XQuery 問い合わせ処理の最適化に関する研究
– 省メモリかつ高速な XQuery 処理系の提案–

2005MT001 秋山 啓　　2005MT057 久保田 雅司　　2005MT097 佐藤 文映
指導教員　野呂 昌満

1　はじめに

XML 文書に対する問い合わせ言語として XQuery[5] が提案されている．XML 文書の大規模化にともない XQuery の処理におけるメモリ使用量，処理時間も増加する．例えば XML 文書全体の木構造を構築する処理系の一つに SAXON[4] があるが，SAXON は木構築処理に XML 文書のサイズの約 4.5 倍のメモリを使用する．
本研究室では遅延パーサを用いた XQuery 処理系 [1] を提案した．遅延パーサは問い合わせから参照される部分のみ木作成処理をおこなうパーサであり，これを利用するとき SAXON と比較してメモリ使用量を 2/3 に削減，1.5 倍の高速化を実現した．木の構築に必要なるメモリ量は削減できたが，問い合わせでどのような木でも作成するために XML 文書の木構築に必要な親子兄弟関係などの情報を内部情報として探索対象の XML 文書全ノード分保存し，問い合わせで参照されない内部情報も保存するので無駄なメモリを消費する．
本研究は，遅延パーサを用いた処理系より省メモリかつ高速な処理系の実現を目的とする．遅延パーサを用いた処理系を改良し，内部情報を削減する方法を提案する．本処理系はあらかじめ質問式を解析しておき，XML 文書を解析する際に質問式の解析結果と XML 文書のタグの照合をおこなう．そこで一致したものを内部情報として保存する．質問式に子孫軸が含まれる場合はパスが省略されており，XML 文書との照合をすぐにおこなうことができない．スタックに一時的に保存しておくことで照合処理を遅らせ，問い合わせで参照される部分木の構築に使われる内部情報のみを保存する．
実験により遅延パーサを用いた既存の処理系と比較し，処理時間は約 1.1 倍高速化でき，メモリ使用量は約 1/10 に削減できたことを確認した．また，内部情報のさらなる削減についても考察した．

2　遅延パーサを用いた既存の XQuery 処理系

2.1 節では XQuery について，2.2 節からは遅延パーサを用いた既存の XQuery 処理系の概略と問題点を述べる．

2.1　XQuery

XQuery とは W3C によって提案された問い合わせ言語である．XQuery の質問式は XPath 式と FLWR 式から構成される．XPath 式は XML 文書の木構造を利用し，’/’ で区切ってノード階層を表現する．’//’(子孫軸) は全ての子孫ノードを表し，’ *’(ワイルドカード) は全ての子ノードを表す．例えば，/A//B はルート要素 A の子孫に 0 回以上要素が現れ，要素 B が出現することを表す．また，/A/*/B はルート要素 A に子要素が一つ存在し，要素 B が出現することを表す．

2.2　概略

本研究室は遅延パーサを用いた XQuery 処理系 (以下，既存の処理系) を提案した．遅延パーサの処理過程を図 1 に示す．

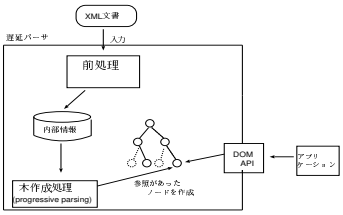

図 1　遅延パーサの処理過程

遅延パーサ [3] は XML 文書の木作成処理を遅らせ，問い合わせで参照される部分のみの木 (以下，探索部分木) を構築するパーサである．遅延パーサの処理は前処理と木作成処理で構成される．前処理では探索部分木の構築に必要な情報 (以下，内部情報) を保存する．内部情報とはノードの種類，親子兄弟関係を示す情報，テキスト表現である．木作成処理でどのような木でも構築するために XML 文書の全ノードの情報を保存する．その後問い合わせで参照があったノードは内部情報をもとに木を構築し，探索部分木を構築する．
既存の処理系は遅延パーサを用いることで探索部分木の構築を実現した．SAXON と比較して省メモリ化，高速化を実現した．

2.3　問題点

既存の処理系は SAXON と比較して木の構築量を削減することでメモリ使用量，処理時間を削減した．しかし，探索部分木の構築に必要な内部情報だけでなく全ノードの情報を内部情報として保存する．すなわち，探索部分木の構築に必要のない内部情報まで保存する．探索対象の XML 文書のサイズの 2 倍〜3 倍のメモリが処理に使用され，問題点として挙げられる．
探索部分木の構築に必要ない内部情報を削減するには次の 2 点の問題点がある．

- XML 文書を読み込む前に質問式の情報を得る必要がある．
- 子孫軸を含む質問式は XML 文書を読み込む前にパスを展開できず問い合わせに参照される可能性がある部分を全て内部情報として保存してしまう．

3 内部情報を削減した遅延パーサを用いた XQuery 処理系の提案

3.1 概略

我々は既存の処理系を改良し，探索部分木の構築に必要な内部情報のみを保存する処理系を提案する．2.3 節に示した問題を解決するためにあらかじめ質問式を解析し，問い合わせ木を作成する．XML 文書を解析する際に問い合わせ木と XML 文書のタグの照合をおこない，一致したものを内部情報として保存する．子孫軸を含む質問式はパスが省略されており保存するノードの特定が照合処理時ではできないので，スタックにノードの情報を一時保存して照合処理を遅らせる．

3.2 問い合わせ木

問い合わせ木は質問式に現れるパスを示したものである．この木と XML 文書のノードを照合して探索部分木の構築に使われる内部情報のみを保存する．
問い合わせ木は次の手順で構築される．

1. 質問式の for 節，where 節，return 節に現れるパスを木構造で表す．パスに含まれる XML 文書の要素，子孫軸，ワイルドカード，DES をノードとして表す．
   - DES ノードは DESCENDANT ノードの略で，where 節，return 節の葉につける．DES ノードはパスで指定したノードの子孫ノードを取得するために用いる．where 節または return 節の葉の子孫ノードを必要としない場合，その節の葉には DES ノードをつけない．
2. 1 で構築した三つの木構造に現れる共通の要素のノードを一つにまとめる．
3. 2 で構築した木の根にルートノードを結合する．ルートノードは探索部分木の開始点を表す特別なノードである．

問い合わせ木の例を図 2 に示す．

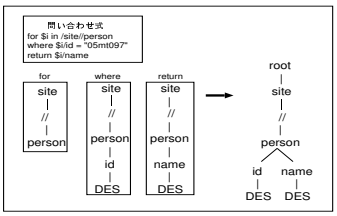

図 2　問い合わせ木の例

特に子孫軸を含む質問式はパスの経路が省略されており，保存するノードの特定が照合処理時ではできない．問い合わせ木と XML 文書のノードの照合を遅らせるためにスタックを用いてノードの情報を一時的に保存する．

3.3 内部情報保存処理

本処理系は探索部分木の構築に必要なノードの情報のみを内部情報として保存する．スタックにノードの情報を一時保存することでこれを実現する．XML 文書を解析しながらタグの種類と問い合わせ木のカレントノードの種類に合わせて内部情報保存処理をおこなう．スタックを用いた内部情報保存処理のアルゴリズムを次に示す．
XML 文書をタグとテキストの列として扱い，問い合わせ木のカレントノードの子ノードと XML 文書のタグ名を比較する．XML 文書の解析が終了するまで次の処理を繰り返す．

```
NodeList = XML 文書のノードのリスト
qtNode = 問い合わせ木のルートノード
Node = NodeList の次のノード
while(Node!=null){
  if(Node==開始タグ){
    if(qtNode の子ノード==DES ノード)
      Node の情報を内部情報として保存する
    else if(qtNode の子ノード==ワイルドカード
            ||qtNode の子ノード==子孫軸){
      Node の情報をスタックに push する
      qtNode = qtNode の子ノード }
    else if(qtNode==ワイルドカード
            ||qtNode==子孫軸)
      Node の情報をスタックに push する
    else{
      if(qtNode の子ノード==開始タグ名){
        Node の情報をスタックに push する
        qtNode = qtNode の子ノード }
      else
        開始タグの対となる終了タグが
        検出されるまで何も処理しない
    }
  }
  else if(Node==終了タグ){
   if(qtNode==子孫軸
      &&qtNode の親ノード==終了タグ名){
     qtNode = qtNode の親の親ノード
     スタックから pop する
     if(pop した Node
        ==問い合わせに必要な情報)
       Node の情報を内部情報として保存する
     else Node の情報を破棄する
  }
  else{ //qtNode が子孫軸以外
    if(qtNode の子ノード==DES ノード){
      if(qtNode==終了タグ名)
        qtNode = qtNode の親ノード }
    else{
      qtNode = qtNode の親ノード
      スタックから pop する
```

```
        if(pop したNode
            ==問い合わせに必要な情報)
          Node の情報を内部情報として保存する
        else Node の情報を破棄する
      }
    }
}
```

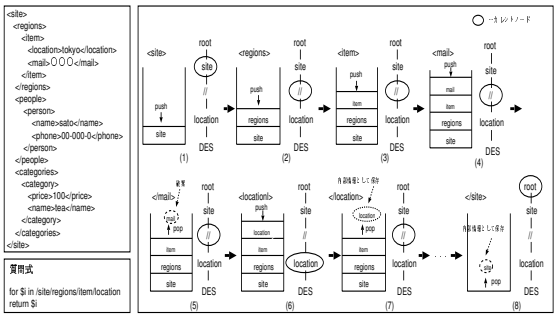

図 3　遅延パーサを用いた処理系の概略

内部情報保存処理の例を図 3 を用いて説明する．

1. 問い合わせ木のカレントノードを root とする．
2. スタックに一時保存として site を push する (1).
3. 2 の処理を<regions><item><mail>の検出時にも行う (2)(3)(4).
4. 終了タグ</mail>検出時，mail を pop する (5). mail は問い合わせに必要でない．よって内部情報として保存しない.
5. 3，4 の処理を繰り返す．開始タグ<location>の検出時，カレントノードの子ノードと location が一致しているので，location を push する (6).
6. 終了タグ</location>検出時, location を pop する (7)．問い合わせ木のカレントノードの子ノードが DES なので，問い合わせに必要な情報と判断できる．よって内部情報として保存する．スタックに積まれた情報も探索部分木の構築に必要なので，内部情報として保存する (8).

4　実験

本研究は既存の処理系を改良して内部情報を削減する．その削減量を評価するために実験をおこなう．メモリ使用量, 処理時間について SAXON, 既存の処理系と比較する．実験には PC(CPU Celeron M 370 1.5GHz, メモリ 1.256GB, OS Vine Linux 3.1, カーネル 2.4.27-0v17.3, Java コンパイラ 1.5.0_11) を用いた．また，XQuery 処理系を評価する標準的なベンチマークである XMark を用いた．XML 文書は XMark の 100MB のものを用い，質問式は XMark の質問式 20 問のうち代表的な 6 問を扱った．表 1 に質問式の特徴を示す.
本処理系では探索部分木の構築に必要な情報のみを内部情報として保存する．XML 文書全体のノード数に対して木作成に必要なノード数の割合を表 2 に示す．この割合からメモリ使用量を割り出したものを表 3 に示す.

表 1　XMark の質問式

NO	特徴
Q1	値による絞り込み
Q5	要素数の取得
Q6	要素数の取得，子孫軸
Q13	文字列の検索
Q15	子要素の取得
Q17	子要素の有無

表 2　XML 文書全体のノード数に占める保存されるノード数の割合

Q1	Q5	Q6	Q13	Q15	Q17
5.6%	2.3%	0.3%	1.7%	3.5%	6.1%

表 3　表 2 から割り出したメモリ使用量 (MB)

Q1	Q5	Q6	Q13	Q15	Q17
47	19	3.0	14	29	51

全体のメモリ使用量と処理時間を表 4，表 5 に示す.

表 4　全体のメモリ使用量 (MB)

	Q1	Q5	Q6	Q13	Q15	Q17
SAXON	416	418	413	415	415	426
遅延パーサ	345	345	342	353	359	356
本処理系	34	31	27	36	43	39

表 5　全体の処理時間 (sec)

	Q1	Q5	Q6	Q13	Q15	Q17
SAXON	13.14	13.22	13.09	13.89	13.02	14.16
遅延パーサ	7.88	7.93	7.57	8.06	7.69	8.16
本処理系	7.26	6.95	6.38	6.98	7.16	7.64

今回の実験からメモリ使用量を削減出来たことを確認できた．既存の処理系では探索対象の XML 文書の内部情報を全て保存していたが，本処理系では探索部分木の構築に必要な内部情報のみを保存しているからである.
同様に，処理時間を削減出来たことも確認できた．本処理系では前処理で XML 文書のタグと問い合わせ木の照合をおこない，一致しなければ内部情報の保存処理をおこなわない．このように保存処理を省いたことで内部情報の保存にかかる時間を削減できた.
実験では Q5，Q6，Q13 は表 3 のメモリ使用量の予測とずれがあった．これらはテキストの保存量が多い質問式である．テキストの保存量を保存ノード数の割合から導くことができず, 実験結果が予測値より増えてしまった.

5　考察

本処理系と他の XQuery 処理系を比較する．また，本処理系の実現方法について考察し，内部情報のさらなる削減について述べる.

5.1　他の XQuery 処理系との比較

省メモリかつ高速な XQuery 処理系としてストリーム指向処理系 [2] がある．この処理系は木を構築せず探索対象の XML 文書を走査し，結果を出力する．木を構築し

ないので省メモリかつ高速に処理できるが，結合処理をおこなう質問式は扱えない．本処理系は木を構築しており，その木を結合処理で用いることができる．

5.2 実現方法に関する考察

今回の実験で扱った質問式ではメモリ使用量の削減を確認できた．しかし，内部情報に占めるテキストの量が多い質問式ではメモリ使用量の削減が見込めない．本処理系は配列を用いて内部情報を保存する．リストを用いる方法もあるがオブジェクトを保存してしまうので木を構築しているのと変わりがなくなりメモリを使用量が増加する．内部情報保存量が初期配列のサイズを超える場合，元の 2 倍の大きさの配列を準備し，新しい配列にコピーする．

今回実験で使用した XML 文書に対して

```
for $i in /site/regions
return $i
```

という質問式の問い合わせ処理をおこなった場合，メモリ使用量は既存の処理系が約 520MB，本処理系は約 500MB と，あまり削減できない．これは配列の拡張処理によりメモリを無駄に消費してしまうからである．

次に内部情報保存処理での配列の動きを説明する．

1. 探索対象の XML 文書サイズの一定の割合の大きさの配列を準備
2. 探索部分木構築に必要な内部情報を保存していく．保存する内部情報が多く，配列に内部情報が収まらない場合は配列を拡張する．

2 において初期配列に内部情報が収まる場合，配列の拡張はおこなわないが配列中の使用されない部分がメモリを無駄に消費する．初期配列に内部情報が収まらない場合は新しく配列を用意し，コピーする処理をおこなうので処理時間が増加する．新しい配列にも使用されない部分があり，メモリを無駄に消費する．

今回実験で扱った質問式の内部情報保存量はすべて初期配列に収まったのでメモリ使用量や処理時間について特に大きな増加は見られなかった．

5.3 内部情報を保存せず探索部分木を構築する処理系の提案

5.2 節の問題を解決するために，探索部分木の構築に使われるノードと判断できた時点で探索部分木を構築する処理系を提案する．実現するためにデータ構造を変更し次の処理を現在の前処理に追加する．

- 前処理中に読み込んだノードの情報をもとに探索部分木を構築
- 木を構築したノードの情報は破棄

これにより，内部情報を配列に保持しつづけることがなくなる．内部情報を全て保存してから木を構築する現在の本処理系と比較し，メモリ使用量の削減が見込める．内部情報が破棄された場所に新たに内部情報を保存していくので，初期配列を小さくできる．配列の拡張処理が起こる可能性も低くなり，拡張処理がおこなわれてもメモリ使用量と処理速度には影響が少ないと予測できる．

5.4 改善後のメモリ使用量と処理時間

改善後の処理系のメモリ使用量，処理時間を表 6 に示す．XML 文書は実験で使用したものと同じものを用いた．質問式は，Q1 は 4 章の実験で用いたものの一つである．もう一方は 5.2 節に示したものである．

表 6　改善前後のメモリ使用量 (KB) と処理時間 (ms)

	改善前		改善後	
	メモリ使用量	処理時間	メモリ使用量	処理時間
Q1	34,148	7,280	28,850	7,419
5.2 の質問式	498,857	15,556	240,604	16,275

改善前は大きくメモリを使用した 5.2 節の質問式だが，改善後は約 1/2 に削減できた．Q1 についてもメモリ使用量を削減できており，改善後の処理系は内部情報保存量が少ない質問式にも効果がある．

しかし，改善後の処理系では処理時間が増加してしまう．実現にあたって，既存の処理系とはデータ構造を変更し，おこなう処理が増加したことが原因である．

6 おわりに

本研究は，探索部分木の構築に使われるノードの情報のみを内部情報として保存し，木を構築する省メモリかつ高速な XQuery 処理系を実現した．遅延パーサを用いた既存の処理系と比較して処理時間は約 1.1 倍高速化でき，メモリ使用量も約 1/10 に削減することができた．

今後の課題として複数の質問式についての問い合わせ処理が挙げられる．

質問式が複数になると，質問式ごとに XML 文書を読み込まなければならない．探索対象の XML 文書が同一のものならば効率が悪くなる．解決策として各質問式から生成された問い合わせ木を一つの問い合わせ木として結合する方法がある．しかし，子孫軸が含まれる問い合わせ木の場合は，子孫軸の結合位置の判別ができないので，木の結合が困難である．カレントノードを複数にすることでこの問題を解決し，複数の質問式についての問い合わせ処理を考えていく必要がある．

参考文献

[1] 古川健太, 長瀬安弘, 坂口博紀,“XQuery 問い合わせ処理の最適化,” 南山大学卒業論文, 2008.

[2] 石野明, 竹田正幸,“パスプルーニングによる決定性有限オートマトンを用いた XQuery 処理の提案,” 日本データベース学会論文誌 vol. 4, no. 4, Mar. 2006, pp. 17-20.

[3] M. L. Noga, S. Schott, and W. Lowe,“Lazy XML Processing,” *Proc. of the 2002 ACM symposium on Document engineering*, 2002, pp. 88-94.

[4] Saxonica, *The SAXON XSLT and XQuery Processor*, 2008; www.saxonica.com/.

[5] World Wide Web Consortium, *XML Query Language*, 2007; www.w3.org/TR/xquery/.