木構造による図面データのバージョン管理

2X－6

木村 公一　出木原 裕順　中村 泰明
広島市立大学情報科学部

1　はじめに

地図情報管理システム・設計プロセス管理システムなどの分野では、地図や設計図といった大量の図面情報をデータベース化することにより、図面の検索・修正などを効率的に管理する図面管理システムの開発が盛んである[1]。

図面管理システムでは、最新バージョンの図面だけでなく、旧バージョンの図面を引用することが頻繁にあるので、各バージョンの図面をより効率的に管理するシステムが必要となってくる。

本稿では、多数の図面データのバージョンを効率的に管理する方式を提案する。

2　図面のバージョン管理

図面のバージョン管理とは、過去から現在に至るすべての図面がどのような順序でどのようにバージョンアップしていったかを管理するシステムである。

図1に示すように図面が更新された場合、従来の方法では、各バージョンに対して図面ファイルが存在していた。しかし、この方法ではバージョン数に比例したデータ量の管理が必要となってくる。一般に、バージョン間でのデータの変化は少なく、それぞれ個々の図面ファイルとして管理する場合、重複するデータが多くなり非効率的である。そこで、このようなデータ重複を削減し、かつ高速な検索が可能なバージョン管理方式を考える。

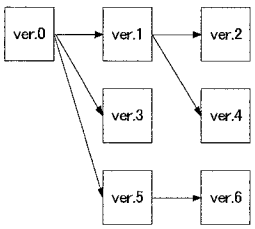

図1　図面のバージョン管理

Multiple Version Management for Drawings by Hierarchical Data Structure

Koichi Kimura, Hiroyuki Dekihara and Yasuaki Nakamura
Faculty of Information Science, Hiroshima City University

3　R木

R木[2]は、図面データ（空間データ）の効率的な管理方式であり、空間位置に基づくデータ検索が高速に実行できることから空間データ管理で利用されることが多い。R木では、図面データを葉に蓄積する。葉に入るデータは長方形であり、ノードはそのノード以下の部分木に含まれる全データを包含する最小の長方形のみを管理する。

4　MVRによるバージョン管理

MVR(Multi Version R-tree)とは、R木によるバージョン管理方式であり、以下の手順で構築される。

① 新しく作成するバージョンを管理するR木のルートは、旧バージョンのルートの子ノードへのポインタを持つ(図2の(1))。

② データが追加されたときにたどったルートから葉までのパスをコピーし、作成する(図2の(2))。

③ たどった葉に空き領域がないときは分割を行う(図2の(3))。

この方法で管理することにより、MVRによるバージョン管理では、各バージョンを単独のものではなく、全てのバージョンを共有することで、重複するデータを一ヶ所で管理することができる。

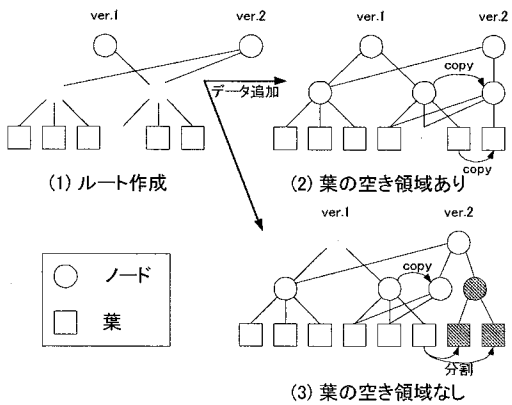

図2　MVRによるバージョン管理

しかし、この方法では、データを追加したとき、葉に空き領域があるにもかかわらず葉をコピーする。その結果、必要メモリ量が増加することが予想される。そこで、葉の空き領域を活用することにより、メモリ量の削減を図る。

5　MVR*によるバージョン管理

MVR によるバージョン管理を改善するために、MVR に以下のような変更点を加えたものを MVR* とする。

④ 葉に空き領域があるかどうかを判別するための構造体を作成し、ノードと葉の間に入れる(図３の(1))。

⑤ 葉に空き領域がある場合、新しい葉を作成せず、空き領域のある葉にデータを追加する(図３の(2))。葉に空き領域がない場合、新しい葉を作成する(図３の(3))。

この方法で管理することにより、葉の空き領域を確認でき、不要な葉を作成することなく効率的に管理することができる。

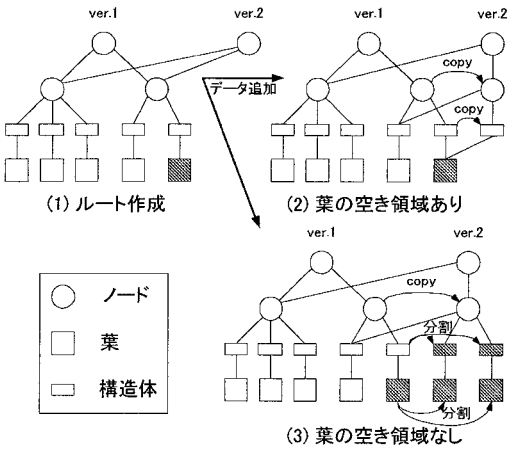

図３　MVR*によるバージョン管理

ここで、MVR と MVR*について比較してみる。葉に空き領域がない状態では、どちらも新しい葉を作成するので葉数は等しい。しかし、葉に空き領域がある状態では、図３で見られるように MVR*は旧バージョンへのポインタを持ち、葉を作成しないので MVR より葉数は少なくなると考えられる。

6　実験、及び結果

R 木、MVR、MVR*によるデータ管理効率の指標として必要なメモリ量を以下の手順で比較する。シミュレーション実験による結果を表１に示す。

① 最初のバージョンに１万、５万、10 万の小長方形データを与える。

② 最初のバージョンから５つのバージョンを作成する。

③ バージョンアップするごとに、最初のバージョンのデータ数の 10%をデータとして追加する。

本実験での R 木では、ノードは最大３個の子を持ち、葉は 20 個のデータを持てるものとした。

表１　バージョン管理の結果

		初期データ数		
		10000	50000	100000
R 木	ノード数	4998	22130	42380
	葉数	5104	25117	49992
MVR	ノード数	2283	8380	14655
	葉数	2392	9302	17046
MVR*	ノード数	2283	8380	14655
	葉数	1429	6882	13392

最初のバージョン(初期データ数 100000)を管理する R 木のノード数、葉数はそれぞれ 5545 個、7320 個であった。表１より R 木の場合、ノード数、葉数ともバージョン数に比例して増加するが、MVR、MVR*では、データの増加数に比例する。このため、R 木の場合に比べ、MVR、MVR*の葉数は、それぞれ 66%、73%削減された。

MVR と MVR*の場合、ノード数は等しいが、葉数は 20～40%削減された。すなわち、MVR*方式は、総メモリ量の削減に貢献している。検索効率に関しては、いずれの場合もほとんど差は見られなかった。

7　おわりに

本稿では、効率的な図面管理システムを実現するために MVR を提案し、R 木の管理に比べ、大幅なメモリ効率の向上を確認した。また、MVR*によりさらなるメモリ量の削減を実現した。

参考文献

[1] R.H.Katz : “Toward a Unified Framework for Version Modeling in Engineering Databases”, ACM Computer Surveys, Vol.22, No.4, pp.375-408, 1990.

[2] A.Guttman : “R-tree・a Dynamic Index Structure for Spatial Searching”, Proc. SIGMOD, pp.47-57, 1984.