5V-7 木構造によるマルチレイヤ・データの管理

中村 泰明* 阿部 茂* 大沢 裕** 坂内 正夫**
*三菱電機株式会社 中央研究所 **東京大学 生産技術研究所

1．はじめに

2次元、3次元空間の点、ベクトルなどのデータの効率的な管理手法として、木構造を用いたデータ管理手法が知られている[1]。しかし、地図や設備図面などの場合には、各データは、道路、家屋、地図記号、文字などの属性によりレイヤ分類し、管理する必要がある。図面の編集時には、複数のレイヤが対象となり、その際のデータ検索は、当然複数レイヤのデータが対象となる。このようなマルチレイヤ多次元データの管理手法として、木構造を用いた新しい管理構造（ＭＬ構造（Multi-Layer structure））を提案する。

2．木構造による従来のマルチレイヤデータ管理構造

マルチレイヤ多次元データは、位置座標とレイヤ属性Ａi（i=1..L)により表現される。木構造（ＢＤ木[2]）により、マルチレイヤ・データを管理する手法として、次のような方法が考えられる（図１）。

(1)ＭＴ(Multi-tree)構造…各レイヤー毎にＢＤ木を作成し、それらの木の根をさらに配列等で管理する。

(2)ＳＴ（Single-tree)構造…レイヤを区別しないで、ＢＤ木で管理し、各データに属性情報を持たせる。

ＭＴ構造は、単一レイヤを指定した範囲検索は、非常に効率が良い。しかし、複数レイヤを指定すると効率は悪くなる。ＳＴ構造は、逆の特性をもつ。

3．ＭＬ構造の概要

図２にＭＬ構造の概念図を示す。ＭＬ構造は、分割領域を管理する木構造(領域管理木)にレイヤを管理する機構(レイヤ管理構造)を付加したものである。領域管理木にはＢＤ木を採用する。レイヤを管理するために、ＢＤ木のノードに葉へのポインタ配列を付加する。領域管理木のノードＩは、次の要素から構成される。

Ｉ＝｛Ｌp, Ｒp, Ｒ, Ｅ, Ｍp[L]｝,

ただし、Ｌp：左部分木へのポインタ、Ｒp：右部分木へのポインタ、Ｒ：分割長方形領域を表すベクトル、Ｅ：データの存在記述式、Ｍp：葉へのポインタ配列、Ｌ：レイヤ数。Ｅは、レイヤ当り２ビットの容量をもち、そのノードの葉、およびそのノード以下の部分木中にどのレイヤのデータが存在するかを示す。

葉Ｌは、次の要素から構成されている。

Ｌ＝｛Ｄ[P]、ｎ、Ｒ｝

ただし、Ｄ[P]：データを格納する配列、Ｐ：格納できる最大数（データ容量）、ｎ：データ格納数、Ｒ：その葉の中の全てのデータを包含する最小の長方形。

データは、すべて葉に存在し、かつ、１つの葉には、同一レイヤのデータのみが存在する。ＭＬ構造では、データ数の少ないレイヤのデータは、領域管理木の根に近い位置に存在する。

3．1 ＭＬ構造の構成法

ＭＬ構造の作成手続きを示す（図３参照）。

(1)INSERT：レイヤjのデータdを投入する。ＭＬ構造の根をrootとする。

①Ｉ←root。

②レイヤｊのデータが、Ｉ以下のＭＬ構造中に存在しないとき、③へ。Ｉの葉に存在するとき、④へ。Ｉの左右部分木に存在するとき、⑤へ。

③新しい葉（Ｌn）を作成し、Ｌnにｄを挿入。ＬnをＩに付加する。終了。（図３データ１、３を投入）

④Ｉの葉からレイヤｊに対応する葉Ｌｊを求める。もしＬjが満杯の時、Ｉのレイヤ管理構造からＬｊを除

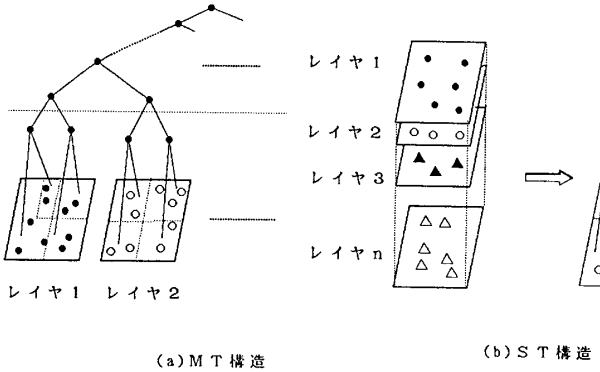


図1 従来のマルチレイヤデータの管理構造

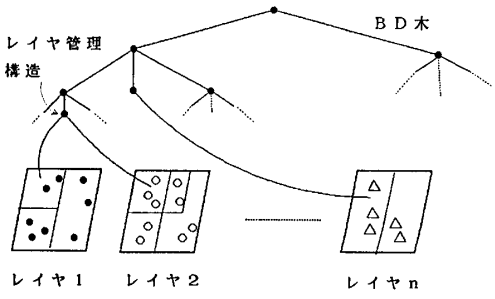


図2 ＭＬ構造

A New Data Structure for Multi-layer Data
Yasuaki NAKAMURA+, Shigeru ABE+, Yutaka OHSAWA++ and Masao SAKAUCHI++
+Mitsubishi Electric Corporation, Central Research Laboratory
++University of Tokyo, Institute of Industrial Science

き、Ｉに分割手続き（NODE_SPLIT）を適用し、終了。
そうでなければ、Ｌｊにデータｄを投入する。終了。
⑤Ｉの左子ノードの領域にｄが含まれるとき、
Ｉ←Ｉの左子ノード
そうでなければ、Ｉ←Ｉの右子ノード
とし、本手続きの②以下を再帰的に適用。

(2)NODE_SPLIT：ノードの分割

ノードＩ中のレイヤｊに対応する葉Ｌｊが満杯かつ、Ｌｊにさらにデータｄが投入されたとき、
①新しい葉Ｌｎを作成。
②Ｉが子ノードを持たないとき、③へ。
そうでないとき、④へ。
③ＢＤ木の領域分割方式[2]によりＩの領域を分割。新しいノードを２個(Ｉ1,Ｉ2)作成し、それぞれに分割領域を対応させる。Ｉ1をＩの左子ノード、Ｉ2を右子ノードとする。次に、Ｌｊ中のデータ、および、ｄのうち、Ｉ1の領域に含まれるデータをＬｊに、Ｉ2の領域に含まれるデータを、Ｌｎに置く。Ｉ1に葉Ｌｊを、Ｉ2にＬｎを付加する。終了。（図３データ６投入）
④Ｌｊ中のデータおよびｄのすべてが、Ｉのどちらかの子ノードの領域に含まれるとき、⑤へ。そうでなければ、Ｉの左子ノードの領域に含まれるデータをＬｊに、右子ノードの領域に含まれるデータをＬｎに置く。Ｌｊ,ＬｎをＩの左右子ノードに追加。終了。
⑤Ｉの左子ノードの領域にすべてのデータが含まれるとき、　Ｉ←Ｉの左子ノード。
そうでなければ、Ｉ←Ｉの右子ノード
とし、本手続きの②以下を再帰的に適用。

(3)Range_search：レイヤ指定の範囲検索

ＭＬ構造の根をroot、指定したレイヤ群を𝕮、検索範囲をＲとし、求めるデータの集合をＳとする。
①　Ｓ　←　空、Ｉ　←　root。
②ＩがNILのとき、return。
③Ｉの葉に𝕮中のレイヤのデータが存在するとき、レイヤｊ∈𝕮のデータが格納されている葉Ｌｊ中のデータのうち、検索範囲Ｒに含まれるものをＳに追加する。return。　そうでなければ、④へ。
④Ｉの左右子ノードのどちらの領域ともＲと共通部分がないとき、return。
⑤Ｉの左子ノードの領域とＲに共通部分があるとき、Ｉ←Ｉの左子ノード　とし、②以下を再帰的に適用。復帰後、⑥へ。
⑥Ｉの右子ノードの領域とＲが共通部分があるとき、Ｉ←Ｉの右子ノード　とし、②以下を再帰的に適用。return。

４．ＭＬ構造の効率試験結果

図４にＳＴ、ＭＴ、ＭＬ構造により範囲検索効率の試験結果を示す。ただし、ケース１は、レイヤ数10、データ数は、各レイヤ　1000個、ケース２は、各レイヤ、3000、2000、1500、1000、750、500、500、250、250、250個とした。(a)は、ケース１を対象とし、複数のレイヤを指定した範囲検索結果を示しており、ＭＬ構造は、複数レイヤが対象の範囲検索において、特殊な場合（１もしくは、９レイヤ以上が対象）以外は、他の手法に優る。(b)は、ケース２を対象とし、データ数の多いレイヤ（第２レイヤ）と少ないレイヤ（第８レイヤ）を１レイヤのみ指定した場合の検索結果である。ＭＬ構造では、データ数の少ないレイヤのデータは、木構造の根に近い位置に存在することが分かる。

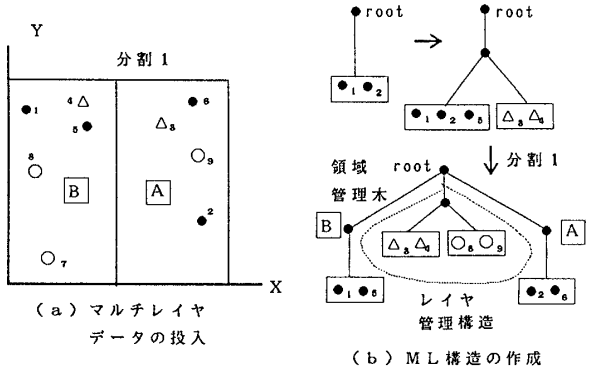
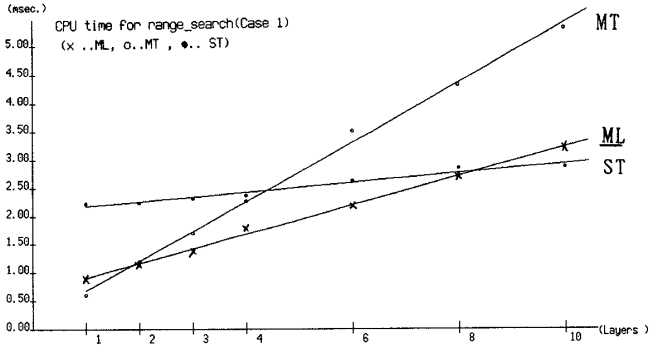
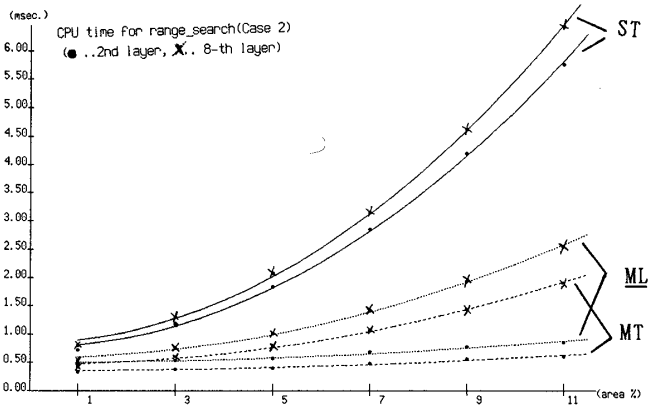


図３　ＭＬ構造の作成例（データ容量Ｐ＝３）



（ａ）複数レイヤを指定した範囲検索結果
（ケース１が対象）



（ｂ）単一レイヤを指定した範囲検索結果
（ケース２が対象）

図４　範囲検索結果(CPU時間、msec.)

５．おわりに

ＭＬ構造におけるレイヤ管理構造としてポインタ配列を用いたが、Ｂ木のような構造でポインタを管理することにより、メモリ量を削減することができる。ＭＬ構造は、対象となるレイヤ数に依存することなく、良好な検索効率が保証できるため、今後、マルチレイヤデータが対象となる各方面への応用が期待される。

参考文献　(1)J.L.Bentley et al.: Data Structures for Range Searching, Comp. Survey, 11,4,1979
(2)大沢、坂内：良好な動特性を持つ多次元点データ管理構造の一提案,信学論(D),J66,10,1983
(3)中村、阿部、大沢、坂内：多次元データの平衡木による管理 -ＭＤ木の提案-,信学論(D)、掲載予定、1988