様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成 ２７ 年　 ６ 月　 ９ 日現在

機関番号：１４３０１
研究種目：挑戦的萌芽研究
研究期間：2011～2014
課題番号：２３６５００４８

研究課題名（和文）木構造データの複数パラダイム混在処理方式に関する研究

研究課題名（英文）Efficient Processing of Multi-Paradaim Operations on Tree-Structured Data

研究代表者
田島　敬史（Tajima, Keishi）

京都大学・情報学研究科・教授

研究者番号：６０２８３８７６

交付決定額（研究期間全体）：（直接経費）　　2,800,000円

研究成果の概要（和文）：木構造データに対する処理の主な形態として，木構造内の枝をたどって節から節へと移動するナビゲーションを行いながら処理を行う形態，木構造内の節を前置順などの特定の順序で走査しながら処理を行う形態，パス式などによるパターンマッチを用いて特定の箇所を抽出して処理を行う形態の三つがある．本研究では，これらの異なる形態が混在するような処理を効率的に実行する手法を開発した．
より具体的には，ナビゲーション処理と検索処理の効率的な処理を同時に実現するような木構造データのディスクへの格納方式や，ナビゲーション処理と走査処理が混在する処理列を，処理の順序を入れ替えることで効率良く実現する手法などを開発した．

研究成果の概要（英文）：There are three paradigms of processing of tree data: processing based on navigation from node to node through edges, processing based on scanning of nodes in some order, e.g., pre-order, and processing based on queries, such as those expressed in path expressions. We developed several methods of efficiently executing operations on tree data including multiple paradigms.
For example, we developed a storage scheme for tree data that achieves efficient execution of both navigation-based operations and query-based operations at the same time. We also developed a scheme for efficiently processing a sequence of operations on tree data including both navigation-based operations and scan-based operations. Our scheme repeatedly transforms the data between a data structure for navigation-based operations and a data structure for scan-based operations. To reduce the number of conversions, it changes the order of operations so that the same kind of operations are contiguous as much as possible.

研究分野：データ工学

キーワード：木構造データ　ナビゲーション　走査　検索　パス式　記憶方式　データ構造　処理方式

１．研究開始当初の背景

１９８０年代において，大規模データの蓄積，変換，検索のためのデータ形式としてもっとも広く用いられていたのは関係と呼ばれるデータ表現を用いるデータ形式であったが，１９９０年代初めごろより，関係の形では表現しにくいデータを扱うためのデータ形式として，木構造データの形でデータを表現するデータ形式が様々な場面で用いられるようになった．そのような例として，タンパク質データベースや遺伝子データベースなどの科学技術データベースで用いられていたいくつかの専用データ形式や，インターネット上での汎用データ交換フォーマットとして１９９０年代半ばに規格化されたＸＭＬ形式などがある．

これに合わせ，１９９０年台半ばより，木構造データの形で表現された大規模データを効率的に処理するための手法に関する研究が多数行われた．特に，データベース技術に関する研究コミュニティと，プログラミング言語に関する研究コミュニティが研究の中心となった．それ以降，本研究を開始した２０００年代終わりまでには，木構造データの蓄積，変換，検索に関する研究はやり尽くされたのではないかと思われるほど，多くの研究が行われた．

しかし，これらの研究は，データベース技術に関する研究コミュニティと，プログラミング言語に関する研究コミュニティで，最後まで主にばらばらに行われたため，これら二つの研究コミュニティそれぞれにおける主要なテーマに関する研究は数多く行われたが，これらのコミュニティにまたがるようなテーマの研究は必ずしも十分には行われなかった．

例えば，木構造データに対するどのような処理を主に想定するかについても，データベース技術に関する研究コミュニティでは，パス式と呼ばれる，ワイルドカードを含む木パターンを記述して，この木パターンにマッチする全ての箇所を，大規模な木データの中から効率よく抽出する検索処理を中心に考えていたが，プログラミング言語の研究コミュニティでは，木構造データ中の枝に沿って，ある節からその子や親の節へとナビゲートしながら行うような処理が中心に考えられていた．そして，これらの研究は主にばらばらに行われていたために，その双方のタイプの処理が混在するような処理を効率的に行うための手法に関する研究はほとんど行われていなかった．

また，木構造データに対する，もう一つの主要な処理形態として，木構造中の全ての節を前置順などの特定の順番で走査しながら処理を行っていくという形態がある．このような処理は，ネットワーク上を流れる木構造データを走査しながら処理を行いたい場合や，メモリに入らないような大きなデータに対して，その全体をメモリ中に同時に保持すること無く，全体を一回走査するだけで処理を行いたい場合などに有用である．このような処理を効率良く行う手法についての研究も多く行われた．しかし，このような走査処理と，前述の検索処理やナビゲーション処理が混在する処理を効率良く行う手法に関する研究も行われていなかった．

２．研究の目的

そこで，本研究では，上述の三つの異なるタイプの処理：
（１）木構造中をエッジに沿って様々な方向にナビゲートしながら行う処理，
（２）木構造を深さ優先前置順等の特定の順序で走査しながら行う処理，
（３）パス式によるパターンマッチ等を用いた検索処理
が混在するような処理を木構造データに対して効率的に行う手法を開発することを目的とする．これら三つの種類の処理が，木構造データに対する代表的な処理形態であり，これら三つの処理形態が混在する場合を扱えれば，ほぼ全ての場合をカバーできると考えられる．

より具体的には，そのような状況の中でも，特に以下の二つについての研究を重点的に行う．
（１）木構造データがディスクに格納されており，それに対してナビゲート処理によるアクセスや検索処理によるアクセスが同時に発生する場合（図１参照）を想定した，これらの処理のいずれをも効率的に処理可能な，木構造データの格納手法．

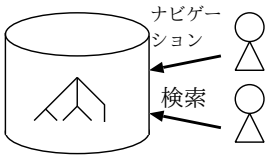

図１：ディスク中の木構造データへの異種操作によるアクセス

（２）ネットワーク上を流れる木構造データに対して，ナビゲート処理によるアクセスや走査によるアクセスが混在する処理列が順に適用される場合（図２参照）を想定した，これらの処理列を効率的に処理可能なデータ構造，あるいは，処理実行手法．

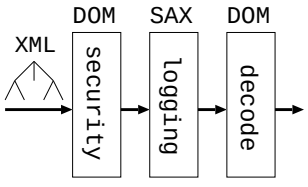

図２：ネットワーク上を流れる木構造データへの異種操作が混在する処理列

前者のような場合としては，例えば，近年，バイオ，天文学，核物理学など，様々な分野の学術データや実験データがインターネット上に公開されているが，これらのデータの中には，巨大な木構造のデータで表現されているもの

が多くあり，そのようなデータに対しては，検索によるアクセス機能が必須である一方，単純な条件式による検索が困難で，その分野の専門家が閲覧・吟味しながらデータ間をナビゲートしていくことによって必要なデータを探すしかないような場合がある．そのような場合，世界中の多くの学者が同時に多数の検索処理とナビゲーション処理をストレスを感じない処理時間で行えることが重要である．

後者のような場合の例としては，ネットワーク上を流れるメッセージパケットのフィルタリング処理がある．大規模なＷｅｂサービスを提供するサーバなどでは，ＸＭＬ形式で記述された大量のメッセージが順に到着し，これらがセキュリティ検査，ログ，暗号の複合処理などの様々な処理を通過していく．大規模システムでは，これらの処理は，独立に開発されたり，既存のものを組み合わせて利用したりするため，ＤＯＭ等のナビゲーション型のライブラリを使って記述された処理と，ＳＡＸ等の走査型のライブラリを使って記述された処理が混在することがある．このような場合，サーバのスループット性能を上げるためには，これらの異なる種類の処理からなる処理列を，効率よくパイプライン処理することが必要である．

３．研究の方法

しかしながら，そのような異なる種類の処理の効率性を両立するのは簡単ではない．例えば，本研究の代表者は，これまでにナビゲーション処理に適した木構造データのディスクへの格納方式に関する研究（引用文献［１］）を行ったが，その研究の中で，ナビゲーション処理に最適な格納方式と，検索処理に最適な格納方式との間にはトレードオフがあり，双方にとって同時に最適にはなりえないことが示されている．

また，前述のＤＯＭとＳＡＸは木構造データの一種であるＸＭＬデータを処理するプログラムを作成する際の方式としてもっとも広く用いられているものだが，ＤＯＭはナビゲーション型のアクセスを用いて処理を記述し，ＳＡＸは走査型のアクセスを用いて処理を記述する．このような二つの方式が用いられているのは，これら二つの間に，前者がプログラミングが容易だがメモリ効率が悪いのに対し，後者はメモリ効率が良いがプログラミングが煩雑であるという，明確なトレードオフがあり，これらの双方の利点を両立する方式の実現がこれまで強く望まれていながら，いまだに実現していないためである．

これらのことからわかるように，単純に一つの方式で，複数の処理形態を効率よく実行することを同時に実現することは容易ではない．そこで，本研究では，異なる種類の処理が混在する場合の効率的な実行方式として，大きく以下の二つのアプローチを検討する．

（１）異なる種類の処理のいずれをもある程度効率よく実行可能なデータ構造を開発する方法．
（２）異なる種類の処理それぞれに適した別個のデータ構造を併用する方法．

後者の（２）については，さらに以下の二通りの方法が考えられる．

①それらの各データ構造でのデータ表現を同時に保持する手法，
②同時には一種類のデータ構造しか保持せず，必要に応じて，他のデータ構造への変換を行う手法．

本研究では，これらのアプローチを用いて，前述のような手法を実現することを目指して研究を行った．

４．研究成果

本研究の一つ目の成果は，上述のアプローチ（１）を用いて，ディスク中の木構造データへのナビゲーション処理とパス式による検索処理の双方を効率良く実行する手法を開発したことである．

開発した手法は，基本的には，本研究の研究代表者らが前述の引用文献［１］で開発した，ナビゲーション処理に最適化されたデータ構造を用いてる．［１］で開発した手法は，木の節をディスク上にある制約を満たすような特殊な順序で並べて格納することにより，ナビゲーション処理をディスク上の少数の領域へのシーケンシャルアクセスのみによって実現できるようにしたものである．この手法は，１ステップのナビゲーション処理に最適化されているため，複数ステップからなるパス式による検索処理を，単純に，パス式中のステップの数だけナビゲーション処理を繰り返す形で実現したのでは，ランダムアクセスが発生し効率が悪い．

そこで，本研究では，［１］で開発した並べ方にさらに制約を加え，１ステップのナビゲーションの性能を落とさない範囲で，ｎ個のステップを含むパス式による検索処理についても，単純に［１］で開発した１ステップの処理をｎ回繰り返すよりも効率的に実現する手法を開発した．

また，二つ目の成果として，当初の目標にはなかったことではあるが，上述の並べ方を用いて，与えられたパス式にマッチする箇所のうち，最小マッチや極小マッチにあたる部分のみを効率よく発見する手法についても開発した．

本研究の３つ目の成果は，上述のアプローチ（２）①を用いて，一つの木構造データに対して同時に発生するナビゲーション処理と走査処理の双方を効率良く処理する手法を開発したことである．開発した手法では，ナビゲーション処理に適したデータ構造と走査処理に適したデータ構造を併用し，一方に生じたデータの更新を必要時まで遅延して他方にも波及させることで，双方のデータの一貫性を保ちながら，

二種類の処理の効率的な実行を同時に実現する（図３参照）．

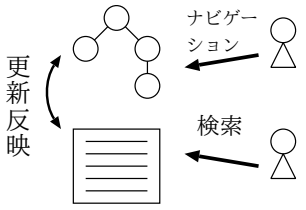

図３：アプローチ（２）①

本研究の４つ目の成果は，上述のアプローチ（２）②を用いて，ネットワーク上を流れる木構造データに対して，ナビゲーション処理と走査処理が混在する処理列を効率良く実行する手法を開発したことである．開発した手法では，一つの木構造データを，ナビゲーション処理の際にはナビゲーション処理に適したデータ構造に変換し，走査処理の際には走査処理に適したデータ構造に変換しながら，処理を行っていく（図４参照）．ただし，その際に，処理結果を変更しない範囲で，処理列内の処理の順序を入れ替えて，できるだけ同種の処理が連続するようにすることにより，データ構造の変換の回数を最小限にするようになっている．

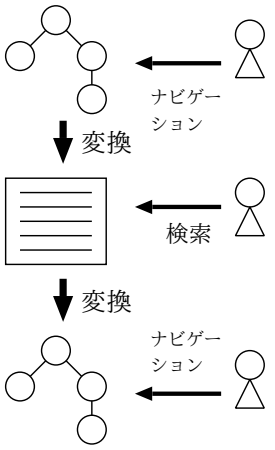

図４：アプローチ（２）②

＜引用文献＞

［１］Atsuyuki Morishima, Keishi Tajima, Masateru Tadaishi, *“Optimal Tree Node Ordering for Child/Descendant Navigations,”* in Proc. of 26th IEEE Intl. Conf. on Data Engineering (ICDE 2010), pp.840–843, 2010

５．主な発表論文等

本研究で得られた研究成果については，現在，成果の公表に向けて，論文作成，論文投稿作業中である．

６．研究組織

（１）研究代表者
田島　敬史（ＴＡＪＩＭＡ，Ｋｅｉｓｈｉ）
京都大学・情報学研究科・教授
研究者番号：６０２８３８７６