rooted tree. We adopt a logical model of XML partial documents as a subtree whose root node corresponds to an element node in an XML document tree. Each root node of a subtree of a logical tree structure of an original XML document corresponds to a partial document, one to one.

In partial document retrieval, it is important to get context information and path expression to show the position on the logical structure of an original XML document. When an XML query language is used, this path expression is selected by a user declaratively, but there is no consistent method to assist the user in getting this path expression to point to the specific position in the original XML document.

Furthermore, as a user is able to know the context in which a keyword is used by displaying a peripheral sentence of an input keyword as search results through am indication of KWIC. As a result, the context of the information is now possible to be given by the user in the XML partial document retrieval by displaying a position on a logical tree structure of an input keyword and the relevant part as search results.

In this thesis, a method for partial document retrieval is proposed under the two premises. (1) When a user has structural knowledge for Standard Vocabulary (Markup Language), and (2) when a user has no structural knowledge at all. In addition, the query model and partial document retrieval model are defined. The query is premised consisting of (1) structure information and keyword, and (2) keyword alone. Then I propose an algorithm to retrieve the most suitable partial document which satisfied a query's condition, and the effectiveness of this method is confirmed by an experiment using specifications of W3C XML files.

Keywords:

XML Information Retrieval, Context Search, XML Query Language, Structured Documents, KWIC

目 次

図 目 次

表 目 次

第1章

序論

1.1 研究の背景と動機

拡張可能なマーク付け言語 XML(Extensible Markup Language)[28, 51, 56] は，ネットワーク上のマルチメディアデータや文書の交換手段として利用されるだけではなく，ワープロ文書，電子新聞記事や電子政府の各種文書などその利用範囲は急速に拡大している．ネットワーク上に流通する XML 文書の急増により，必要な情報を効率良く高速に検索できる XML 文書を対象とした検索エンジンへの必要性も高まっている．

現在のネットワーク上の文書に対する検索エンジンは，いくつかのキーワードを問合せとして入力し，ネットワーク上の文書中の文字列から構築された索引を利用して関連性の高い文書に関して存在する場所，要約などを順位を付けて検索結果として利用者に表示する．利用者は，そのリストから有用と思われる文書を呼び出して，内容を読むことになる．しかし，問合せ結果は常にファイルを単位としていて，問合せとの関連性が高い部分がファイル中の一部である場合は，その部分だけを対象とした検索や結果の表示を利用者は行なえない．従って，XML 文書の一部を抽出し，再利用するためにも文書を部分文書の集合とみなし，部分文書単位の検索が必要である．今後，XML 文書の増大により，XML 文書の特性を生かした検索エンジンでは，XML 文書の一部分を指定して取り出すことが可能であり，問合せ結果の粒度も柔軟に変更可能であるべきである．

構造化文書に対する文書検索の研究は，主に SGML 文書 [25, 26, 27] を対象に行なわれてきた．特に SGML 文書の利用が電子図書館や電子出版などを対象と

し，念入りな設計のもとに構築された論理的な文書構造を重要視していたため，すべての文書が同じ論理構造定義に従った文書構造の固定された XML 文書集合を対象として，その効率的な生成，格納，検索方法に関する研究が多く行なわれていた．しかし，ネットワーク上に手軽に情報発信する手段として HTML(Hyper Text Markup Language) 文書が急速に発達，普及したことにより，HTML が行なえない独自タグによる自由な拡張が可能な XML は，あらかじめその論理構造を設計せずに，必要に応じて構造を柔軟に変化させる文書を出現させている．また，特定分野のプロトコルとして XML の応用マーク付け言語（XHTML, SMIL, VoiceXML, SOAP, SVG など）が W3C(World Wide Web Consortium) 等から多数提案されている．これらの応用マーク付け言語は，XML 文書全体の構造を定義するのではなく，特定の用途に限ったタグについて標準化したもので，必要に応じて XML 文書中にとりいれ，応用プログラムが理解し処理ができるように設計されている．従って，例えば一つの XML 文書中に SMIL を記述するタグと SVG を記述するタグを XML 名前空間 [53] を利用してそれぞれの意味において識別可能なモジュールとして共存させることができる．

このように，従来の文書型定義 (DTD) あるいは XML スキーマ [48, 49, 50] に従った妥当な XML 文書に加え，応用マーク付け言語をモジュールとして取り込んだ XML 文書や DTD を持たない整形式の XML 文書の出現により，これらの構造化文書集合に対する新たな管理技術への要求が拡大してきている．本研究では高度に構造化された XML 文書から半構造データまでの多様な形式の XML 文書に対応した検索エンジンの要素技術として部分文書検索を考察する．

図 1.1 は，様々な文書構造を持つ XML 文書とその管理の概要である．XML 文書には，DTD や XML スキーマによって完全に構造化された妥当な XML 文書と DTD や XML スキーマによる制約のない整形式の XML 文書がある．中間には，XML の応用マーク付け言語や業界ごとにマーク付けの標準として認知された標準語彙[1] を名前空間として指定した要素型や属性として持つ文書もある．文書構造が固定されている妥当な XML 文書集合は，従来の SGML 文書を対象とした

[1] 本論文における語彙は，XML 文書中に利用される要素名，属性名の集合を表す．特に，特定の意味付けを行ない，利用者が共通の認識を持つ要素名，属性名の集合を標準語彙と呼ぶ．情報検索における統制語彙の意味とは異なる．

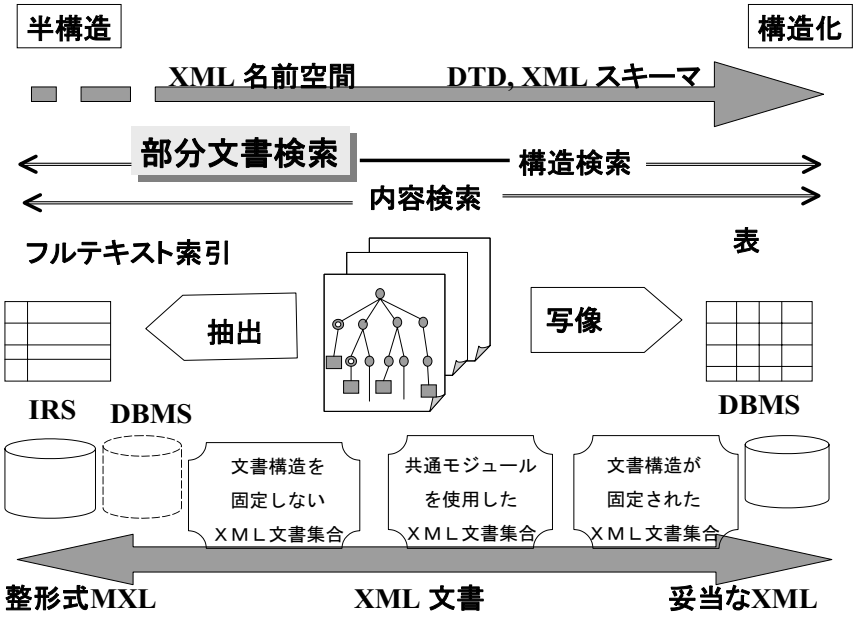

図 1.1 高度に構造化された XML 文書から半構造データまでの多様な XML 文書とその管理

データベースと同様に XML 文書をデータベースで管理することが容易なため，論理構造をデータベースの表やオブジェクトに写像する手法が可能である．一方，文書構造が全く固定されていない XML 文書集合には，従来の SGML 文書データベースとは異なった管理手法を必要とし，XML 文書に特化したデータベースや従来の関係データベースやオブジェクト指向データベースに XML 文書のための拡張を必要としている．また，XML 文書集合への検索において，検索条件や検索結果部分の指定をデータベースの問合せ言語を利用して行なう場合は，利用者があらかじめ文書構造についての知識を持つ場合を想定している．そのため，利用者が文書構造についての知識を持たない場合に，検索条件や検索結果についての曖昧性を問合せ言語中に記述する方法がないのが現状である．

従来の XML 文書検索技術で未達成な点は次の通りである：

1. 利用者が欲しい文書や文書構造について完全な知識がない場合にも問合せできる方法:
   利用者があらかじめ文書の持つ論理構造を知って問合せることは大変難しい．特に XML 問合せ言語が要求している文書構造を指定する経路式表現を利用者が入力するのは非現実的である．しかし，すべての XML 文書と文書中の構造へのアクセスは，XML 文書の位置を識別する URI と文書中の各構造までを識別する経路式によって一様に行なうことが可能であることは，XML 文書の特性である．検索システムが XML 文書の URI と経路式を計算することによって，利用者は，自分の必要とする情報の存在する XML 文書の URI と経路式によって部分文書にアクセスすることができる．
2. 利用者に負担のない問合せの入力方法 :
   現在利用されている検索エンジンにおいて利用者が入力するキーワード数は 2〜3 である．また XML 文書では，文書中の構造を表す要素型や属性を指定することで検索対象部分を絞り込むことが可能である．XML 文書の特性を生かした問合せは少数の良く知られたタグ名とキーワードを入力とするのが妥当だと考える．
3. 利用者の欲しい文書部分を探す方法 :
   利用者にとって必要な情報の粒度は，常に一定ではないはずである．検索結果は，問合せに関連する部分が必要なので，問合せに関係ない部分を取り除くことで，より利用者の問合せに関係の強い文書部分に焦点を当てる方法が必要である．

また Brewer が今後のサーチの注目点として文脈に基づく検索について次のように述べている [7].

> 「利用者の問合せの意味は，誰がいつどこで問合せたのかという文脈に左右される．例えば，たった二つの単語しか指定されていなかった場合,「ぴったり」の文書を探さなくてはならないとする．日常生活のどのようなやりとりでも，特定の文脈が暗黙に前提とされている．

そのため検索においても，何らかの文脈情報が必要となる．しかし，今まで文脈情報に基づいて関連性を識別するという検索エンジンの実践経験がほとんどない.」

この文脈に基づく検索のために，XML 文書内に記述されている構造情報は，有用だと考える．

本研究で求める部分文書は，元文書の論理木構造を保持した形の文書とする．各部分木の根ノードと部分文書が一対一に対応する．また，経路式は，元文書の根ノードと部分文書の根ノード間の経路中のノードのラベルの並びでもあり，元文書における部分文書の構造上の文脈を表している．部分文書検索において，部分文書の文書内容と同様にこの部分文書までの経路式が表す文脈情報を得ることが重要と考える．XML 問合せ言語では，この経路式を利用者が指定することを前提としているが，この経路式自体を利用者が得る方法がない．本研究で提案する部分文書検索は XML 文書における KWIC(KeyWord In Context)（[4, p.290）と考えられる．従来の KWIC は，検索結果として入力キーワードの周辺の文を表示することによって利用者がキーワードが使われている文脈を知ることができた．同様に，XML 文書での KWIC は，検索結果として入力キーワードの周辺の部分文書と，その部分の構造上の位置によって利用者により多くの文脈情報を与えることが可能となる．本研究の部分文書検索は，XML 文書構造における文脈情報として，Brewer が述べる検索エンジンに必要な要素技術と考えられる．

例えば，利用者の「タイトルが “XML” である部分文書を検索せよ」という問合せに対し，結果が，

```
{‘‘<program><title>XML</title></program>’’,
    ‘‘<book><title>XML</title></book>’’}
```

という部分文書が得られる場合，利用者は，前者が元文書の経路式 `/conference/workshop/program` から会議のワークショップのプログラムのタイトルであり，
後者が `/library/computer_science/book/` から図書館のコンピュータ科学の本

のタイトルであるということを知る必要がある．さらに，前者の部分文書中には，プログラムの議長や概要に関する情報も一緒に含まれ，後者の部分文書中には，本の著者や出版社に関する情報も一緒に含まれることが望ましい．本研究の目的は，この例のような XML 文書中に記述されている内容を利用して，利用者にとって最適な部分文書の構造上の文脈情報と文書内容を求めることにある．

本研究では，部分文書検索を共通のモジュールを持つ XML 文書集合，より一般的な XML 文書集合に対して問合せモデルと部分文書検索モデルを定義し，それぞれの場合の部分文書検索手法を提案する．前者は，共通モジュール中に利用されている要素や属性を構造情報としてキーワードと共に問合せに利用する場合の部分文書検索について，後者は，キーワード入力からの部分文書検索について述べ，各々のアルゴリズムを検証し，有効性を確認した．

1.2 研究の概要

本研究では，XML 文書の部分文書検索に関する研究における二つの着眼点に関して，それぞれ次のような主な成果を得た．

(1) 文書構造とキーワード指定による部分文書検索 [32, 60, 61]

i) 問合せ中の述語を満たす部分文書を限定するための文脈ノードの導入．この文脈ノードを根とする部分木に対応する部分文書中には，少なくとも問合せ中の述語を満たす部分を 1 箇所以上含み，文書構造上一つの文脈情報の単位となっていることが確認できた．

ii) 部分文書検索における単純問合せ，AND 問合せと OR 問合せの意味づけと定義．単純問合せは，問合せ述語を満たす構造上の同一文脈を保持した部分を求める． AND 問合せでは，複数の述語を同時に満たす部分文書として，各述語を満たす文脈ノード集合の共通部分に対応した部分文書集合を結果とする．また，OR 問合せは，各述語を満たす文脈ノード集合の和集合に対応した部分文書集合を結果とする．その結果，AND 問合せでは，各述語を満たす粒度の小さい部分文書を取り出

すことができた．またOR問合せでは，各述語を満たす単純問合せ結果の部分文書集合の和集合として，指定した構造は，各文書内に1回以上出現するが，その出現回数は，そう多くならないことを確認した．

(2) キーワード指定による部分文書検索 [62, 63, 24]

i) 文書構造上検索対象とすべき部分文書の特定手法．入力キーワードは，文書内容中の文字列と比較されるものなので，文書内容中の各文字列に対応した部分文書をXML文書を走査することによって特定する．その結果，各文書内容に対応した文書構造上，同一文脈である部分文書を検索対象として保持することが可能である．

ii) 情報検索手法による部分文書抽出法の提案．問合せ中のキーワードは，部分文書中でも1回以上出現している．しかし，出現回数の多いキーワードも存在し，AND, OR問合せでは，問合せ結果が多すぎたり，問合せ結果の検証が困難な場合がある．そのため，問合せ結果に問合せとの関連度を計算して，順位付けを行なう必要があり，実験では，拡張ブーリアンモデルによって検証した．

1.3 論文構成

本論文は，以下のような構成である．第2章では本研究で扱う拡張可能マーク付言語であるXMLについてその概要を述べ，本研究における部分文書検索に関する基本事項を定義する．第3章では，構造とキーワード指定による共通のモジュールを同じ名前空間の語彙として持つXML文書集合における部分文書検索について述べる．利用者が共通のモジュールに関する要素名や属性を手がかりとして，これらが利用されている文書部分を求める手法を提案する．第4章では，キーワード指定によるXML文書集合における部分文書検索について述べる．利用者が文書構造についての知識がない場合に従来のWeb検索エンジンと同様に入力キーワードから関連性の高い部分文書を求める手法を提案する．第5章では，本研究の結論と今後の課題について述べる．

第2章

XMLの概要

本章では，ネットワーク上の文書やデータを統一的に表現するための標準として制定された XML(Extensible Markup Language:拡張可能なマーク付け言語) [28, 51, 56] と XML 文書集合に対する検索，本研究における部分文書検索の基本事項について述べる．2.1 節では，XML 文書とその構成要素について，2.2 節では，XML 文書に対する検索要求とその分類について，2.3 節では，本研究における部分文書検索について述べる．

2.1 XML文書とその構成要素

本節では，XML 文書とその構成要素について本研究に関連する部分の概要を記述する．

XML 文書

XML で記述されたオブジェクトを XML 文書と呼ぶ．XML 文書中の文字列は，開始タグと終了タグで囲まれる．各要素型は，属性を持つことができ，属性名と属性値の対で表す．このタグの入れ子構造が文書の論理構造を表す．XML 文書は，XML 宣言，文書型定義 (DTD) と XML 文書内容で構成される．XML 宣言と DTD は必須ではない．図 2.1 は，XML 文書とその構成要素を表す．XML 文書は根付き木構造でモデル化され，構造情報（経路式）と文書内容情報で構成される．図 2.2 は，本の XML インスタンス例で，図 2.3 はこの XML 文書の DTD で

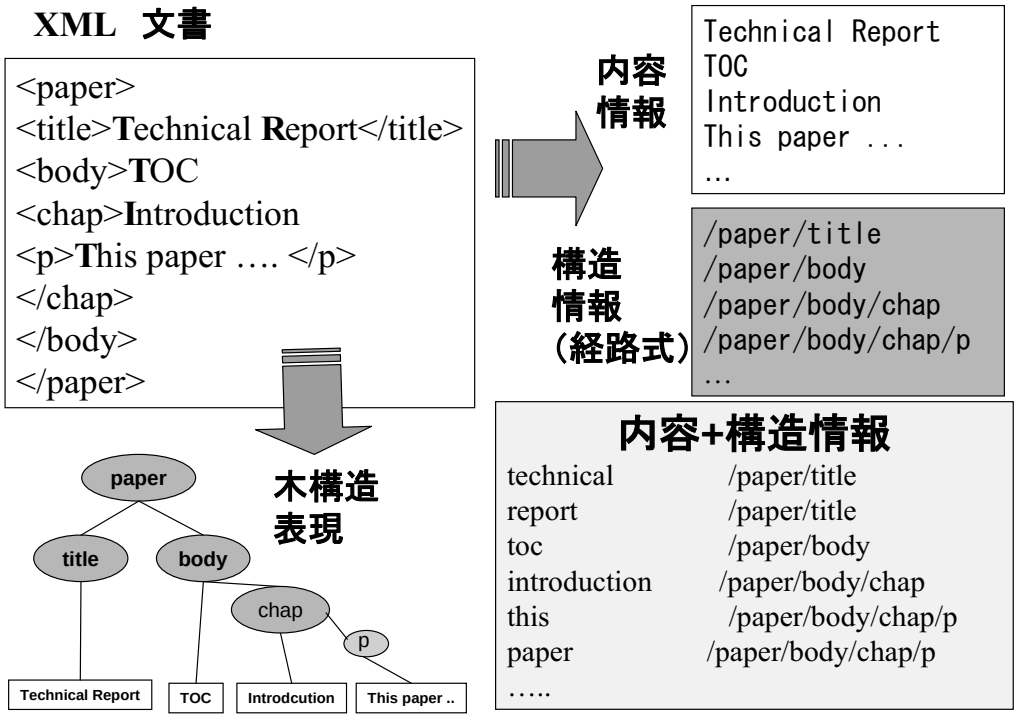

図 2.1 XML 文書とその構成要素

あり，左側の数字は便宜的な参照のための行番号である．図 2.4 はこの XML インスタンスの木構造表現である．

DTD

DTD は，要素型およびその階層構造と要素型の付属情報としての属性の宣言を行ない，各 XML インスタンスのタグ付けの制約を記述する．要素型宣言では，要素型名，要素型の階層構造（親子関係，兄弟関係）と要素の内容を定義する．要素内容として子供として出現する要素に関する基本情報として子要素型の出現順序，出現回数を指定する．そのほか，属性リスト宣言とエンティティ宣言を持つ．図 2.3 は，DTD の例で 1〜3 行は要素型宣言である．DTD は，SGML[25, 26, 27] から引き継いだ文書構造の定義方法で，DTD に従った XML 文書を妥当な XML

```
1  <?xml version="1.0"?>
2  <!DOCTYPE book SYSTEM ‘‘chapter.dtd’’>
3  <chapter>
4     <title>Data Model</title>
5     <section>
6        <title>Syntax For Data Model</title>
7     </section>
8     <section>
9        <title>XML</title>
10    <section>
11       <title>Basic Syntax</title>
12    </section>
13    <section>
14       <title>XML and Semistructured Data</title>
15    </section>
16    </section>
17 </chapter>
```

図 2.2 本の XML インスタンス例 : chapter.xml

```
1    <!ELEMENT chapter (title, section*)>
2    <!ELEMENT section (title, section*)>
3    <!ELEMENT title (#PCDATA)>
```

図 2.3 chapter.xml の文書型定義 (DTD) : chapter.dtd

文書と呼ぶ．妥当な XML 文書は，DTD で定義された文書構造についての制約を満たしている．また W3C から新たに XML スキーマの仕様 [48, 49, 50] が勧告され，XML 文書内容にデータ型を定義できるなど，よりきめ細かに XML 文書の構造と内容に関する制約を記述できるようになってきた．しかし，普及するにはまだ時間がかかると予想される．DTD を持たない整形式の XML インスタンスも多数存在している．これらは，文書内部には，タグの入れ子構造として表現されている論理構造を持つが，その論理構造を宣言的に指定していない． 本研究では，DTD に従った妥当な XML インスタンスだけではなく整形式の XML インスタン

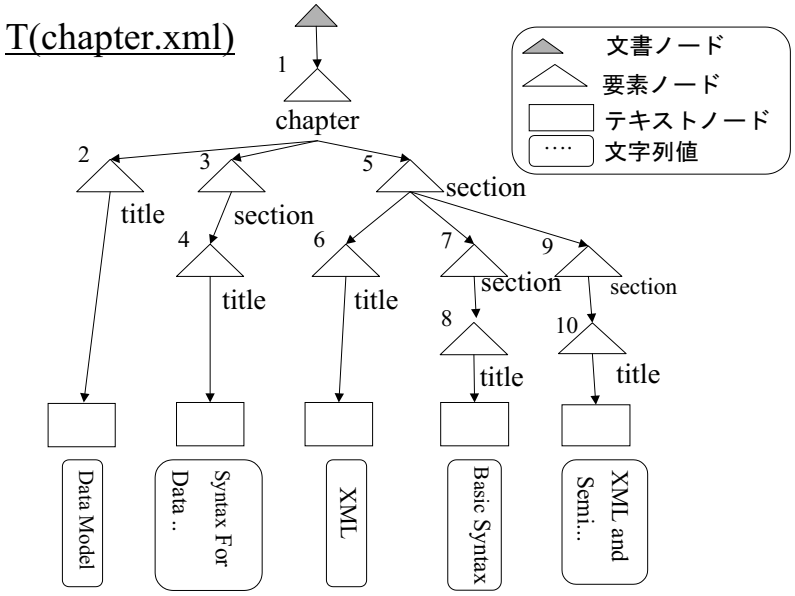

図 2.4 chapter.xml の木構造表現

スも検索対象文書とする．

XPath データモデル

本研究では，XML インスタンスの論理構造を XPath[55] データモデルを用いて表す．XPath データモデルでは，XML 文書 d をノードの根付き木 $T(d) = (d$ 中のノード集合, d 中の枝集合, d の根 $(root(T))$) として扱う．八種類のノードの中で，ここでは文書ノード，要素ノード，属性ノードとテキストノードに絞って論じる[1]．XML 文書中に展開可能な外部実体として，他の XML ファイルを取り込むことができる．従って，一つ以上の XML ファイルから XML 表現が作られる．XML 文書内のすべてのノードで定義する文書順という順序は，展開可能な実体を展開後の XML 表現において各ノードの XML 表現の最初の文字が現れる順序

[1] 他の名前空間ノード，処理命令ノード，参照ノードとコメントノードは，本研究においては本質ではないので省略する．

表 2.1 chapter.xml の経路式と文書内容

構造情報（経路式）	文書内容
/chapter	
/chapter/title	Data Model
/chapter/section[1]	
/chapter/section[1]/title	Syntax For Data Model
/chapter/section[2]	
/chapter/section[2]/title	XML
/chapter/section[2]/section[1]	
/chapter/section[2]/section[1]/title	Basic syntax
/chapter/section[2]/section[2]	
/chapter/section[2]/section[2]/title	XML and Semistructured….

を表したものである．従って我々はこのように実体を展開後の XML 表現を対象とし，以後 XML 文書と呼ぶ．その結果，論理構造としての XML 文書は一つ以上の物理構造としてのファイルで構成される．根付き木の基本的な用語の定義は，付録 A.1 を参照されたい．

1. **文字データ**: 各ノードには，文字列値を決定する方法がある．テキストノードの文字列値は，文字データである．根ノードは木構造の根であり，文字列値はすべての子孫テキストノードの文字列値を文書順に連結したものである．文書要素型の要素ノードは，根ノードの子である．展開後の XML 表現の要素型はすべて要素ノードを持つ．要素ノードの子として，その要素型内容の要素ノード，テキストノードを持つ．また要素ノードの文字列値は，要素ノードのすべての子孫ノードの文字列値を文書順に連結したものである．要素ノードは，関連する属性ノードの集合を持ち，これら属性ノードの親になるが，属性ノードは，要素ノードの子ではない．さらに要素ノードは，展開された

名前を持つ．展開された名前は，名前空間の URI（あるいは null）と局所的な名前を表す文字列からなる．各属性ノードは正規化後の文字列値を持つ．従って，直接文字データを持つノードは，属性ノードとテキストノードだけである．

2. **要素ノードと属性ノード**：一般に属性ノードが持つ文字データは，属性名と属性値の組をデータベースで管理可能な値として扱うことが可能である．一方テキストノードが持つ文字データは，自然言語で書かれた文として扱うことができる．
   しかし，DTD を設計し，DTD に従った XML 文書を作成する場合は，要素型と属性の扱いに明確な方針を持つが，DTD を持たない整形式の XML 文書では，要素型と属性の扱いは様々であり，使い方に共通認識が得られていない．従って，本論文では，属性ノードとテキストノードが持つ文字データに差がないものと仮定する．

構造情報（経路式）

XML 文書中の各文字列を囲んでいる要素名，属性名の並びを文書の論理木構造の根から順に並べたものを経路式とよぶ．すべての内容情報に対応した経路式が文書構造を解析すると得られる．この経路式は，次に述べる XPath 式における絶対ロケーションパスで表現することができる．

XML 文書中の特定の要素や属性などの位置を示すのに XPath 式が用いられる [55]．また，その設計は XML 文書の論理構造を航行する場合の経路式表現と同様の記法に基づいている．ここで XPath 式におけるロケーションパスについて XPath 1.0 の仕様に従って紹介する．XPath 式を評価する際の XML 文書中のノードをコンテキストノードと呼ぶ．ロケーションパスには，相対ロケーションパスと絶対ロケーションパスがある．相対ロケーションパスは，/ で区切られた一つ以上のロケーションステップを並べたものである．この相対ロケーションパス内の各ステップは，左からコンテキストノードに対しての相対的なノード集合を順次選択する．一方絶対ロケーションパスは，/ とその後に続く相対ロケーションパスを組み合わせたものである．この/ は，コンテキストノードを含む文書の根

ノードを意味する．

XPath 式は，このコンテキストノードに対して評価される．ロケーションパスの構文には，省略しない構文と省略構文がある．次に示す構文はその代表的な例である．

- / は文書要素の親である文書根ノードを選択する．
- 一般的に **基準点名::ノード テスト** [**述語**][**述語**] **....** という構文でロケーションステップを記述する．例えば `child::para[position()=1]` では，基準点名 `child`, ノード テスト `para`, 述語 “`[position()=1]`” である．
- `*`, `@*`は，それぞれコンテキストノードのすべての子要素，すべての属性の選択の省略構文である．
- . は，コンテキストノードを選択する省略構文である．
- `//`は，`descendant--or--self::node()/`の省略構文である．
- .. は，`parent::node()` の省略構文である．

以後，XML 文書中の位置の指定は，このロケーションパスを利用する．
次に示すのが要素型を指定した XPath 式の例である。

(1) XML 文書中のすべての number 要素ノードを抽出する．

```
//number
```

(2) 単位の属性 units の値が “Yen” であるようなすべての book 要素ノードを抽出する．

```
//book[@units=‘‘Yen’’]
```

(3) doc の 5 番目の chapter という名前の子要素の 2 番目の section という名前の子要素を抽出する．

```
/doc/chapter[5]/section[2]
```

表 2.1 は，図 2.2, 図 2.4 の XML 文書の論理木構造中の各ノードに対応した経路式と文書内容の対応である．各ノードは，このロケーションパスを指定することで，XML ファイル中からこのノードの文書内容を取り出すことができる．しかし，同じノードを表すロケーションパスは何通りもあるため，文書中で他のノードと識別可能なロケーションパス表現は，(3) に示した文書ノードから該当ノードまでのインデクス付の絶対パス表現である．ただし，インデクスが省略されている場合は，同名の兄弟ノードが存在しないことを意味している．本研究では，このインデクス付の絶対パス表現によって文書中の各ノードを表すことにする．

ただしこの絶対パス表現と根付き木モデル (付録 A.1) における親子兄弟関係を表す経路表現とは，一致しない．絶対パス表現でのインデクスは，同名のラベルを持つ兄弟ノードの順番を表していることに注意する必要がある．

内容情報

XML 文書中の構造を表している文字列を除いた部分．文書内容の分析には，この内容情報を利用する．従って，内容情報は従来のプレーンテキストに該当する．

XML 名前空間

ネットワーク上で XML 文書を交換したり流通させる場合，一般的に文書中の要素型名，属性名の衝突が起こる危険性を含んでいる．XML 文書は要素型名，属性名を独自につけることができるため，XML 文書集合中で同一文字列の要素型名が異なったスキーマ制約に従っていたり，独自の意味で利用している場合もある．このような名前の衝突を回避することを目的として XML 名前空間の仕様 [53] が W3C の勧告となっている．XML 名前空間では，XML データモデルを拡張して，URI(Uniform Resource Identifier) で要素型名と属性名を限定する標準を提案している．

図 2.5 が電子オークションの XML 文書例である．この XML 文書では，5 行目と 15 行目で名前空間を利用することを宣言している．5 行目では，XML スキー

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>

2  <AuctionWatchList>
3   <Auction ID="0321K372910">
4   <Schedule>
5     <Open  xmlns:dt="http://www.w3.org/1999/XMLSchema-datatypes"
6               dt:type="timeInstant">2000-03-21:07:41:34-05:00</Open>
7     <Close dt:type="timeInstant">2000-03-23:07:41:34-05:00</Close>
8   </Schedule>
9   <Price>
10     <Start currency="USD">3.00</Start>
11     <Current currency="USD">10.00</Current>
12     <Number_of_Bids>5</Number_of_Bids>
13   </Price>
14   <Details>
15     <record xmlns="http://www.example.org/music">
16        <artist>Miles Davis</artist>
17        <title>In a Silent Way</title>
18          <remark> The liner notes were written by Frank Glenn.</remark>
19     </record>
20    </Details>
21   <Auction ID="13143816">
        :
        :
30       <Details>
        :
50       </Details>
51   </Auction>
52  </AuctionWatchList>
```

図 2.5 電子オークションの XML 文書例:auction.xml

マで定義されているデータ型を利用することを名前空間で宣言し，接頭辞dtで識別する．これらの接頭語のついた要素型や属性は，これらの名前空間を参照している他の XML 文書と共通である． 一方，接頭語はないが，15 行目で<record>内の要素型が属する名前空間を指定している．従って，この文書の階層構造の下部構造の一部，5〜6 行目，15〜19 行目に他の XML 文書と共通な構造を持つ性質がある．

2.2 XML 文書に対する検索要求とその分類

XML 文書は，従来の構造化文書の一種として従来と同様の文書検索だけではなく，データベースのデータをネットワーク上で交換するためのフォーマットとして利用されるなど，その利用範囲は急速に拡大している．本節では，XML 文書の様々な側面が必要としている検索要求について述べる．

2.2.1 データ指向 XML と文書指向 XML

XML は文書とデータという二つの異なった性質を持っている． データとしての性質を持つ XML インスタンスはデータベース属性と値を記述して, プログラムによる解析，自動化を重視する． データベースに対する操作（データの入力，更新，削除，検索）を利用者が問合せ言語（例：SQL）を使って行なうように，XML のデータ操作を利用者が問合せ言語を使って行なうことを目的として XML 問合せ言語が多数提案されている．一方，文書としての側面では，構造と位置や順序を重視する．従来の文書処理において文書操作を宣言的に行なう問合せ言語は存在せず，現状の問合せは，構造名とキーワード の羅列もしくは，論理結合の場合がほとんどである．本論文では，データ指向 XML を XML データ，文書指向 XML を XML 文書と呼ぶことにする．

2.2.2 XML データ検索に対する要求

XML 問合せ言語はこれまでに多数提案されている [6] が，こららは主にデータ指向の問合せに便利な言語として設計されている．本節では，2001 年 6 月に W3C にワーキングドラフトとして提案された XQuery[58] を取り上げて，データ検索が必要とする文書構造と内容に関する問合せについて述べる．

XQuery は，今までに提案された Quilt[17] を継承し，文書の論理階層構造を表現する経路式表現の構文は，XPath[55] と XQL[40] から採り入れ，XML–QL[15] から変数束縛と新しい要素の構築を採り入れている．また，SQL からは，データの再構築のためのキーワード に基づく一連の句 (SELECT–FROM–WHERE) の

使い方をとり入れている．従って，現時点では XML 問合せ言語として，宣言的に選択条件，新規要素条件，検索結果条件を与えることができる．

XQuery は，XML スキーマの仕様 [48, 49, 50] に基づく型システムであり，名前空間`http://www.w3.org/2001/XMLSchenmas` を利用して基本型と派生型を問合せ中に利用することができる．Path 式の中には，単純データ型で定義されている演算子を記述したり，ビルト イン関数やユーザ定義関数を利用することもできる．特に関数`data` によってそのノード の内容を型付された値に変換することができる．

XQuery を利用した XML 文書の構造と内容に関する問合せの例を示す．XML 問合せ言語を利用することによって，構造に関する問合せと内容に関する問合せを行なうことができる．前者は，文書構造に関する条件を入力して必要な構造を求め，後者は，文書内容に関する条件を入力して必要な構造を求める．XML 問合せ言語では，両者を組み合わせた問合せもできる．

(1) 構造に関する問合せ

```
document(‘‘zoo.xml’’)//chapter[2 TO 5]//figure
```

“zoo.xml” 中の 2 章から 5 章までの間にあるすべての`<figure>`を検索する．

(2) 内容に関する問合せ

```
FOR $b IN document(‘‘bib.xml’’)//book
WHERE $b/publisher = ‘‘Morgan Kaufmann’’
AND $b/year = ‘‘1998’’
RETURN $b/title
```

1998 年に Morgan Kaufmann で出版されたすべての本のタイトルをリストする.

XML の論理木構造をそのまま永続化させ，格納，検索を行なうネイティブ XML データベースが登場してきている．しかし，明確な標準が存在しないため，従来の関係データベースシステムが提供している相互運用性，プログラミングの容易さ，管理の容易さなどが欠落している．また，XML 問合せ言語には，XML データのデータベースへの更新や挿入，削除を表現する方法がない．その結果，従来の関係データベースに XML のための新しい抽象型を定義しているデータベース拡張とネイティブ XML データベースが共存しているのが現状である．

2.2.3 XML 文書検索に対する要求

文書としての XML 文書検索では，利用者の問合せは，データとしての検索に比べ，曖昧であり，検索結果に順位付けを利用することで利用者の検索要求を満たす結果を提供する方法が一般的である．従って，従来の情報検索モデルを基に，XML 文書の特性を加味した XML 文書検索モデルを設定する必要がある．狭義の情報検索の目的は利用者が提示した検索質問に適合する文書をみつけだすことである．利用者の情報要求と文書の内部表現をどちらも索引語の集合として扱う．

情報検索システムにおいて伝統的なモデルと構造化文書検索モデルは，次のように分類される [4].

- 伝統的な IR モデル
  文書を索引語の集合とみなし，問合せと文書の関連性を求めるモデルであ

る．XML 文書検索においても問合せと文書との関連性を必要とする場合は，これらの IR モデルを採用する必要がある．

(1) ブーリアンモデル
文書と問合せが各々索引語集合によって表現される．このモデルは集合論に基づく．問合せを論理式で表現し，検索する方法は，ブール代数を基礎としている．問合せ中の索引語がそのままの形で文書中に出現することを要求する．検索結果に順序付けを行なうことができない．

(2) ベクトルモデル
索引語–文書行列によって文書と問合せが各々t 次元空間のベクトルとして表現される．問合せと文書を同じ索引語の重みベクトルで表し，ベクトル間の類似度によって問合せに対する文書の適合度を計算する．このモデルは，代数的である．

(3) 確率モデル
文書と問合せ表現は確率論に基づいている．ある文書が問合せに対して適合するかどうかは確率的に決定されると仮定している．さらにある文書がある問合せに適合するかどうかは，他の文書がその質問に適合するかどうかとは独立であると仮定する．このモデルは確率的である．

- 構造化文書検索モデル
1980 年代から様々な構造化文書検索モデルが提案されている．しかし，モデルの表現力が高いと問合せ評価の効率が下がるトレード オフがある．次の二つのモデルが代表的なモデルである．

(I) Non–overlapping lists モデル [8, 10, 11]
構造化文書を重なりのないリージョン（文書中の部分文字列の区間）のリストに分割するモデル．各リージョンを独立したデータ構造として扱う．各リージョンを索引の見出しとした転置ファイルを構築する．検索結果はリージョンである．

(II) Proximal nodes モデル [35, 3, 36]
同一の文書の各構造の階層に独立した索引構造を持つモデル．索引構

造は，各階層（例：chapters, sections, paragraphs 等）ごとに作られ，ノードと呼ばれる．このモデル上の問合せ言語は，文字列検索に正規表現を指定したり，構造要素を名前で参照したり，両者の結合で指定できる．

XML 文書検索モデルに，これらの構造化文書検索モデルの適用が考えられる．本研究では，(II)Proximal nodes モデルと同様，文書の論理構造に基づいた部分文書に分割する方式を採用する．論理構造に基づく部分文書は，XML の問合せ言語やノードの位置を与える XPath 式において記述しやすく，部分文書の抽出や再利用が容易であることが理由である．

問合せ

情報検索の各検索モデルに基づき問合せが設定される．宣言的な XML 問合せ言語は，データとしての XML 文書向けであり順位付けの概念が導入されていない，情報検索の問合せでは，問合せ結果が順位付けされることが重要である．

キーワードに基づく問合せ キーワードを与えて，問合せ中の少なくとも一つのキーワードを含む文書集合を問合せとキーワードの類似度によって順位付けしたものが問合せ結果となる．キーワードの与え方によって，(1) キーワード一つの場合 (2) キーワードを並べて句にした場合 (3) 複数のキーワードとその出現文字間隔を与える場合 (4) 複数のキーワードとその論理結合 (OR, AND, BUT) を与える場合 がある．

パターンマッチに基づく問合せ 文書中の部分文字列について問合せをパターンとして与え，そのパターンに合った文字列を含む文書集合を問合せとの類似度によって順位付けしたものが問合せ結果となる．しかし，パターンマッチ式の評価結果を順位付けするのは難題なので，前述のキーワードに基づく問合せの拡張として利用している．パターンとして，(1) 基本の文字列, (2) 接頭辞，(3) 接尾辞，(4) 部分文字列，(5) 辞書順の並びでの文字列の範囲指定 (6) タイプミスや文字認識ミスの許容範囲指定 (7) 正規表現 などがある．

2.2.4 XML 検索エンジンに求められる性質

文書とデータという二面性を持った多数の XML 文書から利用者が必要とする文書やデータを取り出すことが，今後の XML 文書検索には必要となる．XML 問合せ言語は，曖昧さのない問合せを記述するが，文書については，問合せ中に許容される曖昧さの度合によって情報検索の側面を加味していく必要がある．文書検索とデータ検索の違いと両者を合わせ持つ XML 検索エンジンに求められる性質は，次の通りである．

表 2.2 XML 文書検索の性質

	文書検索	データ検索	XML 検索エンジン
情報表現	構造が文脈を表し，内容については，言語や構造などによって無数に存在	データ形式の情報表現はデータ型により有限	文書もデータもタグによる付加情報を持つ
情報要求	利用者が必要としている文書だが，正解は利用者の評価による	利用者が必要としているデータで正確な値が得られるか存在しないかのどちらか	利用者が必要としている文書やデータ
問合せ表現	キーワードやそれらの論理演算	問合せ言語（例 SQL）	XML 問合せ言語あるいはキーワードやそれらの論理演算
情報要求と問合せ表現のギャップ	情報要求の一部を問合せ表現できるだけ	情報要求と問合せ表現が密接に対応	問合せ表現の試行段階で標準がない
問合せ結果の提示法	検索文書についてメタデータ表示，すべて表示，該当数など	問合せ言語実行結果の表やオブジェクト	検索結果のメタ情報表示，要約表示，該当数，該当部分など
問合せ結果	問合せ結果の数が膨大になる可能性，ランキングに基づく足切り 利用者ごとに正解が異なる（曖昧）	常に問合せ結果の数は有限 正確な値で結果に曖昧さはない	問合せによって結果が有限なものからランキングの必要なものまで多様 曖昧さを必要とする場合と排除する場合がある
情報要求と問合せ結果のギャップ	問合せ表現が不適切な場合と情報要求に適合する文書が存在しない場合を含む	別な問合せ表現が限られるので少ない	文書構造を利用することで文書検索よりはギャップが少ない
問合せ結果の検証	利用者の満足度だが測定は困難	結果とパフォーマンスの測定は容易	文書構造を利用するだけ文書検索より満足度が向上

- 情報表現

  文書指向の XML では，要素型が文書構造の文脈情報，属性が付加情報を表現する．文書内容は自然言語で記述されるため，表現方法は無数存在する．

データ指向の XML では，要素型や属性がデータ名やデータ型を表現し，文書内容にはその値が記述されるため，データ型の定義によってその表現方法は有限となる．XML 検索エンジンでは，文書内容が自然言語の場合とデータ値である場合の自動識別によってある程度，各要素型の文書内容を分類できると考えられる．また，XML スキーマは，文書内容を型付けするための枠組も提供しているので，記述されている内容をデータ型で束縛したり，データ型に変換することができる．

- 情報要求
  文書指向の XML に対する問合せにおいて，利用者は，最終的には，必要としている情報が得られるか否かが評価の基準となり，正解は利用者の評価に依存し，あらかじめ設定することが困難である．データ指向の XML において，利用者が問合せに記述するのは，データ名とその値に関係するものであり，データベースへ問合せることによって正解か否かがはっきりわかる．XML 検索エンジンでは，自然言語で書かれた部分については，あいまいさを許容する順位付けを必要とするが，データ名とその値の指定は問合せの条件に合っているか否かの迅速な判定を必要としている．

- 問合せ表現
  文書指向の XML では，問合せは，キーワードやそれらの論理演算で与えることで現在の Web 検索エンジンと同様の問合せ表現となる．データ指向の XML では，データベース内部で利用する問合せ言語で宣言的に記述できる．XML 検索エンジンでは，問合せ言語による問合せ表現に加え，もっと単純なキーワード入力も組み合わせた問合せ表現が必要となる．

- 情報要求と問合せ表現のギャップ
  文書指向の XML では，自然言語の曖昧さを問合せ表現に結び付けることでギャップを埋めることできる．データ指向の XML では，情報要求を問合せ言語に置き換える機会が多いので，ギャップは少ない．XML 検索エンジンでは，ギャップの少ない問合せ表現にむけての試行段階である．

- 問合せ結果の掲示法

文書指向の XML では，検索結果についてのメタデータを表示したり，該当数などの出力方法を選べることが重要になってくる．データ指向の XML では，問合せ言語の中に結果の掲示法が指定されているが，表やオブジェクトの形での表現が重要である．XML 検索エンジンでも検索結果の掲示法には，いろいろなオプション選択が必要となる．少なくとも，検索結果を表示する場合に，従来の情報検索における KWIC 同様，問合せに該当している場所だけではなく，周辺の文脈情報が重要である．

- 問合せ結果
  文書指向の XML では，結果の数が膨大になる場合を考慮して順位付けに基づく足きりが必要である．データ指向の XML では，検索結果の数も有限であり，正確な値が結果となるので，表現に曖昧さはない．XML 検索エンジンでは，曖昧さを認めて順位付けした結果と曖昧さを排除した結果の両方を問合せに応じて適宜用いる必要がある．

- 情報要求と問合せ結果のギャップ
  文書指向の XML では，自然言語の曖昧さによって問合せ表現が不適切だった場合と情報要求に適合する文書が存在しない場合が考えられる．データ指向の XML では，別の問合せ表現が少ないこともあり，ギャップは少ない．XML 検索エンジンでは，文書構造（経路式）を使うことで，ギャップを埋められる可能性がある．さらに対話的な絞り込み検索を行なうためのてがかりとなる情報の提示が必要である．

- 問合せ結果の検証
  文書指向の XML では，問合せ結果の利用者の満足度によって決定されるが，その測定が困難である．データ指向の XML では，問合せ結果の検証やパフォーマンスは，比較的容易に測定できる．XML 検索エンジンについても利用者の満足度をどう測定すればいいのかは困難な問題である．

表 2.2 は，XML 文書検索の持つ性質を一般的な文書検索とデータ検索と今後必要となる両者の検索が可能な XML 検索エンジンへの要求をまとめたものである．

XML 検索エンジンでは，従来の HTML 文書を対象とした検索エンジンに比べ，XML 文書内に記述されている文書構造を利用することができる．この利点を使って，文書検索の問題点である情報要求と問合せ表現のギャップと情報要求と問合せ結果のギャップを埋める手法が必要である．

2.3 本研究の部分文書検索

本研究において，論理構造上の単位となる文書部分の特定のための基本方針は，論理木構造を保持した部分文書を文書単位とする．すなわち，各部分文書は，必ず一つの最上位ノードとなる要素型を持つ XML 文書とする．従って，部分文書をその最上位要素ノードの番号 n を用いて指定することができる．

本研究で対象とする部分文書は，XML 文書の木構造内の各ノードを根とする部分木に対応する XML 文書断片である．従って，XML 部分文書を次のように定義する．

定義 1 (XML 部分文書) d を XML 文書, $T(d)$ を d に対応した木とする．$T(d)$ 中の根ノード（文書ノード）と葉ノードを除外した中間ノードを $n_1, n_2, \ldots, n_{nd}$ とし，$Node_{T(d)} = \{n_1, n_2, \ldots, n_{nd}\}$ を $T(d)$ 中の中間ノード集合とする． $n_i \in Node_{T(d)}$ に対応した部分文書 $pd(d)_{n_i}$ は，n_i を根ノードとする部分木に対応した d 中の，XML 文書断片のことである．

d の部分文書集合は，

$$PD(d) = \{\ pd(d)_{n_i} \quad | \quad n_i \in Node_{T(d)}\ \}$$

と表すことができる．

この定義によって，本研究における部分文書数は，XML 文書 d に対応した木 $T(d)$ の中間ノード数と一致する．

定義 2 (XML 構造表現) d を XML 文書, $T(d)$ を d に対応した木とし，$T(d)$ 中の中間ノード集合を $Node_{T(d)}$ とする．d を識別する URI を $URI(d)$, $T(d)$ 中の根ノード（文書ノード）から中間 ノード $n_i \in Node_{T(d)}$ までの経路式表現を $Path(d)_{n_i}$

とすると，部分文書 $pd(d)_{n_i} \in PD(d)$ の構造表現を，$(URI(d), Path(d)_{n_i})$ で表す．ただし，経路式表現は，ノード集合を表現するように設計されているため，特定のノードを表現するためには，文書の根ノードから中間ノードまでのノード名とさらに兄弟ノードにおける同名のノード中における出現順位を表すインデックスによって文書中の他のノードと識別可能な方法で表すものとする．

d 中の部分文書は，この XML 構造表現によって他の部分文書と識別される．また，この XML 構造表現を利用することで，XSLT や XQuery などの変換言語や問合せ言語中でこの部分文書にアクセスすることができる．

例 1 図 2.2, 図 2.4は本の XML 文書インスタンス ($d =$ "$chapter.xml$") とその木構造表現 $T(d)$ である．図 2.4 から中間ノード集合は，$Node_{T(d)} = \{\#1, \#2, \ldots, \#10\}$ であり，部分文書 $pd(d)_{\#1}, pd(d)_{\#2}, \ldots, pd(d)_{\#10}$ は，中間ノードを 根ノードとする部分木に対応した 10 個の XML 文書断片である．これらの部分文書は，URI と経路式表現から構成される構造表現によって識別される．表 2.3がその例である．

表 2.3 XML 部分文書の構造表現 ($d =$ "$chapter.xml$")

ノード番号	部分文書	構造表現
#1	$pd(d)_{\#1}$	(chapter.xml, `/chapter`)
#2	$pd(d)_{\#2}$	(chapter.xml, `/chapter/title`)
#3	$pd(d)_{\#3}$	(chapter.xml, `/chapter/section[1]`)
#4	$pd(d)_{\#4}$	(chapter.xml, `/chapter/section[1]/title`)
#5	$pd(d)_{\#5}$	(chapter.xml, `/chapter/section[2]`)
#6	$pd(d)_{\#6}$	(chapter.xml, `/chapter/section[2]/title`)
#7	$pd(d)_{\#7}$	(chapter.xml, `/chapter/section[2]/section[1]`)
#8	$pd(d)_{\#8}$	(chapter.xml, `/chapter/section[2]/section[1]/title`)
#9	$pd(d)_{\#9}$	(chapter.xml, `/chapter/section[2]/section[2]`)
#10	$pd(d)_{\#10}$	(chapter.xml, `/chapter/section[2]/section[2]/title`)

本研究では，ここで定義した部分文書集合の中から次に定義する問合せモデルの意味での問合せ結果として問合せを満たす部分文書を取り出す．本研究における部分文書検索の特性は，一つの XML 文書について中間ノード数だけの部分文

書が存在し，問合せ中の条件を満たすか否かは，各部分文書を対象として評価されなければならないことである．

定義 3 (問合せモデル) $\mathbf{D}$ を XML 文書集合，$\mathbf{S}$ を $\mathbf{D}$ 中の XML 文書 d の各部分文書集合の構造名（要素型名，属性名）の和集合，$\mathbf{T}$ をキーワード 集合とする．

部分文書検索の問合せ q は，次の述語 **contains** の有限個の組の順序なしの集合として定義する：

(1) 要素名あるいは属性名として指定する構造名 $s_k \in \mathbf{S}$ にキーワード $t_k \in \mathbf{T}$ を含むという述語 **contains**：

$$q_{(t,s)} = \mathbf{contains}(t, s) \quad t \in \mathbf{T}, s \in \mathbf{S}$$

(2) 構造指定がない場合（すべての構造について）のキーワード $t_k \in \mathbf{T}$ を含むという述語 **contains**:

$$q_{(t,*)} = \mathbf{contains}(t, *) = \bigcup_{\forall s_k \in \mathbf{S}} \{\mathbf{contains}(t, s_k)\} \quad t \in \mathbf{T}$$

のように表し，

(3) キーワード 指定がない場合（すべてのキーワード について）の構造名 $s_k \in \mathbf{S}$ を含むという述語 **contains**：

$$q_{(*,s)} = \mathbf{contains}(*, s) = \bigcup_{\forall t_k \in \mathbf{T}} \{\mathbf{contains}(t_k, s)\} \quad s \in \mathbf{S}$$

この問合せ中の各述語を部分文書中で評価する．ただし，各述語間の関係を，AND，OR と解釈するか，あるいは，重みつきの出現確率と解釈するかは，採用する情報検索モデルに依存する．

例 2 `title` に “XML” を含む部分文書の問合せはこのモデルでは次のように表現する：

$$q = \{\mathbf{contains}(\ \texttt{`XML'},\ \texttt{title})\}$$

本研究では，問合せの結果として要素名`title`の内容に文字列 'XML' を含み，文書構造上の文脈の単位となっている部分文書を求める．

次に本研究における XML 部分文書検索を定義する．

定義 4 (XML 部分文書検索) 本研究の部分文書検索モデル M は次の四つ組 $(\mathbf{D}, \mathbf{Q}, \mathbf{PD}, R)$ で表す．

(1) $\mathbf{D}$: XML 文書集合

(2) $\mathbf{Q}$: 利用者が必要としている定義 3 で定義した問合せモデルによる問合せ集合

(3) $\mathbf{PD}$: $\mathbf{D}$ 中の XML 文書の部分文書集合の和集合 すなわち，

$$\mathbf{PD} = \bigcup_{\forall d \in \mathbf{D}} PD(d)$$

(4) R : 問合せ $q_i \in \mathbf{Q}$ と 部分文書 $pd_j \in \mathbf{PD}$ の関連性を表した実数 $\in \mathbf{R}$ を求める関数．

$$R : \mathbf{Q} \times \mathbf{PD} \longrightarrow \mathbf{R}$$

関数 R の値は，論理値を利用する場合は $\{0,1\}$ とし，類似度を計算する場合はその値となる．この関数は，情報検索のモデルに依存する．

本研究では，XML 文書集合から利用者の問合せを満たす部分文書検索として問合せ条件を満たし，文書構造上の文脈の単位となっている適切な部分文書を取り出すアルゴリズムを提案する．検索結果の XML 部分文書は，定義 2 によって，XML 構造表現によって他の部分文書と識別できる．従って，本研究における部分文書検索は，検索結果としての XML 文書断片とこの XML 構造表現を求めることである．

本章では，本研究が対象としている部分文書について定義した．

本論文では，利用者の入力する問合せの前提として（1）XML 文書に利用されている共通モジュールについての知識を持っていて，構造とキーワード の組として問合せを与える場合,（2）XML 文書の構造についての知識を全く持っていない

ため，キーワード集合として問合せを与える場合に分けて，部分文書検索手法を提案する．

3 章では，構造とキーワードの組として問合せを与える場合の部分文書検索について，1) 部分文書全体の中から問合せの回答候補の部分文書を取り出すこと．2) 回答候補の部分文書のうち，問合せを満たす構造上の最小文脈となっている部分文書を選ぶアルゴリズムを提案する．

4 章では，キーワード集合として問合せを与える場合の部分文書検索について，1) 元の XML 文書中の検索対象となる索引語について，個々にその索引語を含む構造上の最小文脈となっている部分文書をキーワード入力に関する回答候補として選ぶアルゴリズムについて提案し，2) 実際の問合せキーワードから回答候補の部分文書集合の中から問合せキーワードとの関連性の高い部分文書を探す手法を提案する．

第3章

構造とキーワード 指定による共通のモジュールを持つXML文書集合における部分文書検索

XML 文書の問合せ言語や問合せ処理に関する研究は多数行なわれているが，我々の知る限り，標準語彙や名前空間 に関する問合せを取り扱っているものはない．本章では，まず現状の XML 問合せ言語におけるキーワード 検索機能と標準語彙，名前空間について述べる．

次に，利用者に検索対象 XML 文書集合についての論理構造についての完全な知識は持たないが，標準語彙中の要素や属性名についての知識を持つ場合の部分文書検索について述べる．本章の部分文書検索の目的は次の通りである：

1. 利用者が文書の論理構造についての完全な知識を持たないが標準語彙中の要素や属性を名前空間つきで指定する場合を想定した問合せとすること．
2. 利用者の入力キーワード に関連した適切な粒度の部分文書を抽出すること．
3. 入力キーワード の出現する構造上と内容に関する文脈から，利用者が問合せ内容をより明確にすることが可能なこと．

まず 3.1 節では，XML 問合せ言語によるキーワード 検索について述べ，3.2 節では，標準語彙と名前空間の意義について述べる．3.3 節では，XML 名前空間と XML 名前空間を利用している XML 文書の例について述べ，3.4 節では，共通のモジュールを持つ XML 文書集合に対する部分文書検索と検索モデルについて，

3.5節では，単純問合せの場合の部分文書検索について，3.5節では，単純問合せモデルにおける検索結果候補の部分文書を同定方法について述べる．3.6節では，AND問合せについて，3.7節では，OR問合せについて述べ，3.8節で実験データによる部分文書検索とその結果の考察を行ない，3.9節は，本章のまとめである．

3.1 XML問合せ言語によるキーワード検索

構造化文書データベースと情報検索システムの統合利用に関する研究は，これまでにもなされてきているが，構造化文書として主にSGML[25, 26, 27]文書を対象としてきたため，各文書の構造は，文書型定義(DTD)によってあらかじめ定義されているという前提のもとに論議されてきた[33, 23, 36, 46, 19]．しかし，XML文書では，妥当なXML文書だけではなく，整形式の文書も多く，より柔軟に利用できるため，XML文書に対応した検索方法の研究が必要である．

現在の構造化文書検索は，選択条件および出力文書構造をXML問合せ言語を用いて宣言的に指定する方法とWebサーチエンジンにみられる情報検索技術によるキーワード入力による全文検索に分類される．

XML問合せ言語はこれまでに多数提案されている[6]. 本節では，W3Cが提案しているXQuery[58]を利用して実行可能なキーワード検索について述べる．

XQueryでは，ビルトイン関数のコアライブラリを提供している[57]．これらの関数を利用して文字列操作を行なうことができる．次に示すのが文字列関数とノード操作関数である．

(1) 文字列関数

`xf:concat`, `xf:starts-with`, `xf:end-with`, `xf:contains`, `xf:substring`, `xf:string-length`, `xf:substring-before`, `xf:substring-after`, `xf:normalize-space`, `xf:normalize-unicode`, `xf:upper-case`, `xf:lower-case`, `xf:translate`, `xf:string-pad`, `xf:match`, `xf:replace`

(2) ノード操作関数

`xf:local-name`（ローカル名を求める），`xf:number`（ノードの内容を数

値化する），op:node-equal（二つのノードが同一かどうかを識別する），xf:deep-equal（二つの属性ノードが同一かどうかを識別する），op:node-before（文書順で前のノード），op:node-after（文書順で後のノード），xf:copy（ノード以下の部分木に対応するすべてのノードをコピーする），xf:shallow（ノードとその属性だけをコピーする）

次に示すのが，文字列関数を利用したXQueryでの問合せ例である．

同一の文に “sailing” と “windsurfing” を述べている本のタイトルをみつける．

```
FOR $b IN //book
WHERE SOME $p IN $b//para SATISFIES
 (contains($p, ‘‘sailing’’) AND contains($p, ‘‘windsurfing’’))
RETURN $b/title
```

以上のように，現在提案されているXQueryでは，キーワード検索において，特定のノード（集合）とキーワードを指定した条件を満たすか否かを述語とする．現状では，文字列操作はできるが，正規表現での指定はできない．

3.2 標準語彙と名前空間の意義

XMLでは独自のマーク付け言語（語彙）を自由に設計できるが，多くの人が利用する語彙は，標準的な語彙を共有する方が効率が良く，相互運用性も高まる．そこで，ひとつのXML文書を作成するのに，さまざまなマーク付け言語の語彙を利用し，組み合わせて利用できるような枠組が必要となる．複数の語彙をモジュールとして組み合わせるためには，それぞれの語彙をきちんと区別できなければならない．例えば，titleという名前を持つ要素が，ある時は書物の「タイトル」を表したり，別の時は作者の「肩書き」を表したりすることがある．

この問題を解決するためには，それぞれの語彙が世界で一意に定まる名前を持つための方法が必要となる．この実現方法が，1999年にW3Cから勧告されたXML 名前空間 [53] である．XML名前空間は，語彙（要素型名，属性名）をURI[52] と組み合わせる（修飾する）ことで，複数の語彙を混在させる機構であ

表 3.1 応用 XML マーク付け言語とその名前空間 URI

応用 XML マーク付け言語	名前空間として指定する URI
SVG (Scalable Vector Graphics)	“http://www.w3.org/2000/svg/”
MathML (Mathmatical Markup Language)	“http://www.w3.org/1998/Math/MathML”
SMIL (Synchronized Multimedia Integration Language)	“http://www.w3.org/2001/SMIL20”
RDF (Resource Description Framework)	“http://www.w3.org/1999/02/22-rdf-syntax-ns♯”
XHTML (eXtensible Hyper Text Markup Language)	“http://www.w3.org/1999/xhtml”

る．また，XML 名前空間は，タグセットを組み合わせて利用できるようにするだけでなく，Web 上で共有・交換される情報の意味を正確に示し，計算機が理解するための役割も果たす．URI は世界で重複することのない識別子なので，語彙ごとに URI を割り当てれば，ある URI で修飾された要素型名や属性名は，必ず区別できる．XML 名前空間の宣言は，xmlns 属性を使って行ない，名前空間の接頭辞と URI の対応は，宣言を行った要素およびその子孫要素にわたって有効になる．

W3C では多くの応用 XML マーク付け言語, SVG(Scalable Vector Graphics), MathML(Mathmatical Markup Language), SMIL(Synchronized Multimedia Integration Language), RDF(Resource Description Framework), XHTML(eXtensible Hyper Text Markup Language) の仕様を制定している．表 3.1 は，応用 XML マーク付言語とその名前空間指定で利用する URI の例である．

名前空間は業界標準の要素集合を識別するためにも使われている．CML(Chemical Markup Language), DocBook, Dublin Core Element Set, NewsML, RELAX Core などが公開されている．例えば Dublin Core[20] では 15 の要素集合を定義している．我々はこの 15 の要素型を XML 文書作成において Dublin Core の意味において利用することができる．従って, 文書の一部の構造に外部の名前空間で定義されている要素型や属性を利用することで，文書全体としてはスキーマのない整形式でありながら文書中に共通性のある XML 文書が多数出現している．本章では図 3.1 のように, 文書の一部に共通の名前空間の要素を持っている XML 文書を

対象とする．我々は, 従来の構造化文書で必要であったスキーマ設計作業は難しく, XML の利用者にとってスキーマ設計を行なう代わりに名前空間という枠組を利用するほうが容易であることから, このような XML 文書は急増すると考える．実際，業界ごとに認知された標準語彙は急増している．

このような XML 文書に対しては，利用者が XML 文書の完全な構造についての知識を持っていない場合でも，標準語彙についての知識を持っていれば，キーワードと標準語彙中の要素や属性名を組み合わせて問合せを行うことができる．

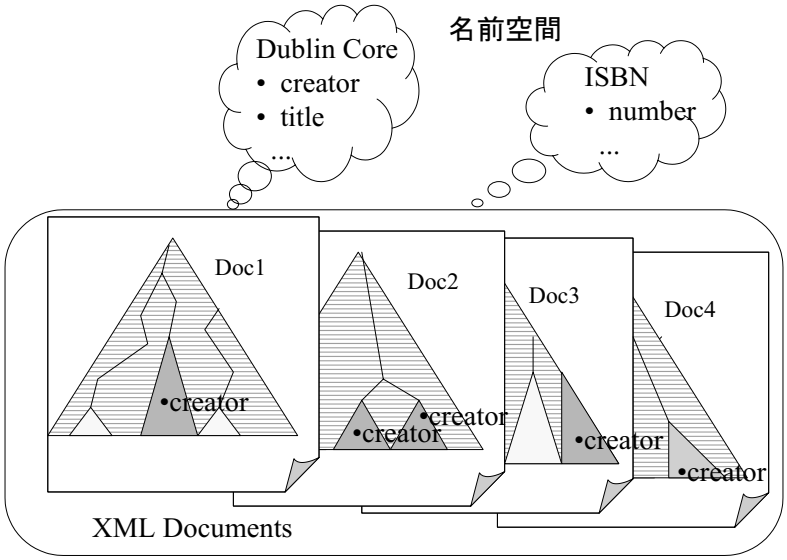

図 3.1 共通の名前空間を利用している XML 文書

3.3 語彙の識別のための XML 名前空間

計算機分野において,「名前空間」という用語は伝統的に名前の集合，すなわち重複を含まない数学的な集合体を指す．しかし，XML 文書中で使われる要素型などの名前をそうした名前空間として扱うことは，その有用性を著しく損なうた

```
<?xml version="1.0" ?>
<stream>
  <head>
   <rdf:RDF
       xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:dc = "http://purl.org/dc/e/elements/1.1/" >
     <rdf:Description about="meta.smi">
       <dc:title>SMIL Database</dc:title>
       <dc:format>text/smil</dc:format>
       <dc:creator>
          <rdf:Seq ID="name">
             <rdf:li>Kinutani</rdf:li>
             <rdf:li>Hatano</rdf:li>
          </rdf:Seq>
       </dc:creator>
     </rdf:Description>
     <rdf:Description about="meta-1999.mpg">
       <dc:title>Video Database</dc:title>
       <dc:creator>Smith</dc:creator>
       <dc:format>video/mpg</dc:format>
       <dc:language>en</dc:language>
     </rdf:Description>
     <rdf:Description about="#scene1">
       <dc:title>XML Database</dc:title>
     </rdf:Description>
   </rdf:RDF>
  </head>
 <scene id="#sene1">
      <title>Network Session</title>
      <location>Room A</location> </scene>
 <scene>
      <title>Database Session</title>
      <chair>Uemura</chair></scene>
</stream>
```

図 3.2 XML名前空間を利用しているXML文書例：stream.xml

め，XML名前空間は，文書中の構造を表す名前がXML文書の範囲を超えて識別できるような機構である．XML文書中でのそうした名前の主な使用目的は，ソフトウェアモジュールによる文書の論理的構造の識別を可能にすることにある．

XML名前空間は, URI参照によって識別される．このURI参照によってXMLの語彙としてはXMLの語彙空間全体で唯一となり，他の要素型や属性と識別できる．XML名前空間を指定した要素型や属性の名前は, 名前空間を示す接頭語とコロンとローカル部分を含むことができる．この接頭語は, 名前区間を選択するためのURI参照に写像される．たとえば, `a:title`では, 名前空間の接頭語は `a`

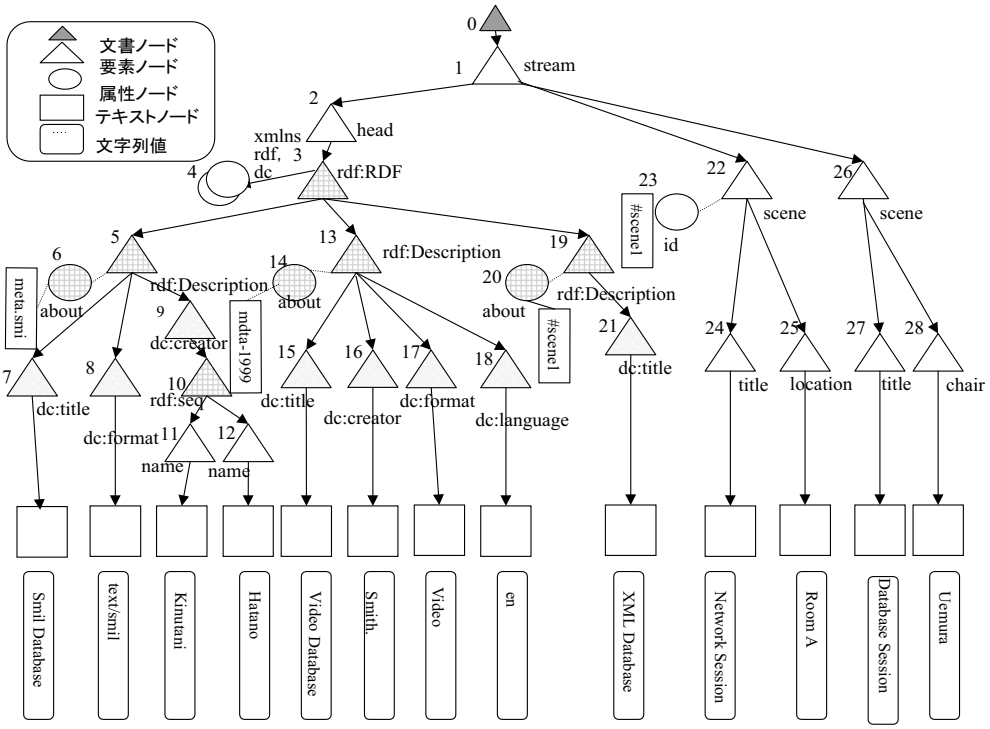

図 3.3 XML 名前空間を利用している XML 文書例 (stream.xml) の木構造表現

で, このaが URI 参照への写像を宣言している. このローカル部分 titleが aで限定され, 文書が解析されるときには, titleがその URI で限定される.

名前空間は, 予約属性xmlns か xmlns:を接頭語として XML 文書中で宣言される. 例えば, <x xmlns:a='http://www.foo.com/foo.xsd'>は, 要素型 x とこの要素型の子孫として宣言されている要素型や属性すべてが'http://www.foo.com/foo.xsd'の名前空間に属するものと指定し, aで限定される.

この名前空間 URI は, 名前空間を識別するためだけに使用される. そのため, 名前空間 URI が指定する先に何かの意味のある情報が置かれているわけではない. 従って, 誰でもが自分の固有の URI を利用して独自の語彙を作成し, 利用することができる.

図 3.2 が二つの XML 名前空間を利用している XML 文書例である. dc と rdf

がXML名前空間の接頭語である．前者の接頭語 dc では，Dublin Coreのメタデータの意味で要素型や属性を指定している．後者の接頭語 rdf では，XMLのリソース記述に関する仕様RDF[54]の意味で要素型や属性を指定している．図 3.3が図 3.2のXML文書の木構造表現である．各中間ノードは，要素型名，属性名のラベルを持つ．葉ノードはテキストノードで文字列値を持つ．各中間ノードには，参照のため#1から #28の番号をつけている．

3.4 共通のモジュールを持つXML文書集合に対する部分文書検索

標準語彙をXML名前空間を利用して採り入れたXML文書は，一般的には整形式なので，文書全体の構造についての構造定義は，存在しない．しかし，同じ名前空間に属する要素型や属性を利用しているXML文書集合には，共通の要素型や属性が標準語彙としてのコンセンサスの得られた意味を持っている．ただし，各文書ごとにその使われ方や構造上の位置も一通りではない．そのため，共通な意味を持つモジュールとして利用されている名前空間とその要素型や属性を指定してその要素型や属性の意味的範囲と構造上の役割を求めることで，利用者のあいまいな構造条件での問合せ処理を補完することができると考えられる．

利用者や応用プログラムにとって，これらのXML文書集合への問合せに際して，取り出したい情報のある場所についての完全な経路を前もって知ることは困難である．できるのは，これらの名前空間に属している要素型名や属性を指定することだけである．しかし，これらの要素型や属性の値を求めることでは，これらの値の前後にある情報を取り出すことができない．情報検索において入力キーワードとキーワード周辺の文を検索結果として表示するKWIC(KeyWord In Context)（[4, p.290）が，利用者にとってキーワードが使われている背景の文脈を知る手がかりとして有用だったように，XML文書全体が属するデフォルトの名前空間の要素型や属性と問合せに利用した名前空間の要素型や属性の間の関係を知りたいなら，指定した名前空間に関連した文脈の範囲を抽出する必要がある．

例えば，Dublin Coreの名前空間に属するタイトルと著者を表す要素型dc:title

と dc:creator についてタイトルに “XML” が含まれ，著者に “Bray” が含まれている論文や，本や，雑誌について知りたい場合，XML 文書中の該当場所の文脈情報として論文であるか本であるか雑誌であるかがわかる記述が検索結果に含まれていれば，利用者に有用な結果となる．

3.4.1 部分文書検索モデル

本章の部分文書検索モデル $M1$ を定義 4 のモデル M に従い，XML 文書集合 $\mathbf{D}$ が 共通のモジュールを持つ XML 文書集合である場合について考察する．

2.3 節の定義 4 より，本研究の部分文書検索モデル $M1$ を次の四つ組 $(\mathbf{D}, \mathbf{Q}, \mathbf{PD}, R)$ で表す．

(1) $\mathbf{D}$: 共通のモジュールを持つ XML 文書集合
この XML 文書の論理表現における特徴は，文書の論理構造の一部に同一の名前空間指定の要素型や属性を利用していることから，各文書中の一部に共通の構造があることである．

(2) $\mathbf{Q}$: 利用者が必要としている 2.3 節の定義 3 で定義した問合せモデルによる問合せ集合，$\mathbf{D}$ を XML 文書集合，$\mathbf{S}$ を $\mathbf{D}$ 中の XML 文書 d の各部分文書集合の構造表現の和集合，$\mathbf{T}$ を $\mathbf{D}$ 中のキーワード集合とすると，本章の問合せモデルは、次のように表すことができる：部分文書検索の問合せ q は，次の述語 **contains** の有限個の組の順序なしの集合として定義する：

(1) 要素名あるいは属性名として指定する構造名 $s_k \in \mathbf{S}$ にキーワード $t_k \in \mathbf{T}$ を含むという述語 **contains**：

$$q_{(t,s)} = \mathbf{contains}(t, s) \quad t \in \mathbf{T}, s \in \mathbf{S}$$

(2) 構造指定がない場合（すべての構造について）のキーワード $t_k \in \mathbf{T}$ を含むという述語 **contains** は本章の前提を満たさないので除外する．

(3) キーワード 指定がない場合（すべてのキーワード について）の構造名

$s_k \in \mathbf{S}$ を含むという述語 **contains**：

$$q_{(*,s)} = \mathbf{contains}(*, s) = \bigcup_{\forall t_k \in \mathbf{T}} \{\mathbf{contains}(t_k, s)\} \quad s \in \mathbf{S}$$

この問合せ中の各述語を部分文書中で評価する．ただし，各述語間の関係を，AND，OR と解釈するか，あるいは，重みつきの出現確率と解釈するかは，採用する情報検索モデルに依存する．

応用マーク付け言語や標準語彙は，それらの目的のため，独自の名前空間と要素型，名前を定義している．従って，問合せ中に指定できる構造表現は，名前空間つき要素型名と属性名である．

(3) $\mathbf{PD}$: $\mathbf{D}$ 中の XML 文書の部分文書集合の和集合 すなわち，XML 文書中のすべての中間ノードに対応した部分文書全体である．

$$\mathbf{PD} = \bigcup_{\forall d \in \mathbf{D}} PD(d)$$

(4) R : 問合せ $q_i \in \mathbf{Q}$ と 部分文書 $pd_j \in \mathbf{PD}$ の関連性を表した実数 $\in \mathbf{R}$ を求める関数.

$$R : \mathbf{Q} \times \mathbf{PD} \longrightarrow \mathbf{R}$$

関数 R の値は，論理値を利用する場合は {0,1} とし，類似度を計算する場合はその値となる．本章の対象とする XML 文書では，取り出した部分文書に対しての問合せとの関連性をどう数値化し，順位付けするかは構造と内容に関する類似度をどう定義するかに依存する．本研究では，この類似度については，研究対象としない．

3.5 単純問合せ

本節では，3.4.1 節で述べた問合せにおいて，キーワードと構造指定の組が一組の場合の部分文書検索について述べる．

単純問合せを，次のように表す.

問合せ $q \in \mathbf{Q}$ は，構造表現（要素型，属性）$s \in \mathbf{S}$ とキーワード $t \in \mathbf{T}$ との組み合わせとして：
(1) $q = \{\mathbf{contains}(t, s)\}$ $(t \in \mathbf{T}, s \in \mathbf{S})$ あるいは，
(2) $q = \{\mathbf{contains}(*, s)\}$ $(s \in \mathbf{S})$
で表す．

例 3 次の式は，図 3.2 の XML 文書に対する単純問合せの例である[1] ．

$$\{\mathbf{contains}(\text{`XML'}, \texttt{dc:title})\} \quad (Q1)$$
$$\{\mathbf{contains}(*, \texttt{dc:creator})\} \quad (Q2)$$

本研究の部分文書検索における問合せ (Q1) の意味は,「構造表現dc:title 中にキーワード 'XML' が含まれる部分文書のうち，構造上の文脈の単位となる部分文書を検索結果とせよ」ということである．この問合せによって，利用者は，Dublin Core の意味でのタイトルの一部にキーワード 'XML' が含まれている XML 文書の一部とこのタイトルという要素が XML 文書の論理構造上，どんな役割を持った場所に出現するかを知ることができる．例えば，該当箇所がこの XML 文書のメタデータを表す場所にあるとか，あるいは，会議録中の多数の論文の一タイトルであるのかを知ることによって，検索結果の絞り込みや再検索を支援することができる．

問合せ (Q2) の意味は,「構造表現 dc:creatorが含まれる部分文書で，dc:creator が一定の文脈を持つような部分文書を検索結果とせよ」ということである．この問合せによって，利用者は，Dublin Core の意味でのクリエータの内容だけでなく，(Q1) と同様，この要素が XML 文書の論理構造上負っている役割を知ることができる．

3.5.1 単純問合せにおける検索結果候補の部分文書の特定

本節では，共通のモジュールを持つ XML 文書集合 $\mathbf{D}$ に対して，単純問合せ q において問合せの述語を満たすノードを含む部分木に対応する部分文書を特定す

[1] dc は名前空間の接頭辞だが，記述の簡単化のためここでは，名前空間を URI : http://purl.org/dc/elements/1.0/で展開しない

る方法について述べる．従来の文書単位の検索では，元文書が問合せ q を満たすか否かのいずれかであった．部分文書検索では，部分文書を単位として問合せ q を満たすか否かを判定する必要がある．本研究では，部分文書間に入れ子関係があるため，問合せを満たすノードと文書ノードの間のすべての中間ノードに対応する部分文書では，問合せ q を満たすことになる．従って，これらの部分文書が検索結果候補の部分文書となる．

まず，問合せ中に指定された構造を持つ対象文書集合中で問合せを満たす各ノードをカレントノードと定義し，カレントノードの文書中の位置から，検索結果の候補となりうる部分文書を求める．

定義 5 (単純問合せにおけるカレントノード :$curr(q)$) XML 文書 d に対する単純問合せにおけるカレントノードを次のように定義する. 問合せ $q \in \mathbf{Q}$ のカレントノード集合を $curr(q)$ と記述する．：

(1) 構造表現 $s \in \mathbf{S}$, キーワード $t \in \mathbf{T}$ を指定した単純問合せ $q = \{\mathbf{contains}(t, s)\}$ におけるカレントノード 集合 *curr(q)* は，次の XPath 式を満たすノード集合である：

curr(q) = `document(d)//s[contains(.//text(),t)]`
OR `document(d)//s[contains(attribute::*,t)]`

(2) 構造表現 $s \in \mathbf{S}$ を指定した単純問合せ $q = \{\mathbf{contains}(*, s)\}$ におけるカレントノード集合 *curr(q)* は，次の XPath 式を満たすノード集合である：

curr(q) = `document(d)//s`

定義 6 (単純問合せを満たす候補ノード : $C(q)$) 単純問合せを満たす候補ノードは，XML 文書 d とその木構造を $T(d)$ とすると，定義 5 で定義したカレントノードと文書ノードの間の経路上に存在するすべてのノードである．これらのノード集合を $C(q)$ と表す.

この $C(q)$ 中のノードに対応した部分文書は，単純問合せに対する検索結果の部分文書候補である．この単純問合せを満たすノード集合 $C(q)$ は，次の XPath 式を満たすノード集合である：

(1) 単純問合せ $q = \{\mathbf{contains}(t, s)\}$ $(t \in \mathbf{T}, s \in \mathbf{S})$ の場合：

C(q) = `document(d)//*[.//s[contains(.//text(),t)]]`

OR `document(d)//*[.//s[contains(attribute::*,t)]]`

(2) 単純問合せ $q = \{\mathbf{contains}(*, s)\}$ $(s \in \mathbf{S})$ の場合：

C(q) = `document(d)//*[.//s]`

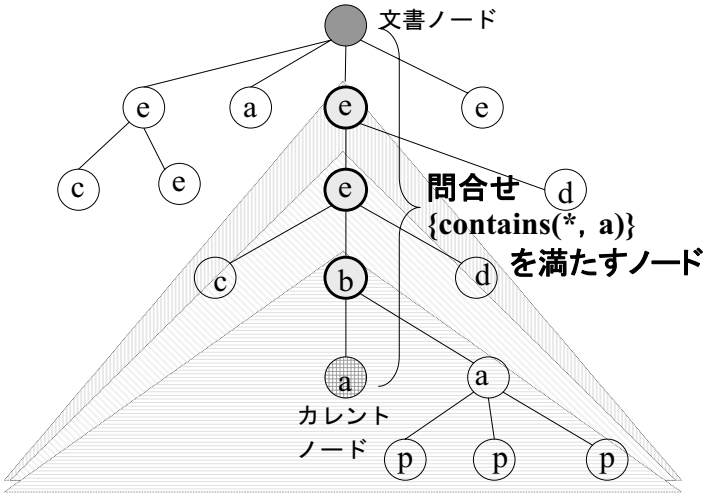

図 3.4 問合せ $q = \{\mathbf{contains}(*, a)\}$ に対する $curr(q)$ と $C(q)$ の関係

図 3.4 は，問合せ $q = \{\mathbf{contains}(*, a)\}$ に対するカレントノード $curr(q)$ と述語を満たすノード $C(q)$ の関係を表す.

```
document(stream.xml)//*[.//dc:title[contains(.//text(),'XML')]]
  OR
document(stream.xml)//*[.//dc:title[contains(attribute::*,'XML')]]
```

図 3.5 (Q1) の XPath による候補ノード C(Q1) 抽出例

```
<results>
{FOR $a IN distinct(document("stream.xml")//dc:title
              [contains(.//text(),‘XML’) OR contains(.//*/@,‘‘XML’)])
  RETURN  <result>
  {FOR $b IN distinct(document("stream.xml")//*[//$a])
     RETURN $b }
       </result>}
</results>
```

図 3.6 (Q1) の XQuery による候補ノード C(Q1) 抽出例

```
document(stream.xml)//*[.//dc:creator]
```

図 3.7 (Q2) の XPath による候補ノード C(Q2) 抽出例

例 4 図 3.2 の XML 文書を管理する場合を考える．問合せを行なう利用者や応用プログラムが，Dublin Core の名前空間 (“http://purl.org/dc/elements/1.1”) を知っていて，そこでは，要素として`title`や `creator`があることが前提である．

図 3.5 は，(Q1) の候補ノード C(Q1) の XPath による抽出例である．XML 文書の根ノードから任意の経路上で，Dublin Core で定義されている 要素ノード名 `title`をラベルに持つノードを子孫ノードに持ち，しかも下部構造中の文字列が “XML” を含んでいる要素ノードを抽出する XPath 表現である．また，図 3.6 は，図 3.5 と同様，(Q1) の候補ノード抽出を XQuery の構文で書いたものである．この問合せは，`dc:title` のタグで囲まれている文字列中に “XML” を含み，この `dc:title` タグを子孫ノードとして含むノード集合を求める式である．これらの式で求められるノードに対応した部分文書は，すべて単純問合せの述語を満たしている．従って，これらの式で取り出されたノードに対応する部分文書が検索結果候補の部分文書となる．

取り出されるノードは，図 3.3 からノード番号#1, #2, #3, #19, #21 となり，各ノードに対応した XPath 表現は，表 3.2 となる．

例 5 図 3.7 は，(Q2) の XPath による検索結果候補抽出例である．XML 文書の根

表 3.2 単純問合せを満たす候補ノード: $C(Q1)$

述語を満たすノード	該当ノード までの経路式
#1	document(‘stream.xml’)/stream
#2	document(‘stream.xml’)/stream/head
#3	document(‘stream.xml’)/stream/head/rdf:RDF
#19	document(‘stream.xml’)/stream/head/rdf:RDF/rdf:Description[3]
#21	document(‘stream.xml’)/stream/head/rdf:RDF/ref:Description[3]/dc:title

ノードから任意の経路上で，Dublin Core で定義されている 要素ノード名`title`をラベルに持つノードを子孫ノードに持つ要素ノードを取り出す XPath 表現である. この式で取り出されたノードに対応する部分文書が検索結果候補の部分文書となる．取り出されるノードは，図 3.3 からノード 番号#1, #2, #3, #5, #9, #13, #16 となり，各ノードに対応した XPath 表現は，表 3.3 となる．

表 3.3 単純問合せを満たす候補ノード $C(Q2)$

述語を満たすノード	該当ノード までの経路式
#1	document(‘stream.xml’)/stream
#2	document(‘stream.xml’)/stream/head
#3	document(‘stream.xml’)/stream/head/rdf:RDF
#5	document(‘stream.xml’)/stream/head/rdf:RDF/rdf:Description[1]
#9	document(‘stream.xml’)/stream/head/rdf:RDF/ref:Description[1]/dc:creator
#13	document(‘stream.xml’)/stream/head/rdf:RDF/ref:Description[2]
#16	document(‘stream.xml’)/stream/head/rdf:RDF/ref:Description[2]/dc:creator

3.5.2 単純問合せにおける文書構造による最適な検索結果となる部分文書

次に，単純問合せに対して文書構造による最適な部分文書を求めるために本研究で提案するアルゴリズムについて述べる．これは，3.4.1 節で述べた部分文書検索モデルの (5)R:のランキングアルゴリズムとして与えることができる．問合せ $q \in \mathbf{Q}$ と検索結果の候補部分文書 $pd_{\in}\mathbf{PD}$ から実数を求める関数は，最適な部分文書に 1 を，それ以外の部分文書に 1 未満の値を与えるものと考えられる．

ここで仮定している単純問合せは，構造とキーワード指定の問合せであり，XML文書集合中にある共通モジュールを手がかりとしてそのモジュールの文書中での意味的範囲と構造上の役割を理解することが目的である．従って，単純問合せ中に指定した構造表現，例えば`dc:title`が，検索結果の部分文書中での出現が1回であるか，あるいはなるべく少ない方が問合せ結果として望ましいと考える．しかし，問合せで指定した構造だけの部分文書では，何も利用者に情報を与えない．従って，問合せ中に指定した構造表現の出現は少なく，他の構造についての出現は多くなるような部分文書を求めたい．DTDを設計する場合は，章，節，段落，文が一般的な文書構造として用いられ，章や節を繰り返す構造とすることから，ヒューリスティックスとして同名の要素型の出現が構造上の切れ目となることを利用する．これは，DTDでの要素型の内容モデル中で繰り返しを指定する+ や* に相当する部分を文書を走査して求めることに当たる．本研究では，単純問合せとして指定した構造表現が問合せ結果部分文書で出現する回数が少ないものを検索結果の部分文書中から選ぶことにする．

単純問合せにおいての検索結果となる最大の部分文書は，元文書それ自身である．前述の定義6から単純問合せの述語を満たすノードは，カレントノードと文書ノードの間の経路上のノードで，このノードを根とする部分文書が検索結果候補である．これらの部分文書で，問合せで指定された構造表現の出現が少なく，それ以外の構造（要素型）の出現が多い部分文書を検索結果する．そのために検索結果となる部分文書の根ノードを求める必要がある．このノードを文脈ノードとして定義する．この文脈ノードを単純問合せの条件を満たす論理構造上の極小部分文書の根ノードとすることで，単純問合せを満たす構造上の文脈の範囲を求める．

定義 7 (要素ノードと属性ノードに対応する文脈ノード： $c(n)$) 単純問合せ q, XML文書 d, $T(d)$ を d に対応した木，中間ノード集合 $Node_{T(d)}$ 中のカレントノード $n \in curr(q)$ について， n の文脈ノード $c(n) \in C(q)$ を次のように定義する：

1. n が最上位の要素ノード（文書ノードの子ノード）の場合，すなわち親ノードが文書ノードの場合は， $c(n)$ を文書ノードとする．
2. n が最上位の要素ノード以外の要素ノードの場合， $c(n)$ は，次の条件を満た

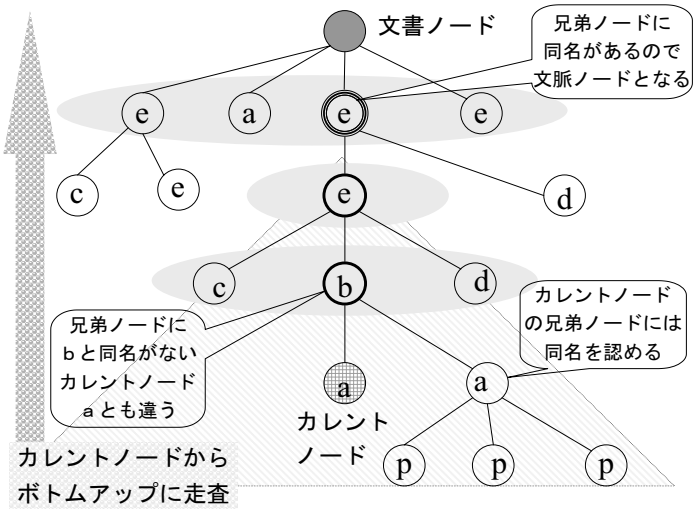

図 3.8 カレントノードと文脈ノードの関係

す D 中のノードとする：

(a) $p(n)$ を n の親ノードとする. $c(n)$ は， $p(n)$ と文書ノードの間の経路上にあるノードである．言い替えると，$c(n)$ は，n の祖先ノードであり，しかも 文書ノードかその子孫ノードである.

(b) $T(d)_{c(n)}$ を $c(n)$ を根ノードとする d 中の部分木とする． $T(d)_{c(n)}$ は，$p(n)$ と $c(n)$ の間の経路中のノード mが兄弟ノードに m と同じ要素名を持たず，しかも m とその兄弟ノードに n と同じ要素名を持たない最大の部分木である.

3. nが属性ノードの場合は， $c(n)$ をその属性を所有する要素ノードとする.

図 3.8 がカレントノード，文書ノードと文脈ノードの関係を示している.

XML 文書中のカレントノードから文脈ノードを求めるアルゴリズムを付録 B に述べた.

この文脈ノードが，問合せすべての場合において，最適な部分文書を特定していることは保証できないが，ある程度の有効性を持つことを 3.8 節の実験において確認する.

カレントノードに対応した文脈ノードを求める関数 **context** を導入する．この関数によって，問合せ条件を満たす経路に注目してその条件の構造上の文脈の単位となる文脈ノードに対応した部分文書を自動的にみつけることを目的としている．

定義 8 (関数 context) XML文書 d 中のノード n について，定義7で定義した文脈ノード $c(n)$ を求める関数を **context** とする．

定義 9 (ノード集合に対する文脈ノード関数 context) 文書 d 中のノード集合 $\mathbf{N} = \{n_1, \ldots, n_k\} \subset Node_{T(d)}$ に対して, 関数 **context** を次のように定義する:

$$\mathbf{context}(\mathbf{N}) = \{context(n_j) | n_j \in \mathbf{N}, j = 1, 2, \ldots, k\}$$

この定義によって，ノード集合を返すXPath式をこの関数 **context** の引数に適用できる．

定義 10 単純問合せ q の意味 単純問合せ q の部分文書検索の意味を次のように定義する．

$$\mathbf{context}(\mathrm{curr}(q))$$

関数contextの結果の各ノードが，そのインスタンス文書中での問合せ中で指定した名前空間についての文脈の最大領域と考えられる．すなわち，同じ名前空間の同名の要素の他の値と排他的な部分木集合が求められる．

表 3.4 単純問合せを満たす文脈ノード $context(curr(Q1))$

文脈ノード	文脈ノードまでの経路式	dc:title	title	'XML'
#19	document('stream.xml')/stream/head/rdf:RDF/rdf:Description[3]	1回	1回	1回

例 6 例3の問合せ(Q1)について関数 **context** の結果は，次の通りである．表3.4は文脈ノードと文脈ノードまでの経路式，部分文書中の出現回数である．

$$\begin{aligned}
\mathbf{context}(curr(Q1)) &= \mathbf{context}(\texttt{//dc:title[contains(.//text(),'XML')} \\
&\quad \texttt{OR contains(@*,'XML')]}) \\
&= \mathbf{context}(\#21) \\
&= \{\#19\}
\end{aligned}$$

表 3.5 単純問合せを満たす文脈ノード：$context(curr(Q2))$

文脈ノード	文脈ノードまでの経路式	dc:creator	creator
#5	document(‘stream.xml’)/stream/head/rdf:RDF/rdf:Description[1]	1 回	1 回
#13	document(‘stream.xml’)/stream/head/rdf:RDF/ref:Description[2]	1 回	1 回

表 3.6 カレントノードと文脈ノードとの対応

カレントノード	同名の兄弟数	文脈ノード	文脈ノードまでの経路式
#1, #2, #3, #4,	1	#1	document(‘stream.xml’)/stream
#5, #13, #19	3	#1	document(‘stream.xml’)/stream
#6, #7, #8, #9, #10	1	#5	document(‘stream.xml’)/stream/head /rdf:RDF/rdf:Description[1]
#11, #12	2	#5	document(‘stream.xml’)/stream/head /rdf:RDF/rdf:Description[1]
#14, #15, #16, #17, #18	1	#13	document(‘stream.xml’)/stream/head /rdf:RDF/rdf:Description[2]
#20, #21	1	#19	document(‘stream.xml’)/stream/head /rdf:RDF/rdf:Description[3]
#22, #26	2	#1	document(‘stream.xml‘)/stream
#23, #24, #25	1	#22	document(‘stream.xml‘)/stream/scene[1]
#27, #28	1	#26	document(‘stream.xml‘)/stream/scene[2]

例 7 例 3 の問合せ (Q1) について関数 **context** の結果は，次の通りである．表 3.5 は文脈ノードと文脈ノードまでの経路式，部分文書中の出現回数である．

$$\begin{aligned} \mathbf{context}(curr(Q2)) &= \mathbf{context}(\texttt{//dc:creator}) \\ &= \mathbf{context}(\{\#9, \#16\}) \\ &= \{\#5, \#13\} \end{aligned}$$

表 3.6 は，図 3.2, 図 3.3 におけるカレントノード，文脈ノード，文脈ノードまでの経路式の一覧である．単純問合せを満たすカレントノードから自動的に文脈ノードが計算できる．

3.6 AND問合せ

3.4.1節で定義したように，部分文書検索の問合せqは，要素名あるいは属性名として指定する構造表現$s_k \in \mathbf{S}$にキーワード $t_k \in \mathbf{T}$を含むという述語**contains**の組の順序なしの集合である.（キーワードの指定がない場合も含む）

$$\begin{aligned} q \;=\; & \{\mathbf{contains}(t_1, s_1), \mathbf{contains}(t_2, s_2), \ldots, \mathbf{contains}(t_p, s_p), \\ & \mathbf{contains}(*, s_{p+1}), \ldots, \mathbf{contains}(*, s_r) | \\ & t_k \in \mathbf{T}, s_k \in \mathbf{S}, k = 1, 2, \ldots, p, \ldots, r\} \end{aligned}$$

本節では，問合せqを各述語間の関係を論理積と考えAND検索としてその意味を定義する.

AND検索における順序なし木の包含関係の問題

q_1, q_2を単純問合せとする．ここでは，問合せq_1と，問合せq_2で指定されている構造表現とキーワードからなる問合せ条件を両方満たす適切な部分文書を求めたい．しかし，定義6で述べた単純問合せを満たすノードは各問合せ中の述語を満たすカレントノードと文書ノードの間の経路上に存在するすべてノードであるため，q_1の述語を満たすノード集合$C(q_1)$とq_2の述語を満たすノード集合$C(q_2)$の要素の組み合わせは，両者に該当するノードの積となり，述語が増加すると各述語を満たすノードの組み合わせが指数オーダーとなり，爆発してしまう．しかし，利用者が必要としている部分文書は，同じ文脈上に両者の条件を満たすノードが存在することと考えれば，文脈の境界を超えた組み合わせを必要としない．また，両者の述語を満たすノードと両者のカレントノードとの関係は，順序なし木の包含問題と同様の問題である.

Kilpeläinenが分析した順序なし木の包含関係については，付録Aに掲載している．本研究におけるAND検索の順序なし木包含問題とKilpeläinenの違いは，問合せパターンを木で表した場合，Kilpeläinenは，木の根ノードのラベルを指定しているが，本研究では，この根ノードについて明示的な指定はしていないが，一つではなく数個存在するところである．図3.9が本研究における問合せ木Pと検索対象木Tの関係を示す.

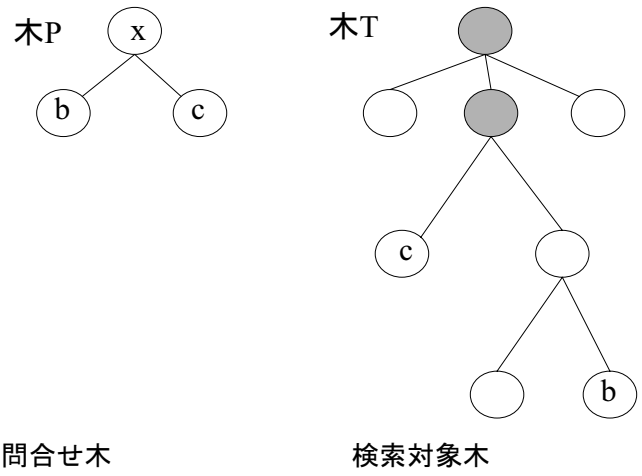

図 3.9 問合せ木 P と検索対象木 T の順序なし木の包含関係

Kilpeläinen は，この問題を解くために工夫をすることによって，検索対象木が順序のある木であり，問合せ木中のノード数が少ない場合は線形時間で解けることを証明している．従って，本研究においても同一要素の属性関に順序をつけることで，検索対象木を順序のある木とみなせば，AND 問合せ中の述語の数が少なければ線形時間で解ける．従って，AND 検索をノード集合間の共通部分とすれば各集合の要素であるノードをソートして比較することで解ける．

本研究では，この問題のすべての解を必要としているのではない．AND 問合せによって与えられた述語をそれぞれ満たすノードを含む部分文書を取り出すことが目的で，その方針は，なるべく小さな部分木中に各述語を満たすノードがそれぞれ一つ含まれることである．従って，AND 検索を各単純問合せ結果の論理結合とする場合が最も狭い文脈中で各述語を満たす部分文書が取り出せると考えられる．

定義 11 (AND 問合せの意味：AND) $q_1, \ldots, q_k \in \mathbf{Q}$ を単純問合せとする．AND 問合せの意味は，各述語をすべて満たすような部分文書検索とし，すべての問合せについて個々に求まる文脈ノード集合の共通部分とする．従って，

$$q_1 \ \mathbf{AND} \ q_2 \ \mathbf{AND} \ldots \mathbf{AND} \ q_k$$

の意味は,

$$\mathbf{context}(curr(q_1)) \cap \ \mathbf{context}(curr(q_2)) \cap \ \ldots \ \cap \mathbf{context}(curr(q_k))$$

である.

例 8 次の例は,「dc:titleとdc:creatorの両方の要素を同時に含み，両者の意味において一定の文脈を持つ適切な部分文書を探す.」問合せである．この問合せのAND問合せの結果は，次のとおりである：

$$\begin{aligned}
& \texttt{contains(* , dc:title)} \ \ \mathbf{AND} \ \ \texttt{contains(* , dc:creator)} \qquad (Q3) \\
= \ & \mathbf{context}(curr(\texttt{//dc:title})) \cap \mathbf{context}(curr(\texttt{//dc:creator})) \\
= \ & \mathbf{context}(\{\#7, \#15, \#21\}) \cap \mathbf{context}(\{\#9, \#16\}) \\
= \ & \{\#5, \#13\} \cap \{\#5, \#13\} \\
= \ & \{\#5, \#13\}
\end{aligned}$$

この問合せの結果，#5と#13を根ノードとする部分文書 $PD_{\#5}$，$PD_{\#13}$ が求められる．それぞれの根ノードまでの元文書中での経路式は,
document(‘stream.xml‘)/stream/head/rdf:RDF/rdf:Description[1],
document(‘stream.xml‘)/stream/head/rdf:RDF/rdf:Description[2]
であり，この経路式によって問合せの要素が元文書のリソースを記述した部分にあることがわかる.

ここで求められた二つの部分文書には，dc:titleとdc:creator がそれぞれ含まれていてAND問合せの意味「dc:titleとdc:creatorを含む部分文書」を満たし，しかも構造上これより大きな部分文書では，不必要な他のdc:title, dc:creatorを含んでしまうことから適当と考えられる．表 3.7は(Q3)のAND問合せの結果である.

表 3.7 AND 問合せ を満たすノード (Q3)

contains(*,dc:title) AND contains(*,dc:creator)

ノード	該当ノード までの経路式	dc:title	dc:creator
#5	document(‘stream.xml‘)/stream/head/rdf:RDF/rdf:Description[1]	1 回	1 回
#13	document(‘stream.xml‘)/stream/head/rdf:RDF/rdf:Description[2]	1 回	1 回

3.7 OR 問合せ

次に問合せ q の OR 検索の意味を定義する．部分文書検索の問合せ q は，要素名あるいは属性名として指定する構造表現 $s_k \in \mathbf{S}$ にキーワード $t_k \in \mathbf{T}$ を含むという述語 **contains** の組の順序なしの集合である.（キーワード の指定がない場合も含む）

$$\begin{aligned} q \;=\; & \{\mathbf{contains}(t_1, s_1), \mathbf{contains}(t_2, s_2), \ldots, \mathbf{contains}(t_p, s_p), \\ & \mathbf{contains}(*, s_{p+1}), \ldots, \mathbf{contains}(*, s_r) | \\ & t_k \in \mathbf{T}, s_k \in \mathbf{S}, k = 1, 2, \ldots, p, \ldots, r\} \end{aligned}$$

定義 12 (OR 問合せの意味：OR) $q_1, \ldots, q_k \in \mathbf{Q}$ を単純問合せとする．OR 問合せの意味は，それぞれの問合せを満たすような部分文書問合せについて個々に求まる文脈ノード 集合の和集合とする．従って，

$$q_1 \;\; \mathbf{OR} \;\; q_2 \;\; \mathbf{OR} \;\; \ldots \;\; \mathbf{OR} \;\; q_k$$

の意味は，

$$\mathbf{context}(curr(q_1)) \cup \mathbf{context}(curr(q_2)) \cup \;\ldots\; \cup \mathbf{context}(curr(q_k))$$

である.

例 9 次の例は,「dc:title に ‘XML‘を含むかdc:title に ‘SMIL‘のどちらかを含み，これらの文脈上適切な部分文書を探す.」問合せである.

表 3.8 OR問合せを満たすノード (Q4)

contains(‘XML’,dc:title) OR contains(‘SMIL’,dc:title)

ノード	該当ノードまでの経路式	dc:title	‘XML’	‘SMIL’
#5	document(‘stream.xml’)/stream/head/rdf:RDF/rdf:Description[1]	1回	0回	1回
#19	document(‘stream.xml’)/stream/head/rdf:RDF/rdf:Description[3]	1回	1回	0回

$$
\begin{aligned}
& \texttt{contains(‘XML‘ , dc:title)} \;\; OR \;\; \texttt{contains(‘SMIL’ , dc:title)} \qquad (Q4)\\
= \;& \mathbf{context}(curr(\texttt{//dc:title[contains(.//text(),‘XML’)}\\
& \texttt{or contains(./@*,‘XML’)]}))\\
& \cup \mathbf{context}(curr(\texttt{//dc:title[contains(.//text(),‘SMIL’)}\\
& \texttt{or contains(./@*, ‘SMIL’)]}))\\
= \;& \mathbf{context}(\{\#21\}) \cup \mathbf{context}(\{\#7\})\\
= \;& \{\#19\} \cup \{\#5\}\\
= \;& \{\#5, \#19\}
\end{aligned}
$$

この問合せの結果，#5と#19を根ノードとする部分文書 $PD_{\#5}$，$PD_{\#19}$ が求められる．それぞれの根ノードまでの元文書中での経路式から，
document(‘stream.xml’)/stream/head/rdf:RDF/rdf:Description[1],
document(‘stream.xml’)/stream/head/rdf:RDF/rdf:Description[3] によって問合せの要素が元文書のリソースを記述した部分にあることがわかる．

表 3.8は (Q4) の OR問合せの結果である．

3.8 実験データによる部分文書検索

本節では，本章で定義した問合せモデルによる単純問合せ，AND問合せとOR問合せについて実験データを用いて，行なった実験について述べる．

従来の検索エンジンでは，ファイルを検索対象としていたため，部分文書単位の検索はできない．本実験では，出力インタフェースとして既存のものを利用す

ることで，視覚的な比較が可能なため，検索結果の部分文書をファイルとして出力し，表示することにした．本研究で必要としている部分文書情報は、2 章の定義 2 で定義した XML 構造表現であり，必ずしもファイルである必要はない．

3.8.1 実験システム

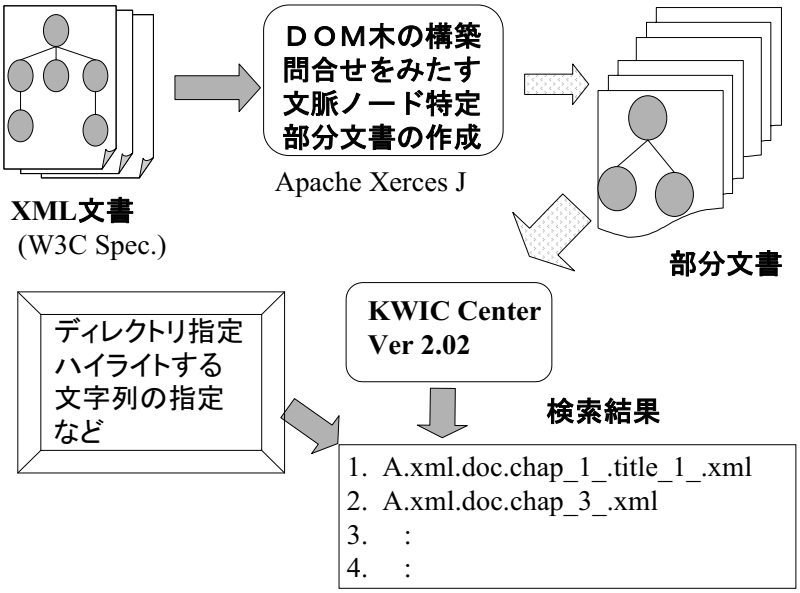

図 3.10 実験システム概要

図 3.10 が実験システムの概要である．本実験では，各問合せに対して，問合せ条件を満たす部分文書をファイルとして出力することによって，既存のキーワード検索システムに検索結果を表示させる．

その処理の流れは次の通りである：

1. 検索対象となる各 XML ファイルごとに XML ファイル中で宣言されている展開可能な実体を展開し，文字コードを統一する． 対象とする XML ファイ

ルは，W3CのXMLに関連した最近の仕様書[2]でXML文書として公開されている12のファイルを用いた．その結果これらの文書中の中間ノードに対応した部分文書の総数は，39905個であった．

2. 問合せを名前空間，要素（属性）名，キーワードの組で与え，(1) この条件を満たす文脈ノードを求め，求めた文脈ノードに対応した部分文書をファイルとして書き出す．(2) この条件を満たすカレントノードと文書ノードの間のノード（単純問合わせを満たす候補ノード）に対応した部分文書をファイルとして書き出す．このときに，ファイル名は，元のXMLファイル名に該当するノードまでの経路式を結合したファイル名として書き出す．ただし，‘/’を‘.’で置き換え，各要素のインデックスを‘[ ]’で囲むのではなく，‘_’を使いディレクトリと区別した表現とした．問合せ結果のXML部分文書ファイル集合を一つのディレクトリに入れる．
3. 問合せ結果の部分文書ファイルを公開されているソフト「KWIC Centre」[3]に読み込ませて表示する．このKWIC Centreは，マイクロソフトWindows上のGREP風検索システムで，全文検索クライアントを組み込んで利用可能なシステムである．ディレクトリとファイルを指定してGREP同様のキーワード検索を行ない，出現キーワードを中心とした一行を表示する機能とそのファイル中の出現キーワードを強調表示する機能がある．本研究の文脈ノードを利用した部分文書検索の出力表示に，ふさわしい機能をそなえている．

単純問合せ

実験データにおいて，(1) 単純問合せを文脈ノードに対応した部分文書を利用した場合と (2) 各問合せの単純問合せを満たす候補ノードを利用した場合の検索結果の比較である．

図3.11が単純問合せ`contains(*, def)`の検索結果を表示した画面の例である．残念ながら，利用したXMLファイルの中に共通の名前空間は存在しないため，名

[2] http://www.w3.org/TR/

[3] http://plaza3.mbn.or.jp/~h_ishida/

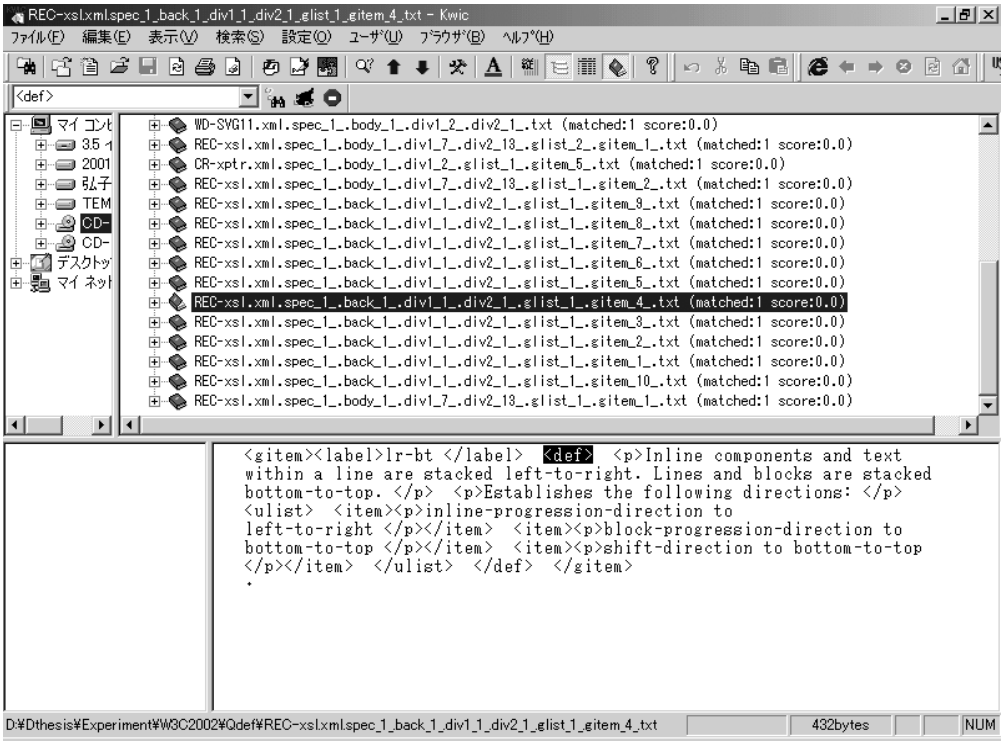

図 3.11 単純問合せ contains(*, def) の検索結果表示

前空間指定の要素についての問合せはできないが，これらのファイルがほぼ似た DTD を利用しているため，共通に利用されている要素型を代用することにした．この画面の右上部には，文字列<def>が含まれているファイル名とそのファイル中での文字列<def>の出現回数が表示されている．この単純問合せ結果には，どのファイルにも<def>が 1 回出現していることが確認でき，問合せ述語をみたしていることも，容易に確認できる．さらに，経路式から，ファイル中の構造上の位置がわかる．図の下部には，選択した部分文書の文書内容を表示している．この表示方法は，キーワード検索の結果表示をキーワードの前後に出現する文字列と一緒に表示する KWIC を目的としているため，検索結果の部分文書の前後の内容も一緒に表示される．

表 3.9 と表 3.10 は，単純問合せを (1) 文脈ノードに対応した部分文書を利用し

表 3.9 単純問合せ結果部分文書と候補部分文書の比較

問合せ	(1) 文脈ノード利用 部分文書数	(2) 候補ノード利用 部分文書数
contains(*,def)	819	1308
contains(*,term)	300	614
contains(*,bibl)	18	55
contains('XML',label)	2	10
contains('XML',term)	5	14
contains('example',head)	5	15
contains('context',note)	12	16
contains('character',titleref)	13	30
contains('tree',figure)	9	19
contains('XML',p)	293	559

た場合と (2) 各問合せの単純問合せを満たす候補ノードを利用した場合の検索結果の比較である．候補ノードには，文脈ノード集合に比べ大きな部分文書が含まれ，これらには、同じ構造名（要素名や属性名）のタグが複数存在しているため一定の文脈を持つ部分文書としては，不適切であることがわかる．

AND問合せとOR問合せ

表3.11は，実験データにおいて，(1)AND検索を文脈ノードに対応した部分文書を利用した場合と (2) 各問合せの単純問合せを満たす候補ノードを利用した場合の検索結果の比較である．図3.12は，AND検索のKWIC表示例である．平均回数は，指定した要素が各部分文書内に出現した回数の平均を，回答ファイル数は，検索結果の部分文書数を示している．この結果から，(1) 各問合せの文脈ノードを利用したAND問合せの結果は，指定した要素を一つ以上含むが出現回数が少ない部分文書を結果としている．つまり，(2) の候補ノードを利用すると，部分文書内に複数の文脈が出現している可能性が高くなり，部分文書の大きさも文脈ノードを利用したときよりも大きいものが多くなる．従って，この実験のAND問合せから次の問題点が明らかになった．

(1) 文脈ノードを利用すると，単純問合せ同様，同一文脈の極小部分文書が得ら

表 3.10 contains(‘XML’,term) の (1) 文脈ノード 利用部分文書と (2) 候補ノード 利用部分文書の比較

ファイル名	(1)	(2)
REC-xml.xml.spec_1_.body_1_.div1_1_.p_3_.termdef_1_.@term.xml		√
REC-xml.xml.spec_1_.body_1_.div1_1_.p_3_.termdef_1_.xml	√	√
REC-xml.xml.spec_1_.body_1_.div1_1_.p_3_.xml	√	√
REC-xml.xml.spec_1_.body_1_.div1_1_.xml		√
REC-xml.xml.spec_1_.body_1_.div1_2_.div2_8_.p_1_.term_1_.xml		√
REC-xml.xml.spec_1_.body_1_.div1_2_.div2_8_.p_1_.xml	√	√
REC-xml.xml.spec_1_.body_1_.div1_2_.div2_8_.xml		√
REC-xml.xml.spec_1_.body_1_.div1_2_.p_1_.xml	√	√
REC-xml.xml.spec_1_.body_1_.div1_2_.xml		√
REC-xml.xml.spec_1_.body_1_.xml		√
WD-charmod.xml.spec_1_.body_1_.div1_6_.note_1_.p_1_.xml	√	√
WD-charmod.xml.spec_1_.body_1_.div1_6_.note_1_.xml		√
WD-charmod.xml.spec_1_.body_1_.div1_6_.xml		√
WD-charmod.xml.spec_1_.body_1_.xml		√

れる．しかし，各述語が同じ文脈ノードを持つという条件は，厳しく，現に，表3.11のcontains(‘example’,head) AND contains(‘context’,note) のように元の文書においては，この問合せ条件を満たしているが，AND 問合せでは、それらの組み合わせが検索結果にはならない場合があること．

(2) 候補ノードを利用すると，AND 問合せよりは，条件がゆるいため，より大きな部分文書が結果となっている．しかし，指定した要素もキーワードも複数存在するため，AND 問合せの意味を逸脱してしまっていること．

3.8.2 実験の考察

本実験から得られた結果は次の通りである：

- 単純問合せでは，文脈ノードを利用した部分文書が、問合せ述語を含み、同じ構造名（要素名や属性名）を部分文書中に含まない極小の部分文書となっている場合が多かった．

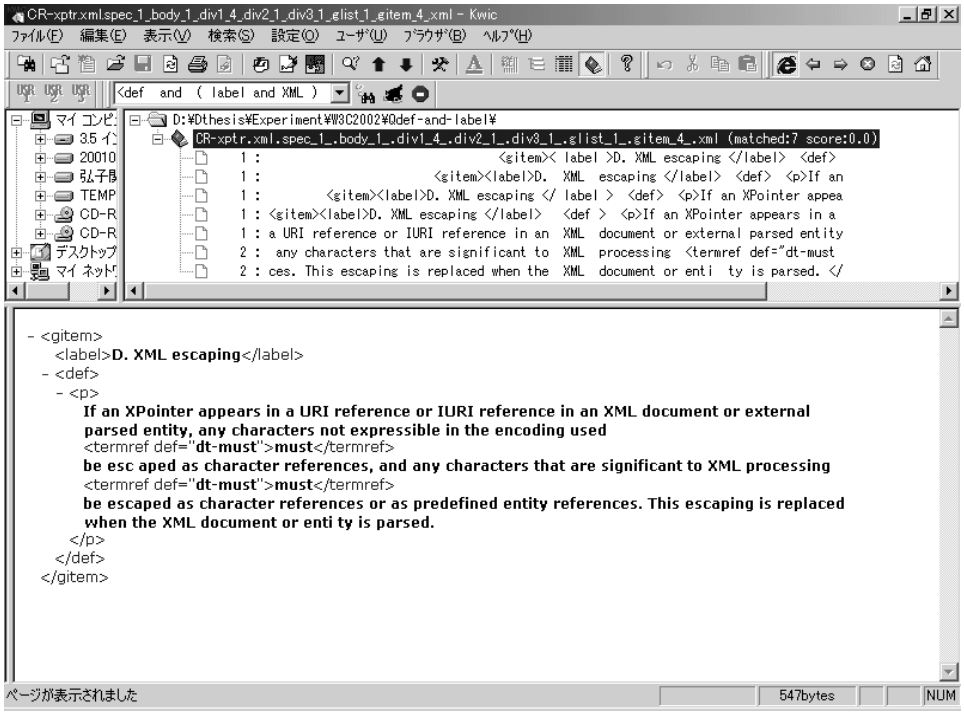

図 3.12 AND 問合せ contains(‘XML’,label) AND contains(*,def) の検索結果表示

- 指定した構造名が XML 文書中の木構造の上位ノードである場合は，多数の同名の構造を含んでいる場合があった．これは、部分文書を必ず元の論理構造の木構造を保持した部分木と対応させていることから下部構造の枝がりを考慮することによって解決可能と思われる．

- 構造とキーワードの両方を指定した単純検索では，問合せを満たす部分文書が抽出できた．しかし，AND 問合せで取り出された部分文書は当初の要件を満たしているが，今回定義した AND 問合せの意味は条件がきついために，同一文脈上に問合わせ述語を満たす部分が存在しないために検索されなかった部分文書でも、利用者に必要とするものが存在する可能性がある．

- OR 問合わせでは，部分文書数が多くなり，利用者がこれらをすべて見るこ

表 3.11 AND 問合せ (1) 文脈ノード利用部分文書と (2) 候補ノード利用部分文書の比較

問合せ	(1) 文脈ノード利用要素出現平均回数	(1) 文脈ノード利用回答ファイル数	(2) 候補ノード利用要素出現平均回数	(2) 候補ノード利用回答ファイル数
contains('XML',label) and contains(*,def)	1 1	2	8 8	10
contains(*,term) and contains(*,def)	2 1	3	36 27	23
contains('example',head) and contains('context',note)	0 0	0	80 70	3
contains('example',head) and contains('XML',p)	1 5	1	54 318	7

とは不可能である．表 3.12 は，いくつかの OR 検索結果数を表しているが検索結果の部分文書数が多い．そのため、条件を満たす部分文書を分類したり，順位をつけて利用者に提示する必要がある．

3.9 まとめ

本章では，利用者が検索対象 XML 文書についての論理構造についての完全な知識は持たないが，標準語彙中お要素や属性名についての知識を持つ場合の部分文書検索について述べた．

実験を通して，文脈ノードを利用した部分文書を検索結果とするほうが，問合せの述語を満たすすべての候補ノードを利用した部分文書を検索結果とするより，同一文脈の範囲での部分文書を取出すことができた．さらに，従来の KWIC 表示方法を利用して，しかも利用者の問合せに適合する部分とその周辺の構造上同一文脈とみなされる部分までを表示することで，利用者が結果から新たな情報を得ることが示せた．

次に，利用者に検索対象 XML 文書についての論理構造についての知識がない場合の部分文書検索について述べる．

表 3.12 OR問合せ (1) 文脈ノード利用部分文書と (2) 候補ノード利用部分文書の比較

問合せ	(1) 文脈ノード利用回答ファイル数	(2) 候補ノード利用回答ファイル数
contains(‘XML’,label) or contains(*,def)	819	1338
contains(*,term) or contains(*,def)	1116	1727
contains(‘example’,head) or contains(‘context’,note)	17	38
contains(‘example’,head) or contains(‘XML’,p)	297	567

第4章

キーワード指定による XML 文書集合における部分文書検索

本章では，利用者に検索対象 XML 文書集合についての論理構造についての知識が全くない場合の部分文書検索について述べる．本章の部分文書検索の前提と目的は次の通りである：

1. 利用者が文書の論理構造についての知識を持たないため，問合せの入力はキーワードの列挙とする．
2. 利用者の入力キーワードに関連した適切な粒度の部分文書を抽出すること．
3. 入力キーワードの出現する構造上と内容に関する文脈から，利用者が問合せ内容をより明確にすることが可能なこと．
4. 利用者の問合せと文書内容や構造の類似度を考慮して検索結果を順位付けすること．

まず 4.1 節では，XML 文書に対する検索での情報検索に関する関連研究について述べる．次に 4.2 節で本章における問合せモデルを，4.3 節で文書構造が固定されている XML 文書集合における部分文書特定について 4.4 節では文書構造が固定されていない XML 文書集合に対して，論理構造上の単位となる文書部分の特定法について述べ，4.5 節では，利用者の問合せとの関連性の高い文書部分の抽出法について述べる．4.6 節で，評価実験について述べ，4.7 節はまとめである．

4.1 XML 文書検索での情報検索に関する関連研究

現在の構造化文書検索は，選択条件および出力文書構造を XML 問合せ言語を用いて宣言的に指定する方法と Web サーチエンジンにみられる情報検索技術によるキーワード入力による全文検索に分類される．本節では，XML 文書検索での情報検索機能に関する研究について述べる．

4.1.1 XML 問合せ言語の情報検索機能拡張

3.1 節で述べた W3C が提案している XQuery[58] では，情報検索の問合せ向けに必要なキーワードの重み付けや結果の順位付けを表現する機能はほとんどない．さらに，利用者があらかじめ文書構造についての知識を持っていて，検索結果の構造を宣言的に指定することを前提としている．

構造化文書の検索を情報検索としてとらえた場合 XML 問合せ言語に情報検索機能を追加する必要性があるため，XML 問合せ言語への情報検索機能拡張が提案されている．

XML Query Engine[30] は，XQL にテキスト検索演算子を追加している．しかし，検索結果の順位付けは考慮されていない．また，利用者が検索結果文書の構造を指定しなければならない．

XSD[21] は，XML 文書構造をスキーマディレクトリとしてまとめて，構造に関する検索に利用する．ApproXQL [45, 44] は，XML 問合せと文書中の部分木構造との類似度を木から木への変換コストとして計算する．

ELIXIR[9] は，XML–QL をテキスト類似演算子で拡張し，文書内容についての類似度から検索結果の順位付けを行なっている．

XML 問合せ処理とキーワード検索を統合する手法が Florescu らによって提案されている [14]．この研究では， XML 検索とキーワード検索との統合利用を目的として XML–QL[15, 16] を拡張し，利用者が文書構造を知らない場合の問合せを想定しているが，取り出す要素型の条件を利用者が指定する必要がある．次の問合せは，利用者が XML 文書についての構造を知らない場合，article 要素以下の文書構造の深さ 3 までに “Dingle”, “1999”, “web” を要素名，要素内容，属性

名，属性値のいずれかに含む場合の article 要素を取り出す問合せである．

```
WHERE <article></article> ELEMENT_AS $E IN ‘‘bib.xml’’
      contains($E, ‘‘Dingle’’ 3, any),
      contains($E, ‘‘1999’’, 3,  any).
      contains($E,‘‘web’’, 3, any),
CONSTRUCT $E
```

XXL[2] は SQL 形式の問合せ言語だが，問合せ条件に要素型名，属性名，テキストに曖昧性を持たせオントロジーやシソーラスを利用して問合せと文書中の出現した語の確率を求めて検索結果に順位付けを行なう言語である．しかし，利用者が文書構造についての一部の知識を持つことが前提となっている．同じ大学のプロジェクト Niagara サーチエンジン [34] は，XML–QL を拡張した言語による問合せを要素型と内容で構成される検索エンジン問合せ言語に変換して XML 問合せに情報検索エンジンを統合する手法をとっているが，順位付けはない．

XIRQL[37, 38] は，宣言的 XML 問合せ言語である XQL[41, 39] を情報検索技術が利用できるよう拡張した言語であり，入力キーワードと問合せに重みを指定することができる．さらに，関連性指向の検索のために，問合せ結果の文脈単位となる要素型をデータベース管理者があらかじめ定めて，この要素型を単位として索引を作る．XIRQL では，文脈検索を想定してはいるが，利用者があらかじめ文書構造についての知識を持つことが前提となっている．また，具体的な実装法については言及されていない．

4.1.2 情報検索技術に基づく検索エンジン

従来の情報検索技術は，主にプレーンテキストを対象としてきたため，内部に表題，著者，抄録，章や節などの構造があってもシステムが自動的にこれらの構造を識別することは困難で，文書構造を利用した情報検索は，あらかじめ同じ文書構造を持つ SGML 文書 [25, 26, 27] などの構造化文書に限定されていた．また，検索結果は常に文書全体であるという前提があった．しかし，XML をはじめ構

造化文書に対しての情報検索では，各文書を走査することで文書中に記述されている章や節などの構造をシステムが解析することができるため，システムが解析した文書構造を利用して各構造を単位とした情報検索を行なうことができる．文書構造を利用した情報検索の研究は XML の出現により，ますます重要となっている．

HTML 文書を探す目的で情報検索技術に基づく多数の検索エンジンが登場している．検索エンジンが利用者のいくつかのキーワード入力に対して，HTML ページの中から利用者の問合せを満たす文書の情報を抽出する．一般的にこれらの検索エンジンにおける検索結果の提示は，関連度でランキングされた文書のリストで該当文書の要約や該当文書へのリンク先を伴っている．利用者は，検索結果の一覧からリンクされている文書ファイルへアクセスする．検索結果の初期画面では，入力キーワードがハイライトされるが，実際の該当文書の表示の際，入力キーワードとの関連性の強い文書部分だけをぬきとって表示したり，関連性の弱い文書部分と識別表示することができない．XML では文書構造と文書出力フォーマットを分けて管理でき，検索結果を多様な出力形式に変換することができるので，関連性の強弱を検索結果に反映させることも可能である．

現在の HTML 検索エンジンが検索結果を向上させるために次のような工夫をしている [12].

- **Title タグ:** 検索結果に文書の <TITLE> タグを利用してタイトルを表示する．
- **META Keywords タグ：** 文書中のタグ内容を利用してこの文書のキーワードを検索対象とする．
- **META Description タグ：** 文書中のタグ内容を利用してこの文書の要約を検索と検索結果の表示に利用する．
- **META Robots タグ：** このタグ内容によって，文書作成者が検索ロボットがこの文書中のリンクをたどったり，この文書中から抽出した文字を検索インデクサが格納することを認めるか否かを決める．
- **Links:** 検索ロボットは，文書中のリンクをたどる．

文書中のメタデータは，統制された語彙を利用するため，メタデータを利用する検索は，データベースのキーワード フィールド 同様，XML 検索エンジンにおいても重要である．現在 Dublin Core, UKOLN, Meta Matters, XML.com, Adobe XMP などが公開され，RDF と共に XML 文書のメタデータを記述する場合に利用されている．

XML を利用した検索エンジンや XML 文書を対象とした検索エンジンがいくつか公開され始めている．XSet[5] は主記憶上のデータベースと検索エンジンの組合せで XML をデータ格納言語として利用している．また，XML 文書検索に情報検索の技術を導入し文書の各要素型の特徴量を文書の葉にあるテキストに索引づけをして，上位構造の要素ノード は下位ノード の出現値を積算する BUS(Bottom Up Scheme)[19] を利用した検索システム XRS[18] がある．XYZFind[13] は，利用者の問合せを支援する対話的なシステムで， 利用者に問合せ結果のスキーマを示すことで検索結果の絞りこみを支援している．これら XML 検索エンジンの試みも利用者が入力する少ない検索用語からよりよい検索結果を求める目的は我々と共通する．しかし，いずれも DTD で定義した文書構造を持つ XML 文書を前提にしているところが我々の研究とは異なる．我々の提案する手法は，整形式の文書も対象とした検索手法である．

4.1.3 問合せの曖昧性による関連研究の分類

表 4.1 は，文書構造と内容に関する検索条件による関連研究の分類である．

XML 文書に対する問合せを分類するため，利用者が問合せで入力する条件（内容，構造）と出力条件（特定の構造を指定するか全く構造を指定しない，順位付け）によって次のように分類できる．入力する条件は，採用する IR モデルと曖昧性や順位付けを認めるかどうかによって分けられる．

- 問合せに文書構造を指定するか

  (S) 利用者が問合せ中に文書構造を指定する場合

  (S-1) 要素型名，属性名を指定した検索（前後，包含関係，経路式）

表 4.1 文書構造と内容に関する問合せの分類

	S 文書構造					
	指定検索				指定しない検索	
		あいまい検索				
	S-1 要素型名，属性名を問合せに指定	S-2 要素型，属性の出現頻度を検索に考慮	S-3 問合せで要素型名，属性名にあいまい性指定	S-4 問合せと文書構造の木構造類似性で順位付け	NS-1 検索に文書構造利用しない	NS-2 検索に文書構造利用する
C-1 値，キーワード指定（近接，包含関係）	Q1,Q3,Q7		Q6	Q5		
	XQL[41, 39] XPath[55] XML-QL[15, 16] XQuery[58]	XPath[55] XQuery[58]	XML-QL[15, 16] XXL[2]	XSD[21]		**本研究)** [32, 61, 60]
C-2 ワイルドカードや正規表現を指定するキーワード検索拡張（検索結果の順位付けなし）	Q8			Q10		
	Florescu[14] Niagara[34]		XXL[2]	ApproXQL [45, 44]		Florescu[14]
C-3 重み付けや類似度を指定する IR 拡張（検索結果の順位付けあり）	Q2,Q4			Q9	**従来の情報検索**	
	ELIXIR[9] Hayashi[59] XRS[18]	XPRES[47]	XIRQL[38] XXL[2]		Passage[43, 29]	XIRQL[38] **本研究** [63, 62, 24]

下線は，部分文書検索に対応

(S-2) 要素型，属性の出現頻度を考慮した検索（要素型，属性の転置ファイル）

(S-3) 要素型名，属性名にワイルドカード，正規表現，シソーラスを利用した検索（文書処理，パターンマッチ）

(S-4) 問合せと文書構造の木構造類似性による構造検索（順位付けあり）

(NS) 利用者が問合せ中に文書構造を指定しない検索（変数束縛を含む）

(NS-1) 検索システムが文書構造を利用しない（プレーンテキストと同じ）

(NS-2) 検索システムが文書構造を利用する（部分文書検索）

- 利用者が問合せ中に文書内容を指定するか

(C) 利用者が問合せ中に文書内容を指定する検索

(C-1) 文書内容の値，キーワードを指定する検索（近接，包含関係）

(C-2) 文書内容の値，キーワードにワイルドカードや正規表現を指定するキーワード検索拡張（検索結果の順位付けなし）

(C-3) 文書内容の値，キーワードに重み付けや類似度を指定する IR 拡張（検索結果の順位付けあり）

(NC) 利用者が問合せ中に文書内容を指定しない検索

これら検索には，完全照合を前提とした検索から正規表現や範囲指定，類似度を利用した曖昧性を許す検索まで考えられる．現在提案されている XML 問合せ言語では，データベース問合せ言語の SQL 同様，検索対象文書構造，検索条件および出力文書構造を宣言的に指定するが，問合せの値を範囲指定したり，キーワードにワイルドカードや正規表現の指定を想定しているだけで，検索結果の順位付けや入力キーワードへの重み付けは想定されていない．また文書構造の指定にワイルドカードや正規表現を想定しているものは，ほとんどない．一方，Web サーチエンジンにみられる情報検索技術による全文検索では検索対象文書構造や出力文書構造は，物理構造上の単位であるファイルが基本であり文書中の論理構造を利用した情報の単位による検索については，議論されてこなかった．

問合せの曖昧性による分類

(Q1) 構造と内容条件の論理結合：XML 問合せ言語で記述可能 – (S-1, C-1)

```
Find <doc>s containing ‘pararell’ AND (‘computing’ OR
‘processing’)
Find <coress>s with attribute CONFIDENTIAL = YES
```

(Q2) 構造条件と類似内容条件：section を単位としたランキング – (S-1, C-3)

```
Find <section>s about ‘parallel processing’
```

(Q3) 構造条件： XML 問合せ言語で記述可能 – (S-1, NC)

```
Find elements where the first child is <title>
```

(Q4) 構造条件と内容条件の論理結合：XML 問合せ言語で記述可能 –(S-1,C-2)

> `Find articles, papers and books with ‘parallel AND processing’ in the title`

(Q5) 利用者が XML 文書についての構造を知らない場合，文書構造の深さを指定してその部分木内にキーワードを要素名，要素内容，属性名，属性値のいずれかに含む場合の部分文書へのアクセス：(NS-2,C-1)

> `Find elements containing the words ‘‘Dingle’’, ‘‘web’’, and ‘‘1999’’ in the subtree of depth 3`

(Q6) 文書中の構造についてのあいまいな知識からキーワードを含む部分文書へのアクセス – (S-2, C-3)

> ‘‘XQL’’ を重み 0.6，‘‘syntax を重み 0.4 で含むような要素型を持つ <section> を順位つきで取り出す.

(Q7) 文書中の構造についてのあいまいな知識から順位つきで似た構造の部分文書へアクセスする – (S-4, C-1)

> ラフマニノフの CD で講評や説明のあるものを取り出す.

(Q8) Abiteboul らは XML のビューについて, ビューワークスペース管理に関連した議論で, XML 問合せの意味をもっと深く考慮すべきであると述べている [1]. 例えば, 次の問合せを考える– (S-3, C-3)

> *XML* **の論文の著者は誰か***?*

この問合せについて次の SQL 風の構文を使って, 各々の論文についての特定のデータを返すことができるだろう：

```
select P
from Mybiblio.paper P
Where P.keyword contains ‘‘XML’’
with P.title, P.abstract, P.author.*
```

ここでの “with” 句の目的は, 明示的に返されるものを指定することである. すなわち, `title`, `abstract`, `author` からたどれるすべてのデータである[1] . この場合は, 同じ DTD を持つ XML 文書インスタンスを対象としているため, paper 以下の論理構造が DTD よりわかっているので “with” 句で限定することができる.

この場合は, 同じ DTD を持つ XML 文書インスタンスを対象としているため, paper 以下の論理構造が DTD よりわかっているので “with” 句で限定することができる.

整形式の XML 文書については, XML 文書の問合せ言語には, XML 文書インスタンスの内容, 構造に関する記述に加え, 期待される問合せ結果を求めるために問合せのセマンティクスを追加する記述が有用である.

XML 文書への問合せ言語は, 多数提案されている. 現在提案されているいくつかの XML 問合せ言語 (XQL[40, 39], XML-QL[15] など) は, ひとつの XML インスタンス中の内容と構造に関する問合せに限られている.

4.2 問合せモデル

2 章の定義 3 に従い，文書構造を指定できない場合の問合せモデルに変更する. すなわち，次の通りである：

[1] 彼らは, ‘with’ 句が構文として標準のものではないことを知っている. しかし, 彼らは, XML の問合せ言語は, 問合せ結果にはりついている（接着している）概念（考え）を述べる何らかの構文の必要性を信じている. ここでは, “with” 句で示してあるが, これは, 他の方法例えば, *select* 句の一部としてもよい.

定義 13 (文書構造を指定しない問合せ) $\mathbf{D}$ を XML 文書集合，$\mathbf{S}$ を $\mathbf{D}$ 中の XML 文書 d の各部分文書集合の構造表現の和集合，$\mathbf{T}$ を $\mathbf{D}$ 中のキーワード集合とする．

部分文書検索の問合せ q は，m 個 $(m > 0)$ の述語 $q_{(t,*)} = \mathbf{contains}(t, *) = \bigcup_{\forall s_k \in \mathbf{S}}\{\mathbf{contains}(t, s_k)\}$ の集合として定義する：

$$q = \{q_{(t_i,*)} | \quad t_i \in \mathbf{T}, i = 1, 2, \ldots, m\}$$

本章で提案する部分文書検索は，2 章の定義 4 で述べた部分文書検索モデルの (4) 論理構造上の単位となる文書部分の特定と部分文書を作成するアルゴリズム，文書内容を用いた利用者の問合せとの関連性の高い文書部分の抽出，で構成される．まず論理構造上の単位となる文書部分を特定し，検索対象部分文書の作成方法について，文書構造が固定されている XML 文書集合の場合と文書構造が全く固定されていない XML 文書集合の場合に分けて述べる．

4.3 文書構造が固定されている XML 文書集合における部分文書特定

高度に構造化された XML 文書には，文書構造を制約する DTD や XML スキーマが付随する．これらの XML 文書はデータベースで管理することが比較的容易なため，XML 文書の構造をデータベーススキーマに写像して文書を管理することでデータベース技術を利用することができる．構造と内容に関する問合せにも DTD や XML スキーマの知識を利用して宣言的に行うことができる．本節では，同一の DTD や XML スキーマに従っている XML 文書集合に対するキーワードを指定した部分文書検索について述べる．

ここでは，同一の DTD に従っている XML 文書集合において各文書中から論理構造を考慮した文書部分を部分文書として特定する方法について述べる．

この方法は，XIRQL[37] で採用している方法と同様ドメインアナリストの解析を必要とする．検索対象とする XML 文書集合が同一の DTD に従っている妥当な文書の場合は，文書の走査なしに，DTD を見ることによって文書構造を理解できる．従ってドメインアナリストが論理構造上の単位となる文書部分を最上位の要素ノード名で指定することができ，それを検索対象部分文書とすることがで

```
1    <!ELEMENT book    (toc, chapter*)>
2    <!ELEMENT toc     (#PCDATA)>
3    <!ELEMENT chapter (titlepage, section*)>
4    <!ATTLIST chapter label #IMPLIED>
5    <!ELEMENT titlepage (title, author)>
6    <!ELEMENT section (title, (subsec* | para*))>
7    <!ELEMENT title   (#PCDATA)>
8    <!ELEMENT author  (#PCDATA)>
9    <!ELEMENT subsec  (para*)>
10   <!ELEMENT para    (#PCDATA)>
```

図 4.1 Book.xml の文書型定義 (DTD) : pub.dtd

表 4.2 ドメインアナリストが取り出す部分文書に対応した経路式

指定ノード	該当ノードまでの経路式
#4	book.xml.book.chapter.xml
#11	book.xml.book.chapter.section.xml
#20	book.xml.book.chapter[2].xml
#27	book.xml.book.chapter[2].section.xml
#30	book.xml.book.chapter[2].section.subsec.xml
#39	book.xml.book.chapter[2].section.subsec[2].xml
#2	book.xml.book.toc.xml
#6	book.xml.book.chapter.titlepage.xml
#22	book.xml.book.chapter[2].titlepage.xml

きる．例を挙げて部分文書の特定の方法を説明する．

例 10 図 4.1 は，book を最上位要素型とした XML 文書の DTD である．図 4.2, 図 4.3 は，この DTD に従った XML 文書インスタンスとその木構造表現である．この DTD から，この文書中に出現する要素型名と要素の出現順序，入れ子関係がわかる．DTD 中で要素型の複数回出現を +，*で指定している展開された要素型名[2] は，section，subsec， chapter，para である．これらの繰り返しは，文書内容の区切りを表している．ここで，para は，子ノードがテキストノードだけのため，テキストの区切りではあるが，内容としての区切りとみなすには，粒度が小さ過ぎることから子ノードがテキストノード以外のノードを含むsection，subsec，chapterが，構造上の境界となる要素ノードと考えられる．また，展開された要素型名からドメインアナリストは，内容の境界と判断することができる．

[2] この XML 文書インスタンスには名前空間の指定がないので，名前空間を表す URI は null となる．

```
1 <?xml version="1.0"?>
2 <!DOCTYPE book SYSTEM ‘‘Pub.dtd’’>
3 <book>
4  <toc>XML Information Retrieval</toc>
5  <chapter label=‘XML’>
6  <titlepage>
7    <title>XML Data Model</title>
8    <author>Kinutani</author>
9  </titlepage>
10  <section>
11    <title>Tree Structure</title>
12    <para> XML is becoming widely used.</para>
13    <para>We have developed algorithms. </para>
14    <para>A structure is represented
                                as a tree.</para>
15  </section>
16 </chapter>
17 <chapter label=‘IR’>
18  <titlepage>
19    <title>IR Systems</title>
20    <author>Hatano</author>
21  </titlepage>
22  <section>
23    <title>Boolean model</title>
24    <subsec>
25     <para>We propose a heuristic method</para>
26     <para>Context search</para>
27    </subsec>
28    <subsec>
29      <para>Result subdocuments</para>
30    </subsec>
31  </section>
32  <section>
33    <title>The vector space model</title>
34    <para>Exact matching may lead</para>
35  </section>
36 </chapter>
37</book>
```

図 4.2 本の XML 文書インスタンス例 : Book.xml

従ってこの場合，#4, #11, #20, #27, #30, #35, #38, #39 の部分文書を論理構造上の単位とすることが DTD から適当と考えられる．さらに各部分のメタ情報に関する部分文書，#2, #6, #22 をドメインアナリストが論理構造上の単位として追加する場合もある[3]．表 4.2 は，図 4.1 の DTD に従った図 4.2 の XML インスタンスについて指定したノードと経路式である．

以上特定の DTD に対応した XML 文書の部分文書特定方法について述べた．部分文書の特定はドメインアナリストの解析の手法により，対象 XML 文書によって異なる．しかし，文書中に使われている要素型，属性名を手がかりに，前もって決めることができる．

[3] XIRQL では，構造化文書モデルとして重複のないリストモデルを採用しているので “section” に着目すると {#1, #2}， {#4, #5, #6 以下 }，{#11 以下 }， {#20, #21, #22 以下 }，{#27 以下 }, {#38 以下 } に対応した部分文書に分割している．

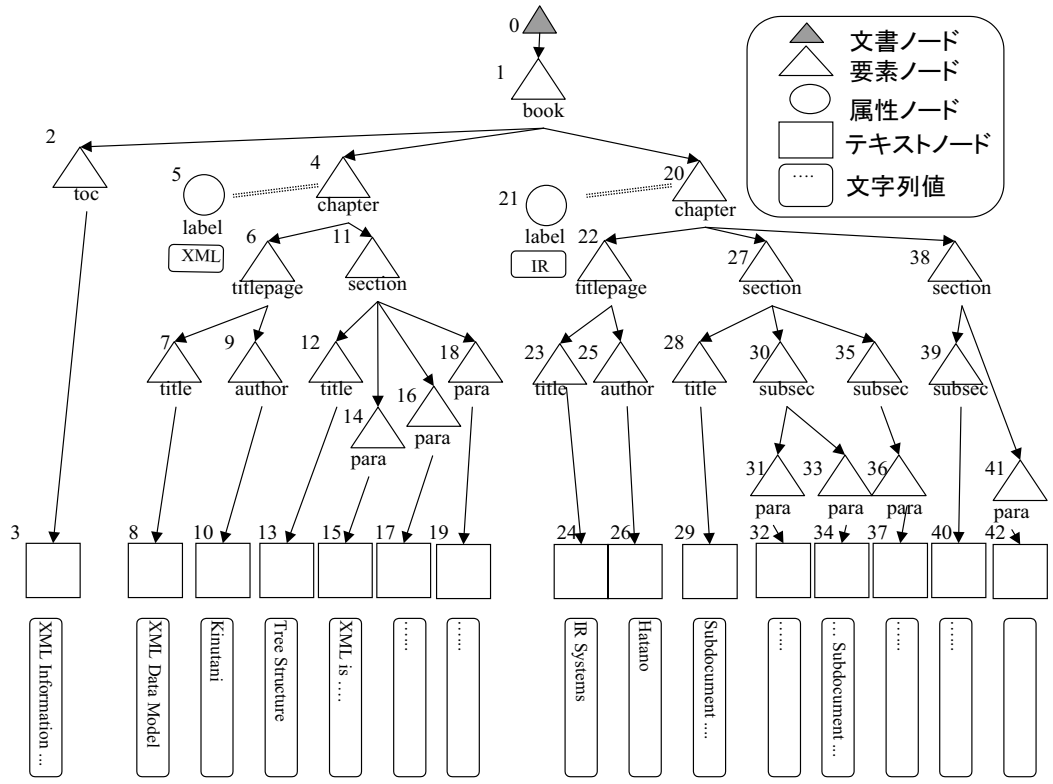

図 4.3 Book.xml の木構造表現

4.4 文書構造が固定されていない XML 文書集合における部分文書特定

例 11 次の単純問合せを，利用者が検索システムに入力し，検索システムがこの問合せとの関連性の高い文書部分を検索結果とする．

$$\{\mathbf{contains}(\text{`}\texttt{XML}\text{'}, *), \quad \mathbf{contains}(\text{`}\texttt{model}\text{'}, *)\} \qquad (Q6)$$

この問合せの意味は，*XML* 文書集合からキーワード *"XML"* と *"model"* に関する内容を持つ文書部分をみつけることである．

部分文書を特定するための方法として (1) すべての XML 文書インスタンスから共通のスキーマを抽出してシステム管理者が部分文書を指定する方法，(2) 各

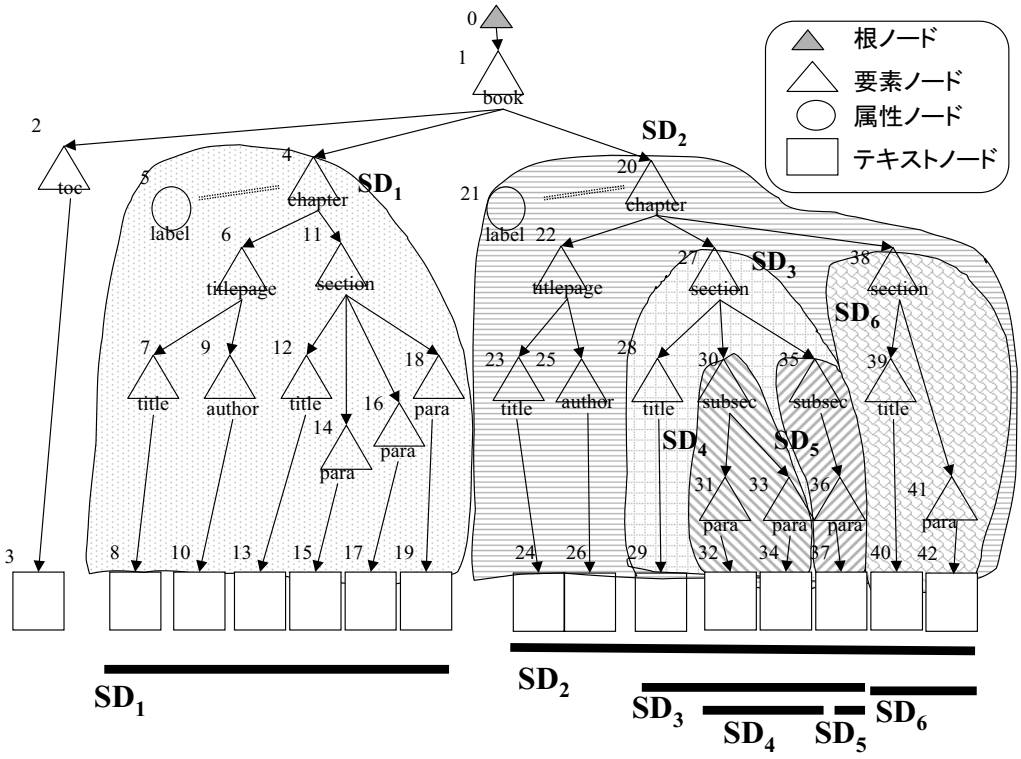

図 4.4 文脈ノードと対応した部分文書 $SD_i(i = 1, \ldots, 6)$

XML 文書インスタンスから部分文書を自動的に特定するためのアルゴリズムを構築する方法，が考えられる．前者は，整形式の XML 文書インスタンスからスキーマを抽出する研究 [42, 22] を適用できる．しかし，各 XML 文書インスタンスごとに抽出したスキーマが異なる場合は，抽出されたスキーマごとにシステム管理者が論理構造上の単位となる文書部分を指定する必要があるため，XML 文書構造の種類が少数の場合に適するが，種類が増加するとシステム管理者の負担が増加する．後者の方法は XML 文書構造の種類が多い場合にも適用できるため我々はこの方法を採用する[4]．

本研究で採用した XML データモデルでは，すべての文字列値はテキストノー

[4] XML 文書インスタンス内のタグも索引語と同一視して情報検索の手法で扱う研究もあるが，構造名を自然言語と同一に扱えるほど多様ではないのが現状である

ドか属性ノードが持っている．一方検索対象の XML 文書インスタンスの構造について意識しない利用者の，XML 文書内容に関する問合せは，従来の HTML サーチエンジンと同様，いくつかのキーワードの列挙であると想定する．利用者の要求は，入力キーワードを内容に持つ文書であり，入力キーワードを構造に持つ，すなわち要素型名や属性名として持つ文書ではない．そのため，入力キーワードと一致する文字列値を直接持つ各テキストノードと属性ノードに対して次に定義する文脈ノードを使って構造上の単位となる部分文書の特定に利用する．

本節で提案する手法で特定される部分文書が完全にシステム管理者が文書中の文脈の境界を各文書ごとに解析する結果と一致するわけではないが，このアルゴリズムによってシステム管理者の処理を軽減することができると考える．

定義 14 (テキストノードに対応する文脈ノード) XML 文書 D 中のテキストノードあるいは属性ノードを n とする． D 中の n に関する**文脈ノード** $c(n)$ は，n のある一つの祖先ノードであり，次のように定義する：

1. 属性ノード $n \in D$ の場合，$c(n)$ は n の親に当たる要素ノードである．
2. n がテキストノードの場合，n の親ノードまたは祖父母（親の親）ノードを $p(n)$ とし，最上位の要素ノードを n_d としたとき，$c(n)$ は，経路 $(p(n), n_d)$ 中のノードのうち，次の条件を満足し，しかも $p(n)$ に最も近いノード m である：

   m の同じ親を持つ兄弟関係にある要素ノードで，m と同じ展開された要素名を持つものが存在する．ただし，このようなノード m が存在しない場合は，n_d を $c(n)$ とする．

   なお，$p(n)$ を n の親ノードとするか祖父母ノードとするかは次の方法によって選択する．

   (a) n に兄弟ノードがある場合，すなわち n の親ノードの内容が混在内容である場合は，$p(n)$ は n の親ノードとする．

   (b) n に兄弟ノードがない場合は，$p(n)$ は n の祖父母ノードとする．

定義 14 の 2.(b) によって，n に兄弟ノードがない場合は，n の親ノードは文脈ノードとはなり得ないことになる．この場合，n の文字列値がそのまま親の要素ノードの文字列値となるが，この親の要素ノードは構造上の文脈の境界単位としては粒度が 小さすぎると考えるためである．

表 4.3 ノードと対応する文脈ノード，部分文書

テキスト ノード	文脈 ノード	極小 部分文書	ノードを含む 部分文書
#3	#1		
#8,#10, #13, #15, #17,#19	#4	SD_1	SD_1
#24, #26	#20	SD_2	SD_2
#29	#27	SD_3	SD_2,SD_3
#32,#34	#30	SD_4	SD_2,SD_3,SD_4
#37	#35	SD_5	SD_2,SD_3,SD_5
#40	#38	SD_6	SD_2,SD_6
属性 ノード	文脈 ノード	極小 部分文書	ノードを含む 部分文書
#5	#4	SD_1	SD_1
#21	#20	SD_2	SD_2

定義 15 (極小部分文書) XML 文書 D 中のテキストノードあるいは属性ノード n に対して，定義 14 で定義した文脈ノード $c(n)$ を根とする D 中の部分木に対応した文書を n の**極小部分文書**と呼ぶ[5] .

XML 文書中のすべてのテキストノードあるいは属性ノードに対する極小部分文書の集合を構造の上から文脈の単位として特定された部分文書集合とする.

例 12 図 4.2，図 4.3 の XML 文書インスタンスを例にして文字列値を直接持つ各ノードに対応する文脈ノードを示す．図 4.4，表 4.3 がテキストノード，属性ノードとそれらの文脈ノード，特定された部分文書との関係である．XML 文書インスタンス中に入力キーワードと一致する文字列が存在する場合，極小部分文書として特定された 6 個の部分文書 $SD_i(i = 1, \ldots, 6)$ が文書構造上抽出される[6] .

4.5 文書構造が固定されていない XML 文書集合における部分文書抽出法

前節で，部分文書検索の (4) 論理構造上の単位となる文書部分の特定法について述べた．本節では，(5) 利用者の問合せとの関連性の高い文書部分の抽出法について述べる．まず，部分文書検索が採用する IR モデルについて述べる.

[5] n の**極小部分文書**は他のノードの極小部分文書を含むことも，また他のノードの極小部分文書に含まれることもある.

[6] 一般にキーワード検索において元の文書自体を検索結果とする必要性は少ない．必要ならば元文書も抽出単位と考えると 7 個となる.

この検索モデルに全文検索に用いられているブーリアンモデルを適用する．

4.5.1 ブーリアンモデルによる部分文書検索

ブーリアンモデルでは，入力キーワードと文書内容を比較し，入力キーワードを含むか否かを判定し，問合せ条件を満たす文書が検索結果となる．従って，4.4 節で特定された部分文書を対象として，入力キーワードを含む文書が検索結果となる．

例 13 図 4.2，図 4.3 に示した XML 文書インスタンスを例に，例 11 の問合せを満たす部分文書を考える．キーワード “XML” は，ノード番号#3, #5, #8, #15 で示されているノードの文字列に含まれる．これらのノードに対応した極小部分文書は，表 4.3 から SD_1 である．一方キーワード “model” は，ノード番号#8, #29, #40 に含まれ，対応した極小部分文書は，SD_1, SD_3, SD_6 である．さらにこれらの部分文書を含む SD_2 も ‘model” を含む部分文書の抽出候補である．従って “XML” と “model” を含む部分文書は二つのキーワードを同時に含む部分文書 SD_1 となる．つまり例 11 の検索結果として SD_1 の部分文書が特定される．

検索システム

我々の部分文書検索を目的とした検索システムは，XML 文書から部分文書を作成する処理と索引の構築，部分文書検索に分けられる．図 4.5 が本システムの概略図である．

DOM 木からの部分文書を作成

XML 文書から部分文書を作成する処理は，次の通りである：

1. 表記統一 ： XML 文書中の不要な空白を除去し文字コードの統一を行なう (canonicalize).
2. XML 文書解析と DOM 木構築 ： XML プロセッサ (Apache Xerces version 1.2.2[7]) を利用して展開可能な実体を展開し，DOM 木を主記憶上に作る．

[7] http://www.apache.org/

3. DOM 木中の各テキストノードと属性ノードから文脈ノードを計算し，文脈ノードを再上位ノードとする部分文書を作成してディスクに格納する．

索引構築と部分文書検索

部分文書をファイル形式で出力し，索引構築と部分文書検索は，従来から行なわれている全文検索の手法を利用する．今回は，フリーソフトである namazu version 2.0.5[8] を利用した．索引構築と部分文書検索の処理は次のものである：

1. 索引構築： 部分文書すべてを読み込み索引ファイルを作成する．
2. キーワード検索： 問合せキーワードを論理式，正規表現で指定する．検索結果は，namazu のアルゴリズムで高いスコア順にファイル名とファイルの先頭数行が出力される．スコアは TF/IDF で求められる．

4.6 評価実験

本節では，提案した部分文書分割手法の有用性を評価するために行なった実験について述べる．

実験環境

実験に用いたデータは，著者らの所属している研究室で独自に作成したテストコレクションである．いまだ， XML 文書の評価用テストコレクションで公開されたものは存在しないため， W3C の XML に関連する HTML 形式の 17 個の仕様書[9] を XML 文書に変換したものである．問合せ／解答セットは研究室で作成した次の 3 種類である．

- **問合せ/解答セット 1**

  質問文 XHTML の互換性の問題は将来どう解決されるのか?

[8] http://www.namazu.org/
[9] http://www.w3.org/TR/

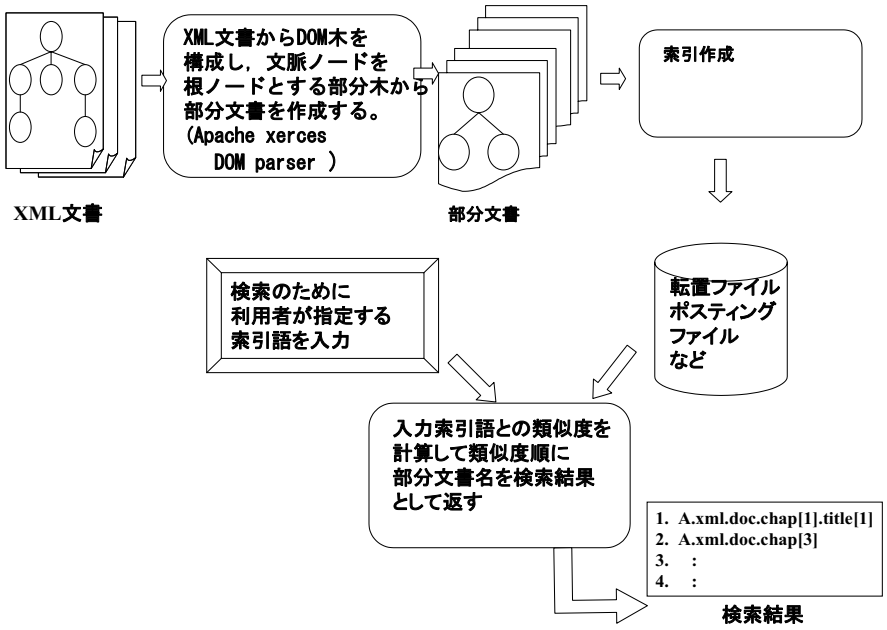

図 4.5 XML 文書検索システムの概略図

問合せキーワード XHTML compatible issue future direction.

解答 `xhtml1-20000126.xml` の 5 章および 6 章

- **問合せ/解答セット 2**

  質問文 XML のエンティティの文字コードは UTF-8 の他に何が利用できるのか?

  問合せキーワード XML entity character encoding UTF-8.

  解答 `REC-xml-19980210.xml` の 2.2 節および 4.3.3 小節，付録 F 章と，`REC-xml-20001006.xml` の 2.2 節および 4.3.3 小節，付録 F.1 小節.

- **問合せ/解答セット 3**

  質問文 XML の要素型名や属性名に使える文字には何があるのか?

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT spec
  (title, authlist, abstract, status, body, back)>
<!ATTLIST spec
    w3c-doctype (cr|note|pr|rec|wd)      #IMPLIED
    other-doctype CDATA                  #IMPLIED
    status (int-review|ext-review|final) #IMPLIED>
<!ELEMENT title (#PCDATA) >
<!ELEMENT authlist (author+) >
<!ELEMENT author (name, affiliation?, email?) >
<!ELEMENT name (#PCDATA) >
<!ELEMENT affiliation (#PCDATA) >
<!ELEMENT email (#PCDATA) >
<!ELEMENT abstract (p+) >
<!ELEMENT status (p+) >
<!ELEMENT body (div1+)>
<!ELEMENT div1 (div2*|p+)>
<!ATTLIST div1 id NMTOKEN #REQUIRED>
<!ELEMENT div2 (div3*|p+)>
<!ATTLIST div2 id NMTOKEN #REQUIRED>
<!ELEMENT div3 (p+)>
<!ATTLIST div3 id NMTOKEN #REQUIRED>
<!ELEMENT back (div1+)>
```

図 4.6 テストコレクションを定義する DTD

問合せキーワード attribute element type name character code qualify

解答 `REC-xml-names-19990114.xml` の 2, 3, 4 章および付録 A.3 小節と `xml-19980210.xml` および `xml-20001006.xml` の 2.2, 2.3, 3, 3.1 節および付録 B 節.

本実験では， 提案した部分文書検索を行なう場合の検索対象となる部分文書の違いによる検索結果の比較を目的とし，次の三種類の方法によって実験を行なった.

1. 全中間要素ノード アプローチ： 全中間要素ノードを根ノードとする部分木に対応した部分文書. すなわち，2 章で定義した部分文書集合，$PD(d) = \{\ pd(d)_{n_i} n_i \in Node_{T(d)}\ \}$ とする．この方法は，最も効率は悪いが単純な方

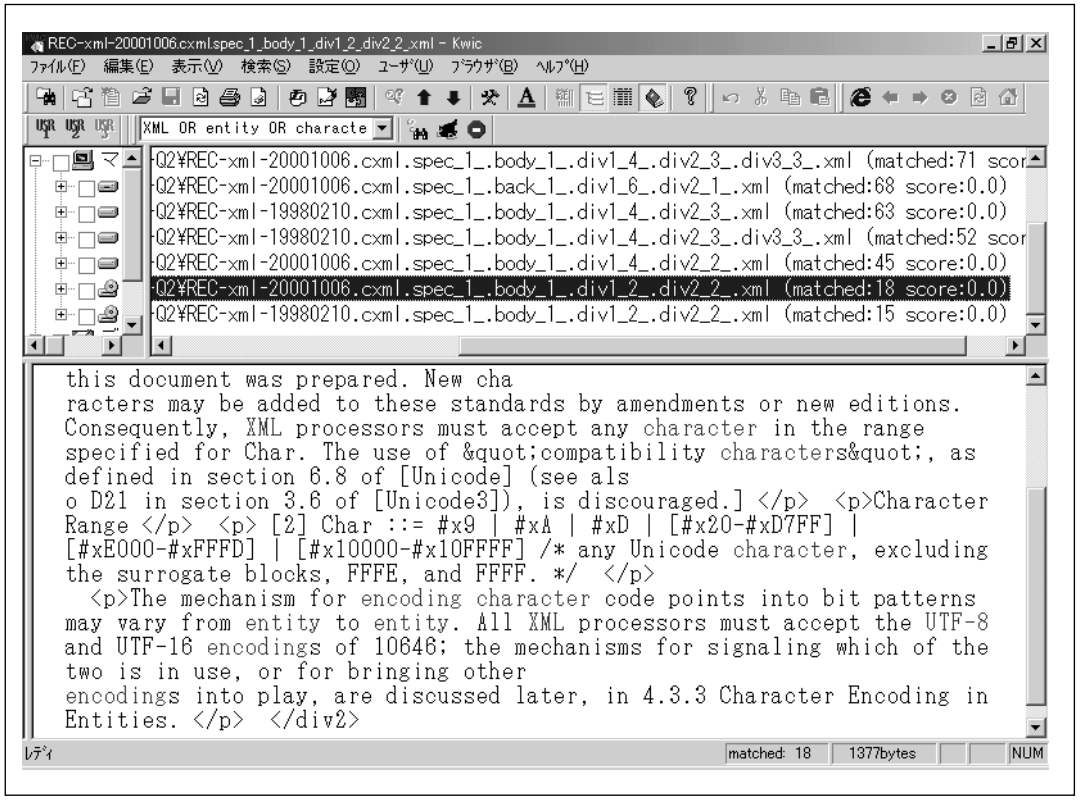

図 4.7 **問合せ/解答セット 2** の検索結果の表示

法である.

2. 選択ノードアプローチ：4.3 節に述べた文書構造が固定されている XML 文書集合の場合に作成した部分文書．XML 文書を定義する DTD がある場合で，管理者による解析の結果選ばれた要素ノードに対応した部分文書であるため，取り出したいと管理者が期待する部分文書を検索対象とすることができる．図 4.6 がテストコレクションの構造を定義している DTD であり，この DTD を解析した．構造上の単位となる要素型として `title`, `authlist`, `abstract`, `status`, `div1`, `div2`, `div3` を指定した．
3. 文脈ノードアプローチ： 4.4 節に述べたテキストノードに対応した文脈ノードを根ノードとする部分文書．特定の DTD との対応のない XML 文書に利用する．ただし，語数 25 未満の部分文書は，粒度が小さすぎることから検索

対象から除外した．

4.6.1 実験結果

我々のテストコレクションから得られた部分文書数は，(1) 全中間要素ノード 4,344，(2) 選択ノード 1,145，(3) 文脈ノード 1,172 である．図 4.7 が**問合せ/解答セット 2** の結果を表示したところである．

実験結果の評価に当たり，本実験においては，検索結果として取り出した部分文書が正解部分文書より大きい文書や小さい文書があり，検索結果文書ごとに正解部分文書と適合しているか否かを一意に判定することができない．また全中間要素ノードを対象部分文書とした (1) の場合は，結果の上位を粒度の大きいファイルが占める結果となっている．これは使用した検索ソフト namazu のスコアの計算において，入力キーワードの頻度計算に文書の大きさを考慮していない点が考えられる．

そこで本実験では検索結果のスコアを文書の大きさで調整した値を新たなスコアとして定義し，検索結果をランキングしなおした．さらにテストコレクションの正解部分集合のうち検索システムによって検索された正解部分の割合を表す再現率として，正解を含む（あるいは正解に含まれる）粒度の違う部分文書が出現した場合，該当順位までにそれらの部分文書によって検索された正解部分の割合の合計（最大 1）を利用することにした．一方，システムによって検索された部分文書のうち正解部分文書集合と合致している部分の割合を表す適合率には各検索結果文書に含まれる正解の割合を（検索された適合文書部分）/（検索された適合文書部分 + 検索されなかった適合文書部分 + 検索された非適合文書部分）の式で求めて利用することにした．本実験では再現率と適合率を新たに次のように定義した．

定義 16 (再現率，適合率) a をテストコレクション中の正解部分文書数，$\{SD_1, \ldots, SD_a\}$ を正解部分文書集合，$\{RD_1, \ldots RD_k\}$ をスコアの高い順に並べた k 個の検索結果文書リストとする．また文書 X の文書の大きさを $|X|$ と表す．スコアが k 番目の検索結果文書 RD_k の再現率 R を，

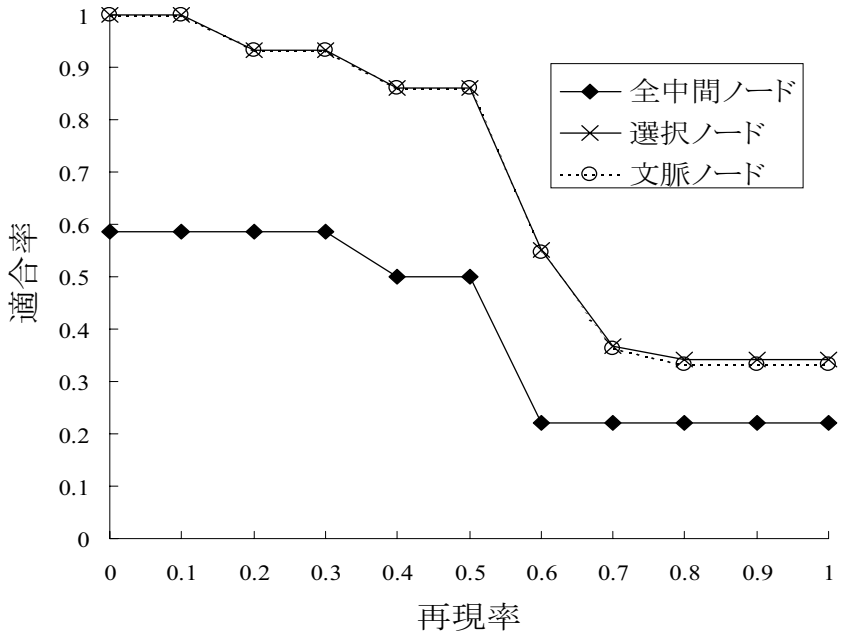

図 4.8 粒度違いを考慮した再現率-適合率グラフ（問合せ１～３平均）

$$R = \frac{1}{a}\sum_{j=1}^{a}\frac{|(\bigcup_{i=1}^{k} RD_i) \cap SD_j|}{|SD_j|} \tag{4.1}$$

とする．この定義式の分子の意味は，1 番目から k 番目までの検索結果文書の正解文書中のリージョンに関する和集合と正解部分文書集合の共通部分の割合を各正解文書を *1* として合計したものである．従って分子の最大値は正解文書数 a となる．

スコアが k 番目の検索結果文書 RD_k の適合率 P を，

$$P = \frac{1}{k}\sum_{i=1}^{k}\frac{|RD_i \cap (\bigcup_{j=1}^{a} SD_j)|}{|RD_i|} \tag{4.2}$$

とする．この定義式の分子の意味は，k 個の検索結果文書に含まれる正解の割合の合計である．

本実験では，この定義によって再現率–適合率のグラフを描いた．図 4.8 の再現率-適合率グラフは問合せ/解答セット 1〜3 の検索結果の平均を (1)「全中間要素

ノードアプローチ」，(2) 「選択ノードアプローチ」，(3)「文脈ノードアプローチ」について比較したものである．文書の大きさを調整した後の検索結果の順位から文書の大きさが小さいほど上位に位置付けられ，正解に近い大きさの部分文書が上位を占めることがわかった．また，検索結果図 4.8 から (1) 全中間要素ノードを対象部分文書とするより，(2), (3) のようにある程度検索されるべき部分文書を限定した方が検索適合率がよいことがわかった．

これらの検索対象部分文書は，文書構造から文脈の単位となる文書であり，検索適合率がよいことから文書中の内容に関しても意味ある単位となっていると考えられる．従ってキーワードの出現密度が他の部分より高い部分を求めることで入力キーワードに関連した部分を取り出すことができたと考える．また (2) と (3) による手法を比較すると検索対象部分文書数において (2) が若干少なかったが検索結果には両者に著しい差が認められなかったことから，我々が提案した文脈ノードを利用した部分文書を対象とする手法によって DTD を見てシステム管理者が検索対象部分文書を指定する手法に近い検索結果を導けることがわかった．

これらの実験結果から，構造化文書から利用者の問合せに最適な部分文書を取り出すために，DTDが利用できる場合は，システム管理者が部分文書を最上位の要素を指定することで，DTDが利用できない多様な XML 文書に対しては，我々が提案する手法を使って求めた極小部分文書を対象として検索する方法が有効であることを示すことができたと思われる．

4.7 まとめ

本節で提案した部分文書抽出手法は，

1. 利用者の問合せは，従来のインターネットでの検索エンジンの利用法と同じであり，利用者が新たに問合せのための準備を必要としないこと．
2. 従来の情報検索システムが培ってきた技術を利用できること．
3. 特定の DTD に対応していない XML 文書であっても，文書構造を走査するだけで検索対象部分文書の作成が可能なこと．
4. 文書構造中のすべてのノードについて部分文書を作成する必要がないこと．

5. 元文書のごく一部に利用者の問合せに強く関連した部分がある場合，その部分を検索できること．

以上の成果をもたらした．

本章では，キーワード集合として問合せを与える場合の部分文書検索について，1) 元の XML 文書中の検索対象となる索引語について，個々にその索引語を含む構造上の最小文脈となっている部分文書をキーワード入力に関する回答候補として選ぶアルゴリズムについて提案し，2) 実際の問合せキーワードから回答候補の部分文書集合の中から問合せキーワードとの関連性の高い部分文書を探した．3章での前提は，XML 文書に使われている要素名とキーワードの入力であれば，共通のモジュールを持たない XML 文書集合にも適用可能である．その場合に問合せキーワードとの関連性の高い部分文書を探すために，本章て提案したアルゴリズムを適用することも可能である．

第5章
結論

本研究では，XML 文書から利用者の問合せに最適な部分文書を文書構造と文書内容の両者を利用して取り出す手法について述べた．従来の文書検索では，検索結果は常にファイルを単位としていたため，構造化文書の構造を利用した検索は，利用者が構造についての知識をあらかじめ持っている場合に限定されていた．しかし，利用者が文書の構造についての知識がない場合にも問合せ可能で，利用者への負担の少ない問合せの入力，利用者の欲しい文書部分を探す方法は，生活のあらゆる場面に計算機が配置され，利用されるようになればなるほど必要である．

本研究では，XML 文書の部分文書検索に関する研究における二つの着眼点に関して，それぞれ次のような主な成果を得た．

(1) 文書構造とキーワード指定による部分文書検索

i) 問合せ中の述語を満たす部分文書を限定するための文脈ノードの導入．この文脈ノードを根とする部分木に対応する部分文書中には，少なくとも問合せ中の述語を満たす部分を 1 箇所以上含み，文書構造上一つの文脈情報の単位となっていることが確認できた．

ii) 部分文書検索における単純問合せ，AND 問合せと OR 問合せの意味づけと定義．単純問合せは，問合せ述語を満たす構造上の同一文脈を保持した部分を求める． AND 問合せでは，複数の述語を同時に満たす部分文書として，各述語を満たす文脈ノード集合の共通部分に対応した部分文書集合を結果とする．また，OR 問合せは，各述語を満たす文脈ノード集合の和集合に対応した部分文書集合を結果とする．その結

果，AND 問合せでは，各述語を満たす粒度の小さい部分文書を取り出すことができた．また OR 問合せでは，各述語を満たす単純問合せ結果の部分文書集合の和集合として，指定した構造は，各文書内に 1 回以上出現するが，その出現回数は，そう多くならないことを確認した．

(2) キーワード指定による部分文書検索

i) 文書構造上検索対象とすべき部分文書の特定手法．入力キーワードは，文書内容中の文字列と比較されるものなので，文書内容中の各文字列に対応した部分文書を XML 文書を走査することによって特定する．その結果，各文書内容に対応した文書構造上，同一文脈である部分文書を検索対象として保持することが可能である．

ii) 情報検索手法による部分文書抽出法の提案．問合せ中のキーワードは，部分文書中でも 1 回以上出現している．しかし，出現回数の多いキーワードも存在し，AND, OR 問合せでは，問合せ結果が多きすぎたり，問合せ結果の検証が困難である．そのため，問合せ結果に問合せとの関連度を計算して，順位付けを行なう必要があり，実験では，拡張ブーリアンモデルによって検証した．

(3) 文書構造とキーワード指定による部分文書検索の場合においても，単純問合せや OR 問合せでは，検索結果の部分文書数が多くなり，これらを KWIC 表示で見る場合にも，やはり利用者に問合せとの関連度による順位づけが必要と思われる．その場合には，キーワード指定による部分文書検索で提案したようなキーワードや指定した構造名についての出現回数を利用した順位付の関数が有効となるはずである．したがってキーワード指定による部分文書検索で提案した手法は，文書構造とキーワード指定による部分文書検索においても適用できる．

5.1 今後の課題

本研究において次の課題解決を必要としていることが判明した．

1. 利用者が文書構造を知らずに構造化文書検索を行なうには，本論文で提案した部分文書検索をきっかけとした対話的な検索行動を支援する枠組が重要と考える．そのためにも検索結果から文脈情報を利用者が得られる KWIC 表示は重要である．さらに，将来の検索エンジンは，コード化された文書情報とコード化の困難な利用者が持つ問合せの背景にある文脈情報とのギャップを埋めるために，1 回目の検索結果から絞り込み検索を行なう方法と問合せの変更による再検索について考察する必要がある．
2. 今回は，実験にふさわしい XML 文書がみつからず，手に入る W3C の仕様書を利用した．また適合率，再現率を出すためには，テストコレクションと問合せと解答セットも必要だが，部分文書を解答とするようなテストコレクションは，まだ存在しなかったため，研究室で独自に作成した．そのため対象とする XML 文書数が少ないこともあり，他のテストコレクションによる評価が必要である．
3. 本研究では，モデル上は，すべての部分文書も XML 文書であるので，すべての部分文書をファイルとして既存のツールを利用して有効性を検証した．しかし，部分文書情報は，元の XML 文書中にあるので，効率的な部分文書検索のためには，文書情報を保持する効率的な索引を必要とする．

謝 辞

本研究を行うにあたり，適切な御指導ならびにきめ細かなご配慮を賜わりました植村俊亮教授に心から深く感謝の意を表し，御礼を申し上げます．植村先生には，ご多用，ご多忙中にも関わらず本研究の主指導教官になっていただき，本論文の草稿に対しまして，多数のコメントをいただきました．

ご多忙中にも関わらず本研究の副指導教官になっていただきました松本裕治教授に深く御礼申し上げます．

ご多忙中にも関わらず本研究の副指導教官になっていただきましたお茶の水女子大学の増永良文教授に深く御礼申し上げます．

ご多忙中にも関わらず本研究の副指導教官になっていただきました吉川正俊教授に深く御礼申し上げます．

研究会などを通じて，本研究への御助言，および日々の研究生活において，多大な御協力と御支援をいただいた増永研究室の皆様に感謝いたします．

本研究に対する有益な御助言および御協力をいただいたマルチメディア統合システム講座の皆様に感謝いたします．

最後に，本研究に当たり，研究に対する理解と支援をしてくれた夫，絹谷雅生に感謝いたします．

参考文献

[1] Serge Abiteboul. On Views and XML. In *Proc. of the 18th ACM Symposium on Principles of Database Systems*, pages 1–9. ACM Press, May 1999.

[2] Anja Theobald and Gerhard Weikum. Adding Relevance to XML. In *Proc. of the Third International Workshop on the Web and Databases, WebDB2000*, 2000.

[3] Ricardo Baeza-Yates and Gonzalo Navarro. Integrating Contents and Structure in Text Retrieval. *ACM SIGMOD Record*, Vol. 25, No. 1, pp. 67–79, March 1996.

[4] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[5] Ben Y. Zhao and Anthony D. Joseph. XSet: A High Performance XML Search Engine. http://www.cs.berkeley.edu/~ravenben/xset, 2000.

[6] Angela Bonifati and Stefano Ceri. Comparative Analysis of Five XML Query Languages. *SIGMOD Record*, Vol. 29, No. 1, pp. 68–79, 2000.

[7] Eric A. Brewer. When Everything is Searchable. *Communications of the ACM*, Vol. 44, No. 3, pp. 53–55, march 2001. 安藤進翻訳. すべてがサーチ可能になるとき. IPSJ Magazine Vol42 No.12 pp.1231-1233 Dec.2001.

[8] Forbes J. Burkowski. An Algebra for Hierarchically Organized Text-dominated Databases. *Information Processing & Management*,

Vol. 28, No. 3, pp. 333–348, 1992.

[9] Taurai Tapiwa Chinenyanga and Nicholas Kushmerick. Expressive Retrieval from XML Documents. In *SIGIR'01:Proceedings of the 24th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pp. 163–171, sep 2001.

[10] Charlie L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. An Algebra for Structured Text Search and a Flamework for its Implementation. *The Computer Journal*, 1995.

[11] Charlie L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. Schema-Independent Retrieval from Heterogeneous Structured text. In *Proc. of the 4th Annual Symposium on Document Analysis and Information Retrieval*, pp. 279–290, Las Vegas, April 1995.

[12] Search Tools Consulting. Search Tools for Web Sites and Intranets. http://www.searchtools.com, 2001.

[13] Daniel Egnor and Robert Lord. Structured Information Retrieval using XML. In *Proc. of the ACM SIGIR 2000 Workshop On XML and Information Retrieval*, 2000.

[14] Daniela Florescu, Ioana Manolescu, and Donald Kossmann. Integrating Keyword Search into XML Query Processing. In *Nineth International World Wide Web Conference*, 2000.

[15] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. XML-QL : A Query Language for XML, Aug 1998. http://www.w3.org/TR/NOTE-xml-ql/.

[16] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. A Query Language for XML. *WWW8 / Computer Networks*, Vol. 31, No. 11-16,17, pp. 1155–1169, May 1999.

[17] Don Chamberlin, Jonathan Robie, and Daniela Florescu. Quilt: An XML Query Language for Heterogeneous Data Sources. In *Proc.*

of the Third International Workshop on the Web and Databases, WebDB2000, 2000.

[18] Dongwook Shin. XRS: XML Retrieval System. http://www.dlb2.nlm.nih.gov/~dwshin/xrs.htnml, 2000.

[19] Dongwook Shin, Hyuncheol Chang, and Honglan Jin. Bus: An Effective Indexing and Retrieval Scheme in Structured Documents. In *Proc. of Digital Libraries '98*, pp. 235–243, 1998.

[20] Dublin Core. Dublin Core metadata. http://dublincore.org/.

[21] Evangelos Kotsakis and Klemens Böhm. XML Schema Directory: A Data Structure for XML Data Processing. In *Proc. of The First International Conference on Web Information Systems Engineering, (WISE2000)*, pp. 56–63, 2000.

[22] Roy Goldman and Jennifer Widom. Approximate DataGuides. In *Proc. of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats*, Israel, January 1999. http://www-db.stanford.edu/pub/papers/adg.ps.

[23] D. Grossman, D. Holmes, O. Frieder, and D. Roberts. Integrating Structured Data and Text: A Relational Approach. *American Society of Information Science*, 1997.

[24] Kenji Hatano, Hiroko Kinutani, Masatoshi Yoshikawa, and Shunsuke Uemura. Extraction of Partial XML Documents using IR-based Structure and Contents Analysis. In *Proceedings of International Workshop on Data Semantics in Web Information Systems (DASWIS-2001)*, Nov 2001.

[25] ISO. ISO 8879: 1986. *Information Processing – Text and Office System – Standard Generalized Markup Language (SGML)*, Oct. 15 1986.

[26] JIS X 4151:1992 文書記述言語 SGML (Standard Generalized Markup Language), 日本規格協会, 1992.

[27] JIS X 4151:1998 文書記述言語 SGML (Standard Generalized Markup Language)(追補１), 日本規格協会, 1998.

[28] TR X 0008:1998 拡張可能なマーク付け言語 XML (eXtensible Markup Language), 日本規格協会, 1998.

[29] Marcin Kaszkiel and Justin Zobel. Passage Retrieval Revisited. In *SIGIR'97: Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 27-31, 1997, Philadelphia, PA, USA*, pp. 178–185. ACM, 1997.

[30] Haward Katz. XML Query Engine. http://www.fatdog.com, 2000.

[31] P. Kilpel. Tree Matching Problems with Applications to Structured text databases, 1992.

[32] Hiroko Kinutani, Masatoshi Yoshikawa, and Shunsuke Uemura. Identifying Result Subdocuments of XML Search Conditions. In *Proc. of 2000 Kyoto International Conference on Digital Libraries: Research and Practice*, pp. 232–239, Kyoto, Japan, November 2000. Kyoto University.

[33] Karl Aberer Marc Voltz and Klemens Böhm. Applying a Flecible OODBMS–IRS–Coupling to Structured Document Handling. In *Proc. of the* $2o^{th}$ *International Conference on Data Engineering*, pp. 10–19, 1996.

[34] Jeffery Naughton, David DeWitt, and David Maier et al. The Niagara Internet Query System. Technical Report, University of Wisconsin–Madison, 2000.

[35] Gonzalo Navarro and Ricardo Baeza-Yates. A Language for Queries on Structure and Contents of Texual Databases. In *ACM SIGIR '95*, pp. 93–101, July 1995.

[36] Gonzalo Navarro and Ricardo A. Baeza-Yates. Proximal Nodes: A

Model to Query Document Databases by Content and Structure. *Information Systems*, Vol. 15, No. 4, pp. 400–435, 1997.

[37] Norbert Fuhr and Kai Grossjohann. XIRQL An Extension of XQL for Information Retrieval. In *Proc. of the ACM SIGIR 2000 Workshop On XML and Information Retrieval*, Athens, July 2000.

[38] Norbert Fuhr and Kai Grossjohann. XIRQL: A Query Language for Information Retrieval in XML Documents. In *SIGIR'01:Proceedings of the 24th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pp. 172–180, sep 2001.

[39] Jonathan Robie. XQL (XML Query Language). http://metalab.unc.edu/xql/xql-proposal.xml, August 1999.

[40] Jonathan Robie, Joe Lapp, and David Schach. XML Query Language (XQL), Sep 1998. http://www.w3.org/TandS/QL/QL98/pp/xql.html.

[41] Jonathan Robie, Joe Lapp, and David Schach. XML Query Language (XQL). http://www.w3.org/TandS/QL/QL98/pp/xql.html, September 1998.

[42] Roy Goldman and Jennifer Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld, editors, *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, pp. 436–445. Morgan Kaufmann, 1997.

[43] Gerard Salton, James Allan, and Chris Buckley. Approaches to Passage Retrieval in Full Text Infoformation systems. In Robert Korfhage, Edie M. Rasmussen, and Peter Willett, editors, *SI-*

GIR'93:Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, pp. 49–58. ACM, 1993.

[44] Torsten Schlieder. Approxql: Design and Implementation of an Approximate Pattern Matching Language for XML. Technical report, Freie Universität Berlin, May 2001.

[45] Torsten Schlieder and Felix Naumann. Approximate Tree Embedding for Querying XML Data. In *Proc. of the ACM SIGIR 2000 Workshop On XML and Information Retrieval*, 2000.

[46] Sung Hyon Myaeng, Dong-Hyun Jang, Mun-Seok Kim, and Zong-Cheol Zhoo. A Flexible Model for Retrieval of SGMLDocuments. In *SIGIR'01:Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 138–145, August 1998.

[47] Jens E. Wolff, Holger Florke, and Armin B. Cremers. Searching and Browsing Collections of Structural Information. In *Advances in Digital Libraries*, pp. 141–150, 2000.

[48] World Wide Web Consortium. XML Schema Part 0: Primer. http://www.w3.org/TR/xmlschema-0. W3C Recommendation, 2 May 2001.

[49] World Wide Web Consortium. XML Schema Part 1: Structures. http://www.w3.org/TR/xmlschema-1. W3C Recommendation, 2 May 2001.

[50] World Wide Web Consortium. XML Schema Part 2: Datatypes. http://www.w3.org/TR/xmlschema-2. W3C Recommendation, 2 May 2001.

[51] World Wide Web Consortium. Extensible Markup Language (XML) 1.0. http://www.w3.org/TR/1998/REC-xml-19980210, February

1998. W3C Recommendation 10-February-1998.

[52] World Wide Web Consortium. Web Naming and Addressing Overview (URIs, URLs, ...), June 1998. http://www.w3.org/Addressing/Addressing.html.

[53] World Wide Web Consortium. Namespaces in XML. http://www.w3.org/TR/1999/REC-xml-names-19990114/, January 1999. W3C Recommendation 14-January-1999.

[54] World Wide Web Consortium. Resource Description Framework (RDF) Model and Syntax Specification. http://www.w3.org/TR/REC-rdf-syntax/, February 1999. W3C Recommendation 22-February-1999.

[55] World Wide Web Consortium. XML Path Language (XPath) version 1.0. http://www.w3.org/TR/xpath, November 1999. W3C Recommendation 16 November 1999.

[56] World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Second Edition),. http://www.w3.org/TR/2000/REC-xml-20001006, October 2000. W3C Recommendation 6-October-2000.

[57] World Wide Web Consortium. XQuery 1.0 and XPath 2.0 Functions and Operators. http://www.w3.org/TR/2001/WD-xquery-operators-20011220, December 2001. W3C Working Draft 20 December 2001.

[58] World Wide Web Consortium. Xquery: A Query Language for XML. http://www.w3.org/TR/2001/WD-xquery-20010607, June 2001. W3C Working Draft 07 June 2001.

[59] Yoshihiko Hayashi, Junji Tomita, and Gen'ichiro Kikui. Searching Text-rich XML Documents with Relevance Ranking. In *Proc. of the ACM SIGIR 2000 Workshop On XML and Information Retrieval*,

Athens, July 2000.

[60] Masatoshi Yoshikawa, Hiroko Kinutani, Yohei Yamamoto, Hiroyuki Kato, and Shunsuke Uemura. Identification of Query Result Subdocuments and Reverse Indices in XML Databases. In Yoshifumi Masunaga and Stefano Spaccapietra, editors, *In Advances in Databases and Multimedia for the New Century - A Swiss/Japanese Perspective*, Vol. Vol. 10 of *Advanced Database Research and Development Series*, April 2000.

[61] 絹谷弘子, 吉川正俊, 植村俊亮. スキーマのない多様な XML 文書のリポジトリに対する問合せ処理について. 情報処理学会データベースシステム，情報学基礎合同研究会研究報告, No. 2000-DBS-121-19 / 2000-FI-58-19, may 2000.

[62] 波多野賢治, 絹谷弘子, 吉川正俊, 植村俊亮. 情報検索技術による構造化文書の抽出法. 情報処理学会データベースシステム，電子情報通信学会データ工学合同研究会研究報告, 2001.

[63] 絹谷弘子, 波多野賢治, 吉川正俊, 植村俊亮. XML 文書の文書構造と内容を用いた部分文書の抽出手法. 情報処理学会論文誌: データベース, 2002.

付録

A　根付き木の定義

XML 文書の論理構造は，根付き木である．本節では，本研究でモデル化に利用している根付き木とその部分木の基本的な記法について文献 [31] の定義に基づいて述べる．

A.1　基本的な用語

用語 1 (二項関係の推移閉包など) D を集合，R を $D \times D$ の二項関係とする．R の推移閉包を R^+, R の反射的推移閉包を R^* を次のように定義する：

$$\begin{aligned} R^0 &= \{(x,x)|x \in D\}, \\ R^{n+1} &= \{(x,y)|\exists z \in D : (x,z) \in R, (z,y) \in R^n\}, n \geq 0, \\ R^+ &= \bigcup_{n>0} R^n, \\ R^* &= \bigcup_{n\geq 0} R^n \quad = R^0 \cup R^+. \end{aligned}$$

用語 2 (根付き木 $T = (V, E, root(T))$) T を木，V をノードの有限集合，$root(T) \in V$ を木 T の根とし，V 上の二項関係 E が次の条件を満たす時，T を根付き木という．根付き木は， $T = (V, E, root(T))$ で表す：

$(u, v) \in E$ の時，(u,v) を親ノード u からその子ノード v への枝とすると，

1. 根ノードは親ノードを持たない．
2. 根以外の木のノードは，たった一つの親ノードを持つ．
3. すべてのノードは，根から枝を通って到達可能である．すなわち，すべての V 中のノード v に対して，$(root(T), v) \in E^*$.

用語 3 (ラベル，経路) 木中のノードからあるアルファベットへの関数 $label$ が与えられると，その木は，ラベル付けされていると呼ぶ．

また，枝の並び $(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)$ をノード v_1 から v_2 の経路と呼び，その長さは $n-1$ である．

木の中には，根ノードから各ノードへは，それぞれ唯一の経路が存在する．

ノード v_1 から v_2 の経路を各ノードに対応した関数 $label$ の値を根ノードに近い方からスラッシュ'/'で区切って並べたものを経路式と呼ぶ．すなわち，

$$label(v_1)/label(v_2)/\ldots/label(v_n)$$

と表す．また，木 T の根ノード $root(T)$ のラベルは，'/' とする．

用語 4 (親子関係) 木の各ノード $u \in V$ に対する祖先ノード集合，親ノード，子どもノード集合，子孫ノード集合を次のように定義する：

1. 祖先ノード集合 $anc(u) = \{v \in V | (v, u) \in E^+\}$
   i 番名の祖先ノード集合 $A^i(u) = \{v \in V | (u, v) \in E^i\}$ である．この集合は，要素（ノード）が一つであるか空集合である．
2. 親ノード集合は， $parent(u) = A^1(u)$ と表せる．
3. 子孫ノード集合 $desc(u) = \{v \in V | (u, v) \in E^+\}$
4. 子どもノード集合 $children(u) = \{v \in V | (u, v) \in E\}$
   ここで子どもとは，共通の親を持つノードである．
5. 子どもを持たないノードを葉ノードという．

用語 5 (順序付き木) 根付き木の中間ノードにおいて k 個の子どもノードを持ち，1 から k までの番号が付けられ，識別できる時，この木を順序付き木という．順序付き木のノード $u \in V$ の i 番名の子どもを $child(u, i)$ とする．ノード u が k 個 $(k > 0)$ の子どもを持つ場合，$child(u, 1)$ を u の長子ノード，$child(u, k)$ を末子ノードと呼ぶ．

用語 6 (部分木) 木を $T = (V, E, root(T))$ とする．ノード $u \in V$ を根とする T の部分木 $T[u] = (V', E', u)$ では，

$$V' = \{u\} \cup desc(u), \qquad E' = E \cap (V' \times V')$$

である．

木 T がラベル付で順序付きの場合，T の部分木のノードラベルとノード順は T と同じである．

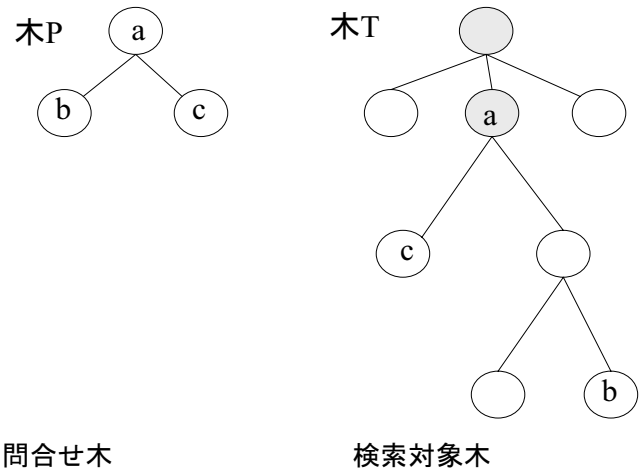

図 A.1 P ： T の順序なし包含された木

用語 7 (森) 森は，木の順序のある並びで，それぞれのノード間に共通のノードがないものとする．木 $T_1, T_2, \ldots, T_k$ に対する森 $F = \langle T_1, T_2, \ldots, T_k \rangle$, 木 $T_i \in F$ 中のノード u において，u を根とする部分木 $F[u] = T_i[u]$ と定義する．

また，通常単一の木 T と森 $\langle T \rangle$ を区別しない．

A.2 順序なし木包含問題

Kilpeläinen は，木の内部関係における 10 の問題について各問題の複雑さを分析している [31]．その中の順序なし木包含関係について述べる．本研究における定義 4 で定義した XML 部分文書検索はこの順序なし木包含問題であり，NP 完全であることを示す．

まず，Kilpeläinen の順序なし木包含問題について述べる．

用語 8 (木包含問題) $P = (V, E, root(P))$ と $T = (W, F, root(T))$ を木とする．P のノードから T のノードへの関数がラベルと祖先関係を保持しているとき，この関数を f が P から T 中への埋め込み *(embedding)* という．この必要条件は，

1. $u = v \iff f(u) = f(v)$,

2. label(u) = label(f(u)),

3. P において u が v の祖先ノードである場合に限り，T 中において $f(u)$ が $f(v)$ の祖先である．

T 中に P の埋め込みがある場合，この P を T の順序なし包含された木と呼ぶ．図 *A.1*が埋め込みの例である．

P が T の順序なし包含された木であると，$root(T)$ の祖先ノードを根とするすべての木において P は T の順序なし包含された木である．従って，P を含む最小の木を考察すればよい．

この P を含む最小の木を求めることが順序なし木包含問題 (UTI) である．

問題 1 (順序なし木包含問題：UTI) 問合せパターン木 P と検索対象木 T を与え，P を含む T 中の最小の部分木を求めること．

定理 1 順序なし木包含問題は，*NP–*完全問題である *([31], p32)*.

しかし，Kilpeläinen はこの問題を解くためにパターンノードの部分集合で構成するシステムで処理する場合，最悪でもパターンの大きさ m の指数の大きさである．従って，m が小さい場合には問題にはならないこと．また前処理でパターンから表を作る方法とこの表によって検索対象のノード数の線形倍で処理ができることを示している．

以上から，本研究における部分文書検索においても問合せ条件が少ない場合には，線形時間で検索可能と考えられる．

B　文脈ノード アルゴリズム

XML 文書を読み込み，DOM 木を内部で作成した後の，木中のカレントノード n に対し，文脈ノード context_node を求めるアルゴリズムを記述する．

入力：　　ノード型 `n`
出力：　　ノード型 `context_node`

メソッド：

```
 if  { nがテキストノード  } then
       do    { ノード型 n1 ← nの親ノード;  };
       else  { ノード型n1 ← n ;  };
ノード型 p ← n1の親ノード;
```

```
if  { pが文書ノード  } then
      do   { context_node ← n1 ;        return context_node;  };
pp ← p の親ノード ;

do  {
      if  { pp が 文書ノード  } then
            do   { context_node ← p ;        return context_node;  };
      int count ← 0;  //同名の兄弟ノード数

      for ( int i = 1 to ppの子どもノード数 )
        if  { 子どもノード ch(i) が p と同じノード名  } then  count++;
        if  { ch(i) が n1と同じノード名  } then
              do   { context_node ← p ;       return context_node;  };
      end for;

      if  { count > 1  } then
            do  { context_node ← p;       return context_node;  };
      p ← pp;
      if  { pが文書ノード  } then
            do  { context_node ← p;       return context_node;  };
      pp ← p の親ノード ;
   } while  { ppが文書ノードでない限り  };

context_node ← p;
return context_node;
```

研究業績

学術論文

1. Masatoshi Yoshikawa, Hiroyuki Kato and Hiroko Kinutani: “Design Framework of a Database for Structured Documents with Object Links”, IEICE Transactions on Information Systems, Vol. E82-D, No. 1, pp. 147-155, January 1999.
2. 絹谷 弘子, 波多野 賢治, 吉川 正俊, 植村 俊亮: “XML 文書の文書構造と内容を用いた部分文書の抽出手法”, 情報処理学会論文誌: データベース,2002.(採録決定)

著書

1. Masatoshi Yoshikawa, Hiroyuki Kato, Hiroko Kinutani and Masahiro Watanabe: “The ParaDocs Document Database System and Visual User Interface for Information Retrieval”, Advanced Database Systems for Integration of Media and User Environments ’98, Advanced Database Research and Development Series – VOL.9 pp. 81-86, (Editors: Yahiko Kambayashi, Akifumi Makinouchi, Shunsuke Uemura, Katsumi Tanaka and Yoshifumi Masunaga), ISBN 981-02-3436-8, World Scientific, Singapore, 1998.（分担執筆）
2. Masatoshi Yoshikawa, Hiroko Kinutani, Yohei Yamamoto, Hiroyuki Kato and Shunsuke Uemura: “Identification of Query Result Subdocuments and Reverse Indices in XML Databases”, in Advances in Databases and Multimedia for the New Century - A Swiss/Japanese Perspective - (Editors: Yoshifumi Masunaga and Stefano Spaccapietra), Advanced Database Research and Development Series - Vol. 10, World Scientific Publishing, ISBN 981-02-4310-3, April 2000.（分担執筆）

国際会議発表

1. Hirotaka Kanemoto, Hiroyuki Kato, Hiroko Kinutani and Masatoshi Yoshikawa: “An Efficiently Updatable Index Scheme for Structured Documents”, Proc. of 9th International Workshop on Database and Expert Systems Applications (DEXA’98), IEEE Computer Society, pp. 991-996, August 24-28, 1998.
2. Hiroko Kinutani, Masatoshi Yoshikawa, Yohei Yamamoto, Takahiro Morimoto and

Shunsuke Umeura: “On Contextual Queries for XML Documents Using Namespaces”, Proc. of International Symposium on Digital Libraries 1999 (ISDL’99), pp.151-154, University of Library and Information Science, Tsukuba, Ibaraki, Japan, September 28-29, 1999.
3. Hiroko Kinutani, Masatoshi Yoshikawa and Shunsuke Uemura: ”Identifying Result Subdocuments of XML Search Conditions”, Proc. of 2000 Kyoto International Conference on Digital Libraries: Research and Practice, pp. 232-239, Kyoto University, Kyoto Japan, November 13-16, 2000.
4. Masatoshi Yoshikawa, Toshiyuki Amagasa, Dao Dinh Kha, Kenji Hatano, Hiroko Kinutani, Noboru Matoba, Junko Tanoue, Masahiro Watanabe and Shunsuke Uemura: “On Two Query Interfaces for Genome XML Databases”, IEEE Workshop on XML-Enabled Wide Area Search in Bioinformatics (XEWA), League City, Texas, December 13-14, 2000.
5. Kenji Hatano, Hiroko Kinutani, Masatoshi Yoshikawa and Shunsuke Uemura: “Extraction of Partial XML Documents Using IR-based Structure and Contents Analysis”, Proc. of International Workshop on Data Semantics in Web Information Systems (DASWIS-2001), In conjunction with 20th International Conference on Conceptual Modeling (ER2001), Yokohama, Japan, November 27-30, 2001.

国内発表

1. 絹谷 弘子，吉川 正俊: ”オブジェクトと文書内文字列間の相互参照機構を有するデータベースシステム Paradocs のアーキテクチャ”, デジタル・ドキュメント・シンポジウム ’98, 情報処理学会シンポジウムシリーズ Vol. 98, No.3, pp.37-46, 1998 年 1 月.
2. 金本 博隆, 加藤 弘之, 絹谷 弘子, 吉川 正俊: ”効率的な更新が可能な構造化文書の索引”, 情報処理学会研究報告, 98-DBS-114, 1998 年 1 月.
3. 絹谷 弘子, 加藤 弘之, 吉川 正俊: ”オブジェクトリンクを有する構造化文書を管理するデータベースシステム Paradocs — アーキテクチャ —”, 電子情報通信学会技術研究報告, DE98-5, pp. 31-38, 1998 年 5 月.
4. 山本 陽平，絹谷 弘子，吉川正俊，植村俊亮：“ XML 文書のための逆経路索引とその応用 ”， 第 3 回 XML 開発者の日, 2000.
5. 絹谷弘子，吉川正俊，植村俊亮: “スキーマのない多様な XML 文書のリポジトリに対する問合せ処理について”, 情報処理学会データベースシステム，情報学基礎合同研究会研究報告, 2000-DBS-121-19 / 2000-FI-58-19, 2000 年 5 月 26 日.
6. 波多野 賢治，絹谷 弘子，吉川 正俊，植村 俊亮: ”情報検索技術による構造化文書の抽出法”, 情報処理学会データベースシステム， 電子情報通信学会データ工学合同研究会研究報告, Vol.2001, No.71, 2001-DBS-125(II)-83, pp.137-144 / Vol.101, No.193, DE2001-96, pp.135-142, 2001 年 7 月 19 日.