検索と更新が並行可能な分割されたXML-DB索引

佐藤 隆士† 張 鵬‡

†大阪教育大学情報処理センター 〒582-8582 大阪府柏原市旭ヶ丘 4-698-1
‡大阪教育大学大学院総合基礎科学専攻数理情報コース 〒582-8582 大阪府柏原市旭ヶ丘 4-698-1
E-mail: †sato@cc.osaka-kyoiku.ac.jp, ‡zp@ss.osaka-kyoiku.ac.jp

あらまし 更新が比較的頻繁に行われるXML文書用の索引管理の研究である。XML文書の経路質問を効率的に処理するため，文書を表す木の節点に構造を反映するラベル付けを行った後，節点ごとに分解して索引に格納する手法が一般的である。検索時には，節点に付けられたラベルから，子孫先祖や兄弟関係などの文書構造上の位置関係に関する情報を得る。従来，XML文書の更新に伴う索引管理コストを低減する様々なラベル付け手法が提案されている。

本稿では索引の更新をバックグラウンドの別プロセスで実行することにより検索処理から更新処理を隠す方法を提案する。索引は検索用と更新用の二系統を用意し，この役割を交替しながら順次更新された索引を検索に利用する。また，索引を一定量のXML文書グループごとに分割管理する手法の提案を行う。索引の並行処理を可能とするとともに，経路質問処理時の{文書番号,ラベル}対の比較範囲が各分割文書グループ内に狭められるため効率化される。

キーワード XML，インデックス，情報検索

Partitioned XML-DB Indices Enabling to Overlap Search and Update

Takashi SATO† and Peng ZHANG‡

†Information Processing Center, Osaka Kyoiku University 4-698-1 Asahiga-oka Kashiwara, Osaka 582-8582 Japan
‡Course of Mathematical and Information Science, Division of Pure and Applied Science, Graduate School of Education, Osaka Kyoiku University 4-698-1 Asahiga-oka Kashiwara, Osaka 582-8582 Japan
E-mail: †sato@cc.osaka-kyoiku.ac.jp, ‡zp@ss.osaka-kyoiku.ac.jp

Abstract This manuscript discusses XML indices which are subject to update frequently. The processing for update is hidden from retrieval by executing update in another background processes. We prepare two sets of indices, one for retrieval ant the other is update, changing their roles while update proceeds. Also we propose the management method of partitioned indices, which correspond to groups of given amount of XML documents.

Keyword XML, Word, Index, Information Retrieval

1. はじめに

XML(拡張可能なタグ付き言語)は標準的なデータ交換手段として用いられるようになるにつれ，大量のXML文書をデータベース(DB)として格納する必要性が増している。XML-DBに対する検索を効率的に処理するためにはXML文書の特性に合った索引が不可欠である。例えば，XML文書の経路質問を効率的に処理するため，文書を表す木の節点に構造を反映するラベル付けを行った後，節点ごとに分解して索引に格納する手法が一般的である。検索時には，節点に付けられたラベルから，子孫先祖や兄弟関係などの文書構造上の位置関係に関する情報を得ている[1-5]。

本稿では更新が比較的頻繁に行われるXML文書を対象にした上記のような構造をもつ索引の更新を扱う。具体的には，バックグラウンドの別プロセスあるいは別プロセッサで索引の更新処理を実行することにより検索処理から更新処理を隠す方法を提案する。この目的のため索引は検索用と更新用の二系統を用意し，この役割を交替しながら順次更新された索引を検索に使用する。また，索引を一定量のXML文書グループごとに

分割管理する手法の提案を行う。索引の並行処理を可能とするとともに，経路質問処理時の{文書番号,ラベル}対の比較範囲が各分割文書グループ内に狭められるため効率化される。 以下，2.では提案の索引の構成を紹介する。3.では分割索引による検索の処理とそのコストを，4.では，二系統の索引を使用した索引の更新処理とそのコストについて論じる。5.はまとめである。

2. 索引の構成

XML-DB 用の索引を複数の L(large)索引とひとつの S(small)索引から構成する。XML 文書番号の昇順に一定量の文書ごとに分割して索引を構成する。L 索引が m 個の場合，各索引は添え字を用いて $L_1,L_2,...,L_m$ と表現される。添え字の大きい L 索引ほど大きな文書番号をもつ文書グループの索引になっている。即ち $i<j$ の時，L_i が索引する文書の番号のいずれもが L_j が索引する文書の番号より小さい。文書を番号の小さいほうから一定量で区切られた残りのグループについて，S 索引を作成する。従って，索引する文書の番号順に並べると $L_1,L_2,...,L_m,S$ となる（図 1 参照)。

各索引の構造は，XML 文書の正規経路質問[1]を効率的に処理するため，文書を表す木の節点に構造を反映するラベル付けを行った後，節点ごとに分解して文書番号とともにラベルを索引に格納する一般的なものを用いる。

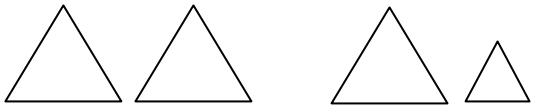

図 1　索引構成の基本形

3. 検索とコスト

3.1. 検索処理

A//B の形式で表される正規経路質問について検索方法を示す。ここで，//は子孫関係にあることを表す正規経路表現である。即ち，文書中に要素 A をもち，且つその子孫(下位層)に要素 B をもつ XML 文書を検索する質問である。本稿では，簡単のため要素による構造検索のみを扱うが，属性やテキストなどの値検索も同様に行うことができる。処理手順は以下のようになる。

(R-1) 各索引 $(L_1,L_2,...,L_m,S)$ について以下の(R-2)と(R-3)を行う。

(R-2) 索引を用いて，要素 A と要素 B の検索を行う。得られた{文書番号,ラベル}対の集合をそれぞれ，E_a および E_b とする。

(R-3) E_a, E_b から要素 A で検索されたラベルと要素 B で検索されたラベルの関係が，子孫関係にある文書番号を取り出す。以後，この{文書番号,ラベル}対の比較処理を**突合せ**という。

(R-4) 必要に応じて，(R-1)から(R-3)で得られた文書番号をマージする。検索に続く処理も分割索引が対象とする文書ごとに処理できる場合は，敢えてマージする必要はない。

3.2. 検索コスト

検索対象の要素サイズを D,要素 A および B による選択度をそれぞれ σ_a および σ_b とする $(0 \leqq \sigma_a,\sigma_b \leqq 1)$。$m>>1$ とし，索引 S の検索コストの影響は無視できるとし，分割索引に対しても要素による選択度が一様であると仮定する。検索処理(R-2)の各分割索引での要素 E_a および E_b の検索の結果，サイズは $\sigma_a D/m$ および　$\sigma_b D/m$ となる。検索にはハッシュなど高速な手法が利用できると，出力のための検索サイズに比例する時間コストで処理される。検索処理(R-3)のラベルによる子孫関係の判定のため，単純に(R-2)で得られた E_a,E_b を突き合わせるとサイズの積 $\sigma_a\sigma_b D^2/m^2$ に比例するコストとなる。この突合せによる選択度を σ_{ab} $(0 \leqq \sigma_{ab} \leqq 1)$とすると，結果のサイズは $\sigma_{ab}(\sigma_a+\sigma_b)D/m$ と表される。結果出力のためこれに比例するコストがかかる。以上まとめると処理コストは，

$$\begin{aligned} CR_{1\text{-}3} &= (\sigma_a+\sigma_b)D/m + \alpha\,\sigma_a\sigma_b D^2/m^2 \\ &\quad + \sigma_{ab}(\sigma_a+\sigma_b)D/m \\ &= (1+\sigma_{ab})(\sigma_a+\sigma_b)D/m \\ &\quad + \alpha\,\sigma_a\sigma_b D^2/m^2 \qquad (1) \end{aligned}$$

となる[2]。ここで α は，突合せと出力の単位コストを整合させるための係数である。分割されている m 個分を合わせると $mCR_{1\text{-}3}$ となる。並列処理しない場合でも索引を分割しない場合に比べ，突合せの部分のコストが $1/m$ になる。また，並列度 $n(\leqq m)$の並列処理システムで行うと $(m/n)CR_{1\text{-}3}$ となる。即ち，索引の分割は並列処理に適しており，処理コストが $1/n$ になる。

なお，検索処理(R-4)は m 個の分割に渡る出力結果の和なので，$\sigma_{ab}(\sigma_a+\sigma_b)D$ となる。突合せ

[1] XML 文書の階層構造を木構造に表した場合，木の枝に沿った経路を指定する質問を経路質問という。更に経路の途中を任意とした質問を正規経路質問という。

[2]索引を分割しない一般的な構成法とする場合の処理コストは，式(1)において $m=1$ としたものになる。

の選択度が $\sigma_{ab} << 1$ となるケースが一般的であるため，この処理コストを含めても索引分割による方法の優位性は変わらない。

4. 更新とコスト

4.1. 更新処理

XML 文書の更新を効率的に索引に反映する方法が提案されている[6-8]。XML 文書変更による XML 木の節点のラベルの付け替えを変更部分に局所化するための工夫がされている。しかし，(1)XML 文書を解析して木構造上変化した部分を同定するために手間がかかる，(2)索引更新中は検索への使用が制限される，などの問題点がある。

そこで本稿では，XML 文書の変更分(差分)の解析を行わず，変更文書は新文書と同等な扱いとし，更新された XML 文書全体の再ラベル付けを行う方式をとる。従って，更新に適した特別なラベル付けは不要である。

各分割索引のうち *S* 索引は，I および II の二系統持たせることにより，検索と再構成を並行に行うことを可能とし，見かけ上の処理コストを低減する。再構成が終了した時点で，系統 I,II を切り替える。また，更新を考慮した分割索引では，2. での構成に加え，最近更新された XML 文書でまだ *S* 索引に反映されていない文書を，検索のためメモリ上に DOM 形式など（「**メモリ形式**」という）でも置いておく。なお，これらの処理はトランザクションに基づいた管理を必要とする。即ち，あるトランザクションのコミットされていない更新内容は，他のトランザクションに見せない。このためにも *S* 索引への反映のタイミングを図る必要がある。

以下に更新処理内容を箇条書きにする。

(U-1)文書削除時には，即座に索引へ反映することはしない。**削除リスト**を作成するのみで，索引や文書ファイルからの実際の削除はしない。従って，検索時には，検索された文書から削除リストに含まれる文書は除くことになる。

(U-2)メモリ形式にない文書の修正は，削除と追加の組みで行う。即ち，修正を行うと新たな文書番号が付与されることになる。従って，更新前の文書番号で参照する必要がある場合は，別途（旧文書番号：新文書番号）形式の**対照表**を用意する。

(U-3)新規に追加された文書(修正で新規扱いされたものも含む)について，メモリ形式を作成する。これに伴い，検索時には，$L_1,L_2,\ldots,L_m,S$ 索引に加えメモリ形式分の検索も行う必要がある。

(U-4)メモリ形式が一定量になると，比較的頻繁に更新されている文書以外から *S* 索引を作成する[3]。例えば作成過程にある XML 文書は更新が頻繁なため索引に反映しないほうがよい。*S* 索引は I,II の二系統を切り替えて使用する。すなわち系統 I が更新中の時は系統 II を検索に使用する。系統 II を更新する時点では逆の切替を行う。メモリ形式中，*S* 索引に反映されたものは削除する。*S* 索引用の削除リスト（*S* 削除リスト）は空に初期化される。

(U-5)*S* 索引は，一定サイズに成長すると *L* 索引に昇格する。*m* は 1 増加する。それに伴い空の *S* 索引が用意される。

(U-6)全文書数に対する削除文書リストが一定割合以上になると，*L* 索引の再構築(クリーンアップ)を行う。*L* 索引用の削除リスト（*L* 削除リスト）は空に初期化される。この作業は，夜間の定時バッチなどで他の処理を止めて行うことが想定されるが，処理が止められない場合は，*L* 索引についても I,II の二系統を切り替えることにより，更新と検索を並行することができる。

(U-7)各 *L* および *S* 索引に対する検索質問と検索結果は，一定量キャッシュしておく。同じ質問は，キャッシュを利用して処理コストを下げる[4]。

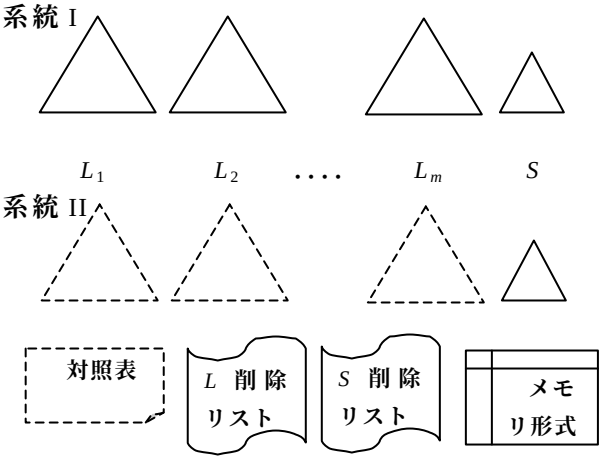

図 2　更新を考慮した索引構成

[3] 頻繁な文書更新処理を索引に反映することを避けて，メモリ形式に集約することにより効率化するものである。なお，更新頻度はメモリ形式中の文書リストの更新記録から得るものとする。

[4] 削除リストに文書番号が追加される可能性があるため、以前の質問結果のキャッシュをそのまま利用することはできない点に注意を要す。

以上で説明した索引構成を図2に示す。Pollari-Malmら[9]は，B木に対して更新を遅らせバッチ更新する方法を提案している。更新を即座に索引には反映しない点で，本稿で提案の方法を関連している。しかし，バッチ更新で個々の更新を別々に行うより効率化されるとは言え，2次記憶上の大型索引に対して複数のエントリを更新するのは高コストな処理と言える。本稿では，分割された小索引（S索引）を再作成している点に特徴を有す。なお，(U-5)の一定サイズとは，S索引がメモリ上で効率よく作成されるサイズを上限としておくべきである。

4.2. 更新コスト

XML文書の更新を即座に索引に反映しないので，索引の更新コストは検索側にはほとんど影響を与えない[5]。検索に負担をかけない優先度の低いバックグラウンドの別プロセスによる処理，あるいは複数プロセッサをもつシステムにおいては，検索処理とは異なるプロセッサの処理とすることができる。

全XML文書の検索対象となる要素数を検索コスト見積もり時と同じくDとする。まず，全XML文書用の索引を分割せず一つにまとめて構成する場合について考える。検索対象の(要素,文書番号,ラベル値)の組のソーティングが主要コストとなる。外部ソートで一般的なマージソートで行うと，$D\log D$に比例する時間がかかる。一方，提案の更新方法では通常の索引更新はS索引に対してであり，対象は全体の$1/m$以下の比較的小さな索引部分の再構成になる。従って，更新コストは$(D/m)\log(D/m)$以下になる。

L索引更新の際は，Dから削除リスト分を減じた要素数を検索対象とする索引の再構成となる。この際も検索時と同様，複数個(削除リスト分が除かれるのでm以下)の分割索引を複数プロセッサで効率的に処理することができる。並列度nの並列処理システムで行うと，$(D/n)\log(D/m)$以下に比例するコストとなる。

5. おわりに

比較的頻繁に更新の行われるXML-DB用の索引として，二系統からなり，更に各系統が分割された索引構成を提案した。この索引構成を用いて，索引の更新処理をバックグラウンドの別プロセスあるいは，別プロセッサで実行することにより，検索処理から更新処理を隠すことができることを示した。また，索引の分割管理により索引の並行処理を可能とするとともに，経路質問処理時の{文書番号,ラベル}対の比較範囲が各分割文書グループ内に狭められるため効率化されることを示した。

[5] 削除リストはメモリ上に置き、且つ文書番号をキーとした探査木で構成するため、ディスクなど2次記憶装置上の索引へのアクセスに比べ処理コストが小さい。

文　献

[1] Quanzhong Li and Bongki Moon: “Indexing and Querying XML Data for Regular Path Expressions”, in *Proceedings of the 27th VLDB Conference*, Roma, 2001.

[2] Hain Kaplan, Tova Milo and Ronen Shabo: “A Comparison of Labeling Schemes for Ancestor Queries”, in *Proceedings of 13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp.954-963, 2002.

[3] Yong Kyu Lee, Seong-Joon Yoo and Kyougro Yoo: “Index Structures for Structured Documents”, in *Proceedings of the ACM 1st Conference on Digital Libraries*, Bethesda, MD, March 1996, pp.91-99.

[4] 佐藤,里本,小畑,潘:階層構造を認識可能な木節点の番号付け, DEWS2002, 109, March 2002.

[5] 佐藤,李,居,張:基数組集合による木節点の番号付け, DEWS2003, 3-C-1, March 2003.

[6] Edith Cohen, Haim Kaplan and Tova Milo: “Labeling Dynamic XML Trees”, in *Proceedings of 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pp.271-281, 2002.

[7] 小林,小林,横田：挿入制限のないXML範囲ラベリング用コード,情処研報,情報処理学会,Vol.2003, No.71, pp.41-48, 2003.

[8] Toshiyuki Amagasa, Masatoshi Yoshikawa and Shunsuke Uemura: “QRS: A Robust Numbering Schema for XML Documents”, in *19th International Conference on Data Engineering (ICDE 2003)*, pp.705-707, March 2003.

[9] Kerttu Pollari-Malmi, Eljas Soisalon-Soininen and Tatu Ylonen: "Concurrency Control in B-Trees with Batch Updates, *IEEE Transactions on Knowledge and Data Engineering*, Vol.8, No.6, pp.975-984, 1996.