デジタル・ドキュメント 27－3
（２００１．３．１４）

大規模 XML 文書の検索と格納技術の開発

難波 功　井形 伸之　小櫻 文彦　山根 康男
{namba, igata,kozac, yamane}@jp.fujitsu.com
富士通研究所
〒 211-8588 川崎市中原区上小田中 4-1-1

本稿では、大量の構造化文書に対する検索要求を処理する方式として、構造検索に対応した全文検索機能並びに木構造の形で格納した XML 文書の部分取得機能を用いる手法について提案する。本手法では、文書の構造とテキストに対する 2 種類のインデックス、並びに XML を木構造で格納する格納部を用いる。 本方式の処理速度は文書件数よりは対象文書の複雑さに依存する。最良の場合には従来の項目検索と同等の性能となるため通常の文書検索との親和性が高い。最悪の場合には検索項目 1 つが文書に含まれるパスの数だけ展開されるため、構造を指定した検索に対する性能は質問文次第となる。実験では、1 文書が 30 程度のタグを含む複雑な XML 文書 200MB に対して、文書のルート以下に単語が含まれるという性能上最悪となる検索式を投入しても秒間 17 質問文の処理が可能という実用的な性能が得られた。

A Method of Indexing and Storing for Large Scale XML Document

Isao Namba　Nobuyuki Igata　Fumihiko Kozakura　Yasuo Yamane

Fujitsu Laboratories Ltd.
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki, Kanagawa, 211-8588 Japan

In this paper, we propose a high-speed query processing method for a large scale structured document set using a full text search index, and storage for XML document in tree structure. Our method uses two types of indexes for document structures and contents, and one storage for storing XML document in tree structure. The processing speed depends on the complexity of tree structure of target documents. In best case its performance is same as that of so-called field search. In worst case, the one term in hierarcical structure query is expanded into the terms of which number is the different path in target documents. In experiment we prepared XML documents which had 30 tags in average and whose volume was 200MB, and run the queries which were expected to be slowest in processing. The query processing performance was 17 query/second, and it is enough for practical use.

1 はじめに

XML(eXtensible Markup Language) は、データ流通の標準として、XMLを処理するアプリケーションに大きな期待が寄せられている。この期待に沿うように、データベースでは、多くのベンダーからXML対応 [1], [2] がなされ、XMLのデータベースへの格納技術に関する研究 ([3], [4] など) が行なわれている。これに対して、XML文書検索に関しては、「利用者にとって使いやすいしかも十分な性能を持つサーチエンジンを開発するためには、従来のデータベースと情報検索両方の要素技術が必要となり、多くの挑戦的な課題が存在する。」 [5] とあるように実用規模の文書での適用を前提においた研究そのものが少ないのが実情である。

現在のところ、商業システムでは、文書検索システムを構造化文書に対応させる場合には、タグをまったく無視して平テキストとして検索するか、従来の全文検索システムの項目検索機能を利用しRDBのようにフラットな構造検索だけをサポートするか、あらかじめユーザーが登録した構造に対してのみ最大繰り返し回数などを規定したうえで構造検索を可能とするという方法が多いと見受けられる。

しかしながら、文書検索システムに構造検索機能を加えることは、 文書の構造を用いた高度な検索要求による検索精度の向上、構造化文書を処理するデータベースのテキスト検索性能の強化などの多くメリットが考えられる。 またゲノム情報データベースのように大量に流通する文書の中でも構造を意識した検索が有効そうなものが現れていること、情報抽出技術などを用いて [6]、平文に対して構造検索の適用が可能なタグをつけることが可能となっており、大規模文書に対してXMLの特性を生かした検索を行なえる環境が整いつつある。

そこで、本稿では、大規模XML文書に対する文書の構造指定を含めた検索要求を処理する検索エンジンの索引方式並びに格納方式について検討する。

2 対象文書と機能

まず。どのような文書に対してどのような機能が必要であるかを検討する。これは、複雑な構造文書に対する検索機能を実装してもそれに見合った文書が存在するのでなければ、そもそもその機能の使い道がないこと、主流となる用途に対しては文書規模が大きくなってもシステムが対応できることが必要となるからである。

2.1 項目検索で用が足りる文書

1つの文書が繰り返しを含まない項目から構成される、もしくは繰り返しや階層構造を含んでもそれを区別して検索する必要がない文書では、従来の項目検索機能 [7] をサポートしていれば、技術的な課題はない。

```
< 特許広報 >
< 公報種別 > 公開特許公報（A） </ 公報種別 >
< 公開番号 > 特開平１３－１３４２０５ </ 公開番号 >
< 公開日 > 平成８年（１９９６）５月７日 </ 公開日 >
< 発明の名称 > ブレーキ用パワーアンプ装置 </ 発明の名称 >
< 国際特許分類第６版 >H02P 15/00 H </ 国際特許分類第６版 >
:
< 抄録 >
< 目的 > 機種の統一による標準化とボリューム調整 ....
簡易巻径検出方 <BR> 式の張力制御装置を提供する。 </ 目的 >
< 構成 > 入力設定回路と電流フィードバック回路と ....
</ 抄録 >
< 特許請求の範囲 >
< 請求項 number="1"> クラッチ・ブレーキを...</ 請求項 >
</ 特許請求の範囲 >
< 発明の詳細な説明 > 本発明は .. </ 発明の詳細な説明 >
:
</ 特許広報 >
```

```
< 請求の範囲 > にクラッチがあり、< 発明の詳細な説明 >
に張力制御を含む文書の < 発明の名称 > の一覧を出力せよ。

SELECT 発明の名称 WHERE < 請求の範囲 > クラッチ </>
AND < 発明の詳細な説明 > 張力制御 </> [1]
```

図 1: 特許のXML化と質問の例

図1 は特許文書をXML化したものとそれに対する質問文の例である。現在大量に流通している文書、例えば帳票、図書カード、特許といったものをXML化した場合にはこの形をとるし、普通の文書をXML化してもこの範疇に属すると予想される。

2.2 構造検索が有効であるもの

1つの文書が繰り返し並びに階層構造を含み、かつその部分構造を区別する検索が有効であるというものである。 典型的な条件は、文書中の特定部分がグループとなりそれの繰り返しがあるというものである。蓄積量が多くこの機能が有効そうなデータには、ゲノムデータ (DTD の一例としては、GAME[8])、論文の参照部分の検索などがある。遺伝子情報データベースのうち、グループ化した検索要求が意味を持ちそうな部分には、属性 (FEATURES) の (Location/Qualifiers) 情報の部分がある。また論文データの検索でも参照部分を検索

する，例えば「富士通のXXが著者であるYYという論文を参照している論文のタイトル一覧が欲しい」という検索を行なう際には、参照項目がグループになっていることを意識する必要がある。構造指定を含んだ検索により検索システムの精度を上げるという観点からするとこのような構造を持ったデータが世に多く出回るとよいのであるが、それほど種類が多くないというのと、そのような検索要求自体が中心的でないというのが実情である。

対象文書としてはこのような自然発生的なもの以外に人為的に作った対象文書、例えば情報抽出技術[6]を用いてタグ付けをした文書がある。情報抽出によるタグ付けは従来の全文検索システムでは不可能であった検索を行なうためのものである。これは、検索結果として文書全体ではなく実際に検索がヒットしたタグで囲まれた領域を結果として返却したり、自然言語で入力された表現より構造化文書に対する問い合わせを生成し、全文検索よりノイズが少ない検索を実現するものである。[9] [10]では自然言語の質問からXMLの構造指定を含んだ検索式に変換する方式の検討を行なっている。また[11]ではサポートセンターの質問応答データに対してタグづけを行ない、検索を行なっている。図2は、情報抽出が適用された文書とそれに対する自然言語による質問文、生成されたXMLに対する問い合わせ例である。

```
<article>
<headline>
ＡＩＤが開発・販売、病院向けにソフト２種――看護婦の勤務表作成など
</headline>
<body>
< 販売情報 >
< 連体修飾 > ソフトウエア < 販売付加行為 > 開発 </ 販売付加行為 > の
</ 連体修飾 >
< 組織名 > アドヴァンスト・インフォーメイション・デザイン </ 組織名 >
（略称 < 略語 > ＡＩＤ </ 略語 >
< 所在地 > 松本市 </ 所在地 >
< 氏名 > 津久井光男 </ 氏名 >
< 役職 > 社長 </ 役職 >
）は、
< 販売製品情報 >
< 製品情報 >
< 製品名連体修飾句 > 病院・医院を対象とした </ 製品名連体修飾句 >
「< 製品名 > 看護婦勤務割りシステム </ 製品名 >」と
「< 製品名 > 放射線科窓口管理システム </ 製品名 >」
</ 製品情報 >
</ 販売製品情報 > を開発、販売を始めた
</ 販売情報 >
このうち勤務割りシステムはパソコンで看護婦の勤務ローテーション表
を短時間に作るもので、手作業による作成のわずらわしさを解消してくれる
病院向けソフトを主力分野の金融機関向けシステムに次ぐ
柱に育てたい考えだ……
</body>
</article>
```

```
松本にある病院向けのシステムを開発している会社の商品は？

その下にある < 所在地 > が松本を含みかつ < 販売製品情報 > がシステムを含みかつ
< 販売製品情報 > が医療を含む < 販売情報 > を検索し、その下にある < 組織名 >
と < 製品名 > を出力せよ。

SELECT 販売情報 #i*/ 組織名, 販売情報 #i*/ 製品名
WHERE < 販売情報 #i/*>
< 所在地 > 松本 </ 所在地 >
< 販売製品情報 /*> システム </ 販売製品情報 >
< 販売製品情報 /*> 病院 </ 販売製品情報 >
</ 販売情報 >
```

図 2: 情報抽出データと問い合わせの例

3 システムの構成

前章で見たように、普通の文書をXML化しても検索要求の中心は従来の項目検索であることが予想される。そのため、大規模データを対象とした場合でも、項目検索対しては性能の低下がないことが望ましい。また構造検索が必要な文書でも検索要求の全てが構造検索を必要とするものではない。これからすると、項目検索に対する性能劣化が全くなくかつ構造指定の検索に対する性能劣化が少ないというのが現在のところ穏当なシステム構成であると考えられる。この構成は索引部分に関しては、旧来の構成に近くなるため、構造検索対応部分を追加すれば、既存資源を有効利用できる。

さて、従来の全文検索に用いられる inverted file [12]による索引が返す結果は文書のトップへのポインタ(以下文書idと呼ぶ)だけである。そのため、2.2章の例で用いられている木構造の部分取得機能を実現するためには、inverted file の検索後、文書idに対応した文書の実体部分を取り出す機能並びに格納部分が必要となる。また inverted file の索引は*文書*単位であるため、索引への登録時に登録単位を明確にする必要がある。

結局、本システム構成は、構造検索を高速に行なう索引で検索条件にあう文書集合を絞り込み、絞り込んだ文書集合に対して部分構造を取得するというものとなる。この構成は文書検索では検索結果全てに対して重たい実体の取得(全文やタイトルの取得)が行なわれないという通常の使い方とも親和性が高い。図3は、本システムの構成図であり、システムは大まかには索引部分、格納部分より構成される。以下の章では、本システムで採用した索引方式、格納方式、検索処理の流れ、言語系それぞれについて説明する。

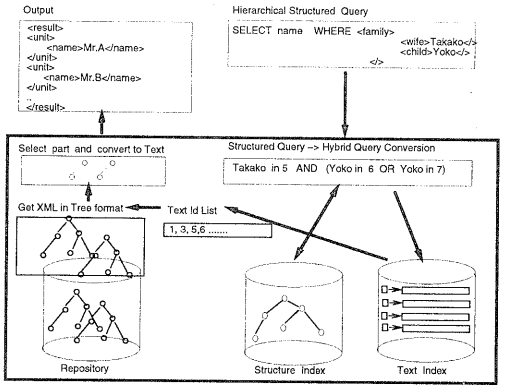

図 3: システム構成

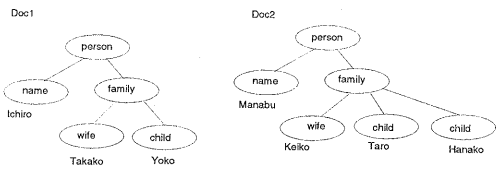

図 4: XML 文書の木構造表現

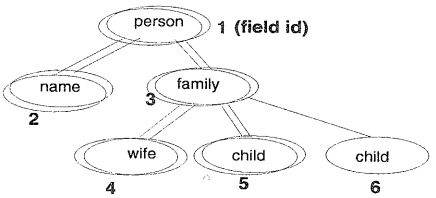

図 5: 構造インデックス

3.1 索引方式

文書検索で用いられる inverted file で実現可能な構造指定を含む文書検索の実現方法としては、大まかにわけると階層構造を演算で求める方式 (Overlapped List [13]) 並びに階層構造をインデックスで持つ方式 (Structure Index + Text Index) の 2 つがある。Structure Index + Text Index の方式は旧来の inverted file をテキストインデックスとし、その外部に XML 文書の木構造情報を保持する構造インデックス (Structure Index) を追加するものであり、従来のシステムに構造インデックスを追加するだけでよいため、既存資産の利用が可能であること、項目検索との親和性が高いという特徴があるため、本システムではこの構成を採用する。

本手法では、文書登録時に、構造インデックスには、各文書の根からの経路情報を全て保持し、その経路情報に field id を振る。(以下、根からの接点までを path と呼ぶ) そして、その field id と単語の対をキーとして、テキストインデックスに登録する。検索実行時には、構造指定を含む検索要求を構造インデックスを参照することによりテキストインデックスのみへのアクセスで処理が可能な検索式へと変換する。図 4は XML 文書を木構造で表したもの、図 5は構造インデックスの概念図、図 6はテキストインデックスの概念図、図 7は、構造を含む検索要求をテキストインデックスへの参照だけで処理可能な質問文に変換したものである。

このように変換された検索式は、一般には図 7の例が示すように、キーは同一だが field id が異なる項目を大量に含む OR 展開された式となる。これを利用して、以下の手順で inverted file へのアクセスを行ない処理の高速化を図る。

1. 文書 id list のアクセスの前に検索式中の全ての項目に対して辞書検索を行なう。その際に、項目をキーとし、データ部分に field id を持つ形で辞書項目をキャッシュする。検索キーの異なり数は展開前の数に依存するので、実際に辞書を引き、I/O が発生する回数はこの数に押えることができる。

2. 辞書項目が inverted file 中に存在しないもの並びにそれと AND 結合されている項目を検索式より除去する

3. 残った単語項目に対して演算を行ない、文書番号リストを得る。

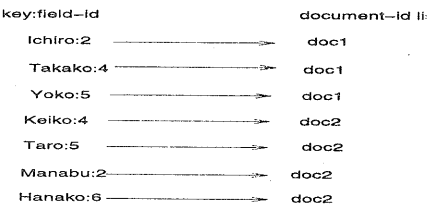

図 6: テキストインデックス

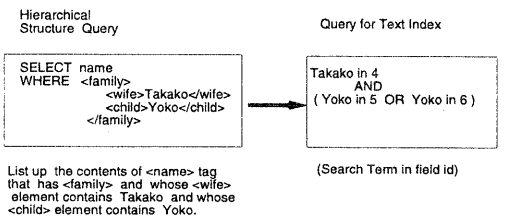

図 7: 質問文の変換

Structure Index + Text Index では、OR 展開される最大数は、全文書数に含まれる木の異なり path となる。検索性能からいうと最良となるのは、従来の項目検索のように OR 展開された結果が 1 となる場合、つまり項目が文書中で一意に決まる場合であり、最悪となるのは、展開数が最大の場合、つまり文書のルート以下の任意の場所に含まれる単語を探す場合である。この場合、文書の構造によっては 1 つの単語が数千件の項目に展開されることもある。ただし実際には、辞書引きの時間はキャッシュにより無視できること、機械的に OR 展開されても検索要求中の単語がそこに含まれることは少ないため、辞書引き後実際に重い演算が適用される部分は少ないこと、木の異なり path の数は文書数の増加にともない一定の部分で落ち着くという傾向がある。そのため、最悪の計算量は大きいが、普通の検索式を入れている限りは性能低下はそれほど大きくないという振舞いをする。

3.2 格納方式

索引部分が文書 id まで確定させるという条件下では、XML 文書の格納に関して考えるべき課題は以下の 3 つの項目に対する性能をどのようにバランスさせるかである。

1. 全文取得

   ヒットした文書の全体を検索結果として返す。

2. ユニークな部分取得

   文書に対して一意に決まる部分を検索結果として返却する。従来の全文検索での項目の出力、例えばタイトル一覧を返すといった場合に相当する。

3. 木探査を必要とする部分取得

   取得条件が部分取得であり、タグの上で該当箇所が複数ある場合には、XML の木をルートより探索し、検索条件にあったタグの部分だけ返却する必要がある。

さて、基本的な格納方式としては、以下の 3 通りが考えられる。

1. text 格納＋実行時にパーズする方法

   XML 文書を木構造のような形にして格納するのではなく、テキストのまま格納する。文書取得時に、パーズして部分取得を行なう。

2. 木構造格納＋固めて格納

   XML 文書を木構造にして、それを I/O の単位から見ると連続領域に格納する。

3. 木構造格納＋分割して格納

   XML 文書を木構造にして、それを I/O の単位から見ると連続しない領域に格納する。

方法 1は、XML パーザーが高速であれば、I/O の点で有利であると考えられる。しかし I/O キャッシュが効く環境下では、パーザー実行時間分、不利である。方法 3はユニークな部分取得以外は特に長所がない。 方法 2は記憶容量の観点で効率的に XML 文書を木構造に変換できかつ、全文取得の場合に木から文書の再構築が高速に処理できるのであれば、選択としては悪くはない。

以上の考察より、格納方式としては方式 2を採用した。これに加えて、木探査の高速化のため、文書ごとにその中に現れタグに対する索引をトップに埋め込み、記憶容量の節約のためにタグ名は外部の辞書で管理し格納された木構造中には含めない方式を採用した。[14] また格納庫としては、Object data base のベースとなる Wood[15] を使用した。

3.3 検索手順

以下に検索処理全体の手順を示す。

1. 構造指定を含む検索要求から、内部的なデータ形式である問い合わせ木を作成する

2. 構造インデックスを参照し、問い合わせ木をテキストインデックスに対する問い合わせに変換する。(Hybrid Query)[16]

3. テキストインデックスを参照し、Hybrid Query を真とする文書 ID の集合を得る。

4. 検索条件に合致する文書 ID 一覧を得る

5. 該当文書 ID より、木構造で格納された文書を取得する

6. 検索条件に該当する部分木構造を取り出す [14]

7. 木構造を XML テキストの形に変換して、検索結果を出力する

3.4 問い合わせ言語

問い合わせ言語の形としては、W3C の標準に従うのが望ましいと考えられるが、言語仕様がデータ指向 (data-oriented) であること、テキスト検索ではい拡張が必要であることなど問題がある。

そこで、言語仕様としては、XML-QL[17] のサブセットに拡張を加えたものとした。XML-QL をベースとしたのは、XQL[18] よりも複数文書に対する問い合わせを意識した言語仕様であること、従来の文書検索で用いられる「このフィールドにこの単語を含む文書のタイトルを出力せよ」といった論理演算を自然に記述できるといった理由による。またサブセットという意味は、SQL にあるような結果の bind のような機能は実装しないということである。また、拡張というのは、論理演算、単語の近接、ランキング検索のための単語の重み付けといった機能を加えたことである。文法は、SELECT によるタグの内容の取得と WHERE による照合条件からなる。詳細な文法記述はここでは略すが、図 8には、従来の項目検索、論理演算並びに構造指定の検索、検索で一致した部分の取得といった検索要求の記述例をあげる。

項目検索 <title> が「検索」を含む文書の <text> 以下を出力
SELECT text WHERE <title> 検索 </title>

構造指定 (AND) <ref> 直下の <title> が「検索」を含み、かつ <author> が Mr.X を含む文書の <text> 以下を出力
SELECT text WHERE <ref><title> 検索 </title><author>Mr.X</author></ref>

パスワイルドカード <ref> 以下にある <title> が「検索」を含み、かつ <author> が Mr.X を含む文書の <text> 以下を出力
SELECT text WHERE <ref/*><title> 検索 </title><author>Mr.X</author></ref>

一致場所取得 <ref> 以下にある <title> が「検索」を含み、<author> が Mr.X を含む文書のマッチした <ref> 以下を出力
SELECT ref/#i WHERE <ref/#i/*><title> 検索 </title><author>Mr.X</author></ref>

論理演算 <a> が「情報」を含み、 が 「検索」を含む文書の <text> 以下を出力
SELECT text WHERE <a> 情報 AND 情報

論理演算 (同一ノード) <a> が「情報」を含み、それと同じ <a> が「検索」を含む文書の <text> 以下を出力
SELECT text WHERE <a> 情報 & 情報

図 8: 構造指定を含む問い合わせの例

4 評価

4.1 測定項目

XML に対するシステムの性能評価の問題として、標準的なテストセットがないため他の実験と比較ができない、実性能は対象となる XML 文書、質問文の性格に左右されるという問題があるが、ここでは以下の 2 点に関して評価を行なう。

1. 検索、取得といった各機能の動作時間並びにサイズといった基本性能を測定する。

2. 文書の複雑さに応じた性能の変化

   Structured Index + Text Index の方式は文書構造の複雑さに依存して検索速度が変わる。そこで文書の複雑さを変えながら、ドキュメントのルート以下の任意のノードに単語が含まれるという最も厳しい質問文に対する処理時間を測定する。

4.2 実験環境

Ultra Sparc II 400MHz x 2CPU, 主記憶 1GB の計算機を使用した。また、文書登録時に必要となる XML parser としては、expat[19] を使用した。

検索対象として 1997 年の日経新聞 1 年分に対して、タグ付けを全く行なっていないもの (文書の開始終了のタグのみを含む) と図 2の形で情報抽出によるタグづけを行なったものと、それに対してタグを間引いたものを用意し、それぞれ低粒度データと高粒度データとした。表 1 に検索対象文書の基本データを示す。 低粒度、高粒度データは 1 文書あたりの平均 path 数にはさして違いはないが異なり path 数が 3 倍以上ある。 また、1 文書あたり平均パス数が 30 程度というのは、新聞の平均サイズ 1KB からすると約 30byte に 1 つタグがある計算となり、タグのつき方からするとかなり複雑な文書であるといえる。

表 1: 検索対象の基本データ

	タグなし	低粒度	高粒度
文書数	179,715	179,715	179,715
文書サイズ (MB)	230	340	345
異なりタグ数	1	24	99
異なり path 数	1	1963	6833
1 文書あたり平均 path 数	1	32.7	33.7

表 2: 構造インデックスサイズ

	タグなし	低粒度	高粒度
filed id 数 (path 数)	1	1,963	6,833
index size(KB)	0.3	61.3	213

表 3: テキストインデックスサイズ

	タグなし	低粒度	高粒度
登録キー数	316,038	2,399,209	2,657,714
辞書部サイズ (MB)	19	130	143
データ部サイズ (MB)	191	317	323
total(MB)	2.88	21.0	21.3

表 4: 格納部サイズ

	タグなし	低粒度	高粒度
total(MB)	501	1220	1224

表 5: 検索時間バッチ (単位: 秒)

	タグなし	低粒度	高粒度
Query 変換	0.02	0.38	0.41
辞書部参照	0.03	1.73	5.12
データ部参照&評価	0.08	0.81	3.13
取得	1.72(28)	2.18(73)	2.21(74)
木の組み立て	0.06	0.40	0.41

4.3 実験結果

表2, 表3に表1のデータに対するインデックスサイズ、表4に木構造で格納した XML 文書のサイズを示す。 インデックスの単位としては、形態素解析を掛けることにより取り出した単語としている。

表5は、 Structured Index + Text Index の方式にとって厳しい質問文 150 個に対する処理時間の合計である。 OR 展開を用いる方式にとって最も厳しいパターンは各文書のルートの下に単語が含まれる文書を探せというものであり、ここでの質問文は全て、「SELECT article WHERE <article/*> イルカ </>」のような形をとっている。これは、 内部的には図7のように「イルカ in 1 OR イルカ in 2 OR ... イルカ in N 」という長大な OR 式となり評価される。 文書取得は、 cold/hot の差が大きいので () の中に cold start の測定時間を含めた。

4.4 考察

表5にあるようにインデックスの検索時間は、文書の複雑さ、具体的には表2の異なりパスの数に比例する形で増大している。現在の実装は、既存のモジュールを利用したため、最初に長大な式に展開し、辞書引き、検索のループを回す形となっている。辞書検索と式の変換を同時に行ない、いらない項目を検索式中から最初に削り落すという処理を行なえば、ループ中の要素を減らすことが可能であり、性能向上の余地がある。

表5の実験では、インデックス部分の性能はタグなしのものが秒当たり 1249 質問文、低密度のものが 51 質問文、 高密度のものが 17 質問文である。サーチャーなどへのヒアリングによれば、使用する上で快適な応答速度は 1,2 秒である。 実験で用いた検索式が Structured Index + Text Index の方式にとって最もきついものであること、 Structured Index + Text Index 方式の速度性能が文書の件数よりは複雑さに依存すること、通常の構造化文書の複雑さが図1より低いと考えられることからすると、 GB 級の文書に対しても秒間 1 質問文以上の処理性能が得られていると考えられる。

表5に示すように個々の処理の中で実際に時間が掛かるのは取得部分となった。 取得部分は、 cold/hot start での差が大きく、 30 倍近くとなる。これはインデックス部分が cold start で初めてもそれなりの質問文数を流せば、 hot とさして性能が変わらないの [20] とは対象的である。 取得部分の性能を悪化させているのは、文書を木構造の形で格納した場合の膨れもある。表1, 表4 に見られるように木構造の形で文書を格納した場合のサイズの 1220MB はテキストの 340MB に対して 4 倍以上の差がある。 I/O を減らすことが平均的な使用状況での処理速度の向上に効果的と考えられるため、 expat のような高速な XML パーザー (600 文書 / 秒 @Ultrasparc 300MHz) を用い、格納部分の戦略として text 格納 +実行時にパーズするという方法は実験結果を見る限りは悪い方法とはいえないこととなる。

5 まとめ

本稿では、大量の構造化文書に対して高速に構造指定を含む検索を行なう手法を提案した。また実際の XML 文書を用いた実験を行い、大規模な構造化文書に対して最悪に近い検索式に対しても十分な速度性能が得られることを確認できた。その一方で I/O キャッシュが効きにくいことを考えると XML 文書の部分取得は大規模対応を行なう場合には、性能のボトルネックとなることも明らかになった。構

造化文書検索に対する要望としてあるのは、構造の参照条件をに完全に一致しなくても出力が得られるというものである。構造検索は図2 にあるように強力な絞り込みを可能とするが、条件が厳しすぎるため、結果が得られないことも多い。このような場合にどのように条件を緩和していくかといったことが、構造を利用した使いやすい検索の実現の今後の課題と考えられる。

参考文献

[1] http://www.oracle.com.

[2] http://www.excelon.com.

[3] J Shammugasundaram et al. Relational databases for querying xml documents: Limitations and opportunities. 25th Intl Conf on VLDB, pages 302–314, 1999.

[4] 吉川正俊 志村壮是 植村俊亮. オブジェクト関係データベースを用いたxml文書の格納と検索. In 情報処理学会論文誌, Vol.40,No.SIG6(TOD3), 1999.

[5] 吉川正俊. Xmlの問い合わせをめぐる最近の話題. 情報知識学会誌 Vol.10, No.3, pages 59–64, 2000.

[6] 西野文人 落谷亮 木田敦子 乾裕子 桑畑和佳子 橋本三奈子. トップダウンなパターン解析に基づく情報抽出. In 情処研報, NL124-13, pages 95–102, 1998.

[7] R. Baeze-Yates. An hybird query model for full text retrieval system. Technical Report DCC-1994-2, Dept. of ComputerScience, Univ. of Chile, 1994.

[8] http://www.bioxml.org/. Game. 2000.

[9] 伊吹潤 西野文人. 質問文からの検索条件と提示情報の決定. 言語処理学会 全国大会論集, 2001.

[10] 西野文人 伊吹潤. 情報ワーク支援システムのための情報抽出と構造化文書検索. 言語処理学会 全国大会論集, 2001.

[11] 柳瀬隆史 難波功 木田敦子 落谷亮. 談話構造を利用した質問応答事例の検索. 言語処理学会 全国大会論集, 2001.

[12] W.Frakes, R. Baeza-Yates, and editors. Information retrieval: Data structures and algorithms. Pretice-Hall, Endlewood Cliffs, New Jersery 07632, 1992.

[13] An Algebra for Structured Text Search and a Framework for its Implementation. C. clarke and g. cormack and f. burkowski. The Computer Journal, Vol.38, No.1, pages 43–56, 1995.

[14] 新田清 小櫻文彦 井形伸之 山根康男 難波功. 知識流通に適用可能なxml文書管理技術の開発. 人口知能基礎研究会, SIG-FAI-9904-13, 2000.

[15] Yasuo Yamane Nobuyuki Igata Isao Namba. High-performance xml storage/retrieval system. Fujitsu Scientific & Technical Journal Vol 36 No.2, pages 185–192, 2000.

[16] 井形伸之 難波功. 大規模な構造化データベースにおけるインデクシングと検索の手法. 情報学基礎 Vol.57, pages 9–16, 2000.

[17] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. Xml-ql: A query language for xml, submission to the w3c. http://www.w3.org/TR/NOTE-xml-ql/, 1998.

[18] J. Robie, J. Lapp, and D. Schach. Xml query language (xql). http://www.w3.org/TandS/QL/QL98/pp/, 1998.

[19] http://www.jclark.com.

[20] I Namba and N Igata. Fujitsu laboratories trec8 report. *The Eighth Text REtrieval Conference*, 2000.