木構造型データの管理方式について

2R－7

西岡秀一　小林伸幸　夏目義久　小西史和

ＮＴＴ情報通信研究所

１．はじめに

ディレクトリサービスや種々の組織図のように、様々な属性を持つデータが木構造として管理されている状況が多く存在する。著者らは、高度通信サービス等での利用を目的とした、高速で高信頼な主メモリ常駐型RDBMSの研究を行っている[1]。そのため、本稿では、主メモリ常駐型RDBMSをベースに、木構造自体を直接扱えるようにRDBMSを拡張することで、木構造型データに対する高速な検索を実現する方式を提案する。さらに、OSIディレクトリサービスを木構造の例に取り、プロトタイプを試作し、その性能を実測した。

２．OSIディレクトリサービス[2]

OSIディレクトリは、通信の対象となる"物"に関する情報のデータベースと、ディレクトリ利用者エージェント(DUA)と呼ばれる応用プロセスからこれらの情報にアクセスする機能を規定している。OSIディレクトリが提供するデータベースをディレクトリ情報ベース(DIB)と呼び、ディレクトリ情報木(DIT)と呼ばれる木構造で表現される(図１)。DITにおける節はエントリを、また枝はエントリ間の従属関係を表す。一つの節から出る枝にはそれぞれ他の枝と異なる名前(RDN：相対識別名)を付与し、木構造の根(ROOT)から特定のエントリに至るRDNの列をDN(識別名)と呼び、そのエントリをDIT内で一意に識別する。

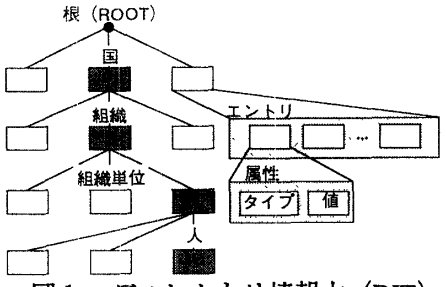


図１　ディレクトリ情報木（DIT）

A Management Method for Tree Structured Data

Shuichi Nishioka (nishioka@dq.isl.ntt.co.jp)
Nobuyuki Kobayashi (kobayasi@dq.isl.ntt.co.jp)
Yoshihisa Natsume (natsume@dq.isl.ntt.co.jp)
Fumikazu Konishi (konishi@dq.isl.ntt.co.jp)

NTT Information and Communication Systems Laboratories

３．従来方式での問題点

DITを表形式で表現する場合、大きく分けて以下の方法がある。

(1)複数段をまとめて表で表現する方法[3]

DIT全体を図２のように一つの表で表現する。

DITテーブル

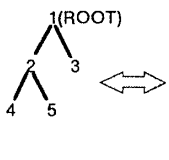
RDN（１段）	RDN（２段）	…	RDN（ｎ段）
ROOT			
ROOT	日本		
ROOT	日本	…	西岡

エントリ

図２　DIT全体を表で表現

(2)木の上下関係を表で表現する方法[4]

各エントリをDIT内で一意に識別するためのキーを定め、各エントリとその直接上位のエントリのキー値を格納するカラムを設けて、上位下位の対応関係を表で表現する（図３）。



エントリ番号	上位エントリ
1	
2	1
3	1
4	2
5	2

図３　上下関係の表による表現

主メモリ常駐型RDBMSは次のような特徴がある。

・主メモリに制限がある。

→　使用メモリ量を極力削減したい（要求１）

・ディスクに対するI/O遅延が無いので、プロセス数の増加に伴いCPUネックが起こり、プロセス数に対して線形にレスポンスが低下する。

→　各処理の負荷を極力抑えたい（要求２）

つまり、上記の方式を主メモリ常駐型RDBMSに適応した場合、

(1) 複数段をまとめて表で表現する方法

あらかじめ木の段数分のカラムを用意するため、格納効率が悪い。

→　要求１に適さない

(2) 木の上下関係を表で表現する方法

木を一段探索する（たどる）たびに、表に対する検索が必要

→　要求２を実現するためには、各処理の負荷を

更に抑える必要がある

となり、主メモリ常駐型DBMSに期待されるだけの性能を引き出すことができない。よって、格納効率を極力犠牲にすることなく、負荷の少ないアクセス手法が必要となる。

4．提案方式の特徴

提案方式では、上記の問題点を解決するために、RDBMSを拡張して木構造型データを格納し、木構造に適したアクセス手法を提供することで、木構造型データに対する高速なアクセスを可能にする。

(1) 物理IDによるリンク情報の格納

DITのエントリをレコードとして表に格納し、各レコードの物理的な位置を表すID(レコードID、ポインタ等）を付与し、DITにおけるエントリ間のリンク情報として、このIDを格納する。このことにより、木の中の任意のエントリをたどることが、表に対する操作（JOIN操作やインデックス経由の検索）を行うこと無く可能になり、高速な検索が実現できる。

(2) 固定長レコードによるDITの表現

DITでは、あるエントリとその下位のエントリとの間には1:nの関係が存在する。そこで、あるエントリから下位のエントリへの枝(リンク)は一つとし、下位のエントリ同士を左右で連結させる(図4)。このことにより、1:nの関係を可変長レコードを用いずに、固定長レコードで表現することで、使用メモリ量を抑えることが可能となる。

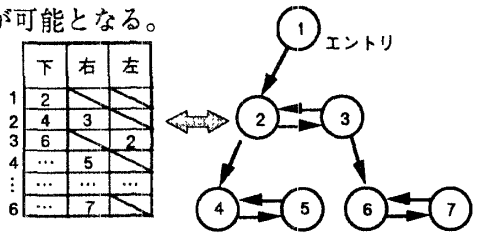


図4　提案方式によるDITの表現

5．提案方式を用いた実測

提案方式を用いたプロトタイプシステムを作成し、OSIディレクトリサービスが提供する大部分の操作に含まれる「名前解析」を例に実測を行った。名前解析処理とは、DIT内で一意に識別可能なDNを構成するRDNを基に、DITを目的のエントリまで検索する処理である。提案方式の性能を見るために、木の深さを5段、エントリ件数10740件のDITに対して、DNをランダムに発生させて、名前解析処理を実測した結果を図5に示す。また、ディスクベースRDBMSで3．の(2)の方式を用いて名前解析処理を行った結果も示す。ただし、ディスクベースRDBMSではバッファ域を充分大きく取り、全てのデータをメモリ上に上げてから行なった。

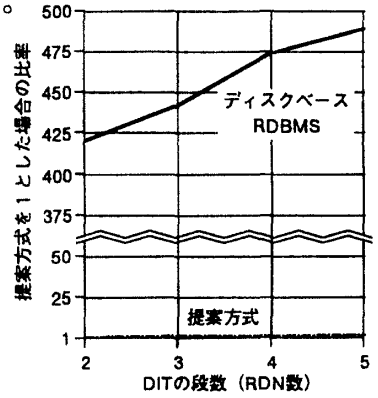


図5　名前解析処理の応答時間

図5は、提案方式による処理時間を1とした場合の相対グラフである。DITの名前解析処理における一段当たりの処理時間は、グラフの傾きで表される。提案方式は、ディスクベースRDBMSに対して高速性を確保し、さらにDITの段数を増加させても、名前解析処理時間はさほど増加しなかった。一方ディスクベースRDBMSの場合、段数の増加に伴い、線形に処理時間が増加していき、その増加率は提案方式の約100倍となった。これは、一段名前解析処理を行うたびに、表に対する検索を行っているためであると思われる。

6．おわりに

本稿では、木構造を直接扱えるようにRDBMSを拡張することで、高速な検索を実現する方式を提案した。またOSIディレクトリサービスを木構造の例に取り、提案方式をプロトタイプシステムに実装し、実測を行った。

今後は、主メモリ常駐型RDBMSにおいて、従来の方式と提案方式を用いた場合を比較した評価実験を行い、提案方式の木構造型データの管理に対する有効性の検証を行う。

参考文献

[1]T.Honishi,etc.,"Design and Implementation of an Enhanced Relational Database Management System for Telecommunication and Network Application",18th Annual Pacific Telecommunications Conference(PTC'96),Jan. 96,T.1.7.4

[2]CCITT Recommendations X.500-X.521(1998)

[3]小花 他,"リレーショナルアプローチによるOSIとディレクトリのDIB(ディレクトリ情報ベース)の実装と評価", 情報処理学会論文誌 ,Vol.32,No.11,pp.1488-1497,1991.

[4]空 他,"ＲＤＢＭＳによるＯＳＩディレクトリの実現", 情処学会データベースシステム 95-10,1993.9.9.