空間データの効率的管理と高速空間検索のためのデータ構造

大沢　裕　埼玉大学工学部情報システム工学科　ohsawa@mm.ics.saitama-u.ac.jp
中村泰明　広島市立大学情報科学部情報数理学科　nakamura@cs.hiroshima-cu.ac.jp

近年，空間データ（地理情報）が注目を集めている．この空間データを管理し，検索・表示を行うシステムは地理情報システム（GIS: geographic information system）と呼ばれている．コンピュータを用いたGISは，1970年ごろから本格的な研究が進められてきた．GISの応用分野としては，行政における都市計画や環境問題，防災分野への応用，地図作成，ガス・水道，通信網，電力供給網などのファシリティー管理（FM: facility management），出店計画などのマーケティングなどが挙げられる．特に我が国においては，1995年1月の阪神淡路大地震の後，ディジタル地図の防災計画や被災状況調査に果たす役割が注目され，国をあげて基盤データの整備やシステム構築が進められてきた．世界的にも1998年1月の（当時の）米ゴア副大統領の“digital earth”演説などを契機として，各国において地理情報への注目が高まっている．本稿で扱う空間データ管理構造は，主として地図データ管理のためのデータ構造であり，以下空間データを地図情報と同義に扱う．

空間データは本質的に2次元または3次元のデータである．紙地図には，地形や人工構造物，行政界などの領域境界線が2次元平面上に表現されている．これをディジタイザを用いた人手やオートディジタイザを用いてディジタル化を行うと2次元のディジタル地図ができあがる．地形の起伏（標高）や建物や道路などの高さを併せて入力すると3次元のディジタル地図となる．さらに最近では，時間データを含めて扱う場合（この際には時空間情報と呼ばれる）もあり，その際には4次元データを扱うことになる．

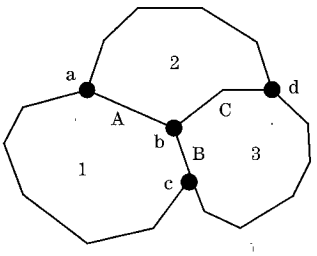
空間データはこのように2次元から4次元程度であり，本特集の別稿で述べられている映像情報処理やパターン認識における特徴分類などで用いられる多次元データに比較すると低次元という特徴を持っている．

空間データの管理と検索

空間データの実体は，点，線，面（領域），立体に分類できる．点は建物の位置のようにマクロに見れば点として扱うことのできる地物が対応する．点データの位置の表現は，単純に(x, y)の座標で行われる．

道路，鉄道，河川，市町村界のような領域境界線，エネルギーや通信，水の供給ネットワークなどさまざまな地物が線として抽象化される．これらの線は折れ線で近似表現される．この折れ線は，両端点と折れ点を表す補間点の合計数およびそれらの点の座標の配列で表されることが多い．

面はシステムや扱う対象の性質により，いくつかの表現法がとられる．まず，建物の外形のように孤立した面は，ポリゴンとして扱われ，両端が同じ座標の折れ線で表現される．一方，市町村界のように，隣り合



Line	FN	TN	LP	RP
A	a	b	2	1
B	b	c	3	1
C	b	d	2	3

図-1　DIME構造

う領域の境界を隔てる線の場合，**図-1**に示すように，線の交差点で分割された折れ線で表現されることが多い．この際には，各折れ線に対してどの領域とどの領域を隔てる線なのかが記述される．古くから空間情報の記述に用いられてきたDIME (dual independent map encoding) [1]では，すべてのノードと折れ線，領域にユニークなID (identifier) を付け，図-1のように表現される．ここでFNは折れ線の始点ノード，TNはその終点ノード，LPは折れ線の左領域，RPはその右領域をそれぞれ表す．また，領域記述をDIMEのように明示的に持たず，必要に応じて空間演算により復元する方式も用いられている．

平面のほかに高さ情報を持つ3次元の地物の表現には，システムに応じてさまざまな方式が採用されている．最も自由にかつ詳細に立体形状を表現する方式は，CADデータと同様にワイヤフレームやボクセルなどで立体形状を記述するものである．しかし，空間情報の場合，非常に広い範囲を扱うこと，3次元データの取得が高価なことなどから，この方式は比較的地理的に狭い範囲を対象とするシステムで用いられている．

空間情報は高さ方向の軸と平面方向の広がりとは若干異質な面がある．ビルなどの地平面からの高さは100m程度である．また地下に作られる構造物についても50m程度の深さである．したがって1つの市町村程度の範囲を考えたとき，高さ軸の変動幅は相対的にわずかな範囲に収まることになる．このため，広い範囲を扱う空間情報システムでは，地形の凹凸をDEM (digital elevation map) として持ち，これに加えて各地物は地表面からの相対的な高さ (比高) で管理されることも多い．

地理情報システムを用いて空間情報を扱う場合に，オブジェクト間のトポロジカルな関係に注目した検索や操作が行われる．地図の編集・修正システムに求められる機能は，地物の属性情報 (たとえば道路名や，幅，交通規制など) の編集を除くとほとんどグラフィックエディタのそれと同様である．

空間データ検索に求められる基本機能として，(1) 完全一致型検索，(2) 範囲検索，(3) 最近接検索，が挙げられる[2]．まず，完全一致型検索とは，ある地物の位置形状が指定され，それに一致するデータベース中の地物を求める検索である．完全一致型検索は主として点データに対して行われることが多い．

次に，範囲検索とはある面 (単純には矩形範囲) が指定され，その矩形内に存在するデータベース中の図形を探査する検索である．空間情報データベースにおいて，表示の際にズームアップを行うとき，データベースから表示している矩形範囲のデータを探査する必要がある．このような場合に，範囲検索が行われる．また，空間情報システムでは，「道路から100m以内の範囲に存在する学校を求める」というタイプの検索も必要になる．このような検索はバッファ検索と呼ばれるが，これも範囲検索である．

最近接検索は，ある点が指定され，その点の最も近くに存在するデータベース中の図形を探査する検索である．空間情報システムは対話的に図形の追加・削除などの編集作業を行う機能を備えている．そのグラフィックエディタである図形を編集する場合，たとえば道路を指定するときには，道路の線上をマウスのボタンでクリックして行うことが多いが，そのような場合に必要となる処理も最近接オブジェクトの検索である．

セル分割による空間データ管理

空間データにおいて最も単純な対象は点データである．まず，点データの管理法について述べ，後半で線や面の管理に拡張する．

空間情報管理では点データを対象としても，前章で述べた各種検索が求められる．完全一致型検索については，空間中に存在するデータを1つの座標軸に沿って1次元に並べることができれば高効率に検索を実行できる．その場合にはソーティングやハッシングなどの方式が適用可能である．

一方，範囲検索や最近接図形の検索にはソーティングは不向きである．ある1つの座標軸に沿ってソーティングされたデータを対象に範囲検索を行う場合，主キーの軸については絞込みの効果が大きいが他の軸については1つずつ指定範囲内に存在するかのチェックを必要とする．

多次元点データの管理法として古くからグリッドフ

ファイル（grid file）（たとえば，文献3））と呼ばれる構造が用いられてきた．これは空間を等間隔のグリッドによりセルに分割し，1つのセルに含まれるデータ同士をまとめて管理する方式である．グリッドファイル上での範囲検索は，指定範囲と重なるセルのデータにアクセスし，1個ずつ真に指定範囲に内包されるデータかのチェックを行うことにより実行できる．

しかし，等間隔のグリッドにより空間を分割する場合，データ分布の粗密により，検索効率が低下する場合が生じる．空間情報システムで管理するデータは，都心部に集中しやすい傾向があり，極端に疎な部分と密な部分ができる場合が多い．このため，次章で述べる階層的な領域分割法が各種提案されている．

線や面同士の重なりを求める場合，計算コストが高いものとなる．たとえば，頂点の数がmとnの折れ線同士の交差を調べようとする場合，2本の線分同士の交差を調べる計算を$m \times n$回実行する必要がある．常にこのような計算を行うのではなく，2つの折れ線が交差している可能性がある場合にのみ交差を調べることにより計算時間を短縮するためには，各折れ線に対して，それを内包する最もタイトな矩形の情報，MBR（minimum bounding rectangle）を持たせておき，MBR同士が重なる線分同士について厳密な交差検査を行う方法がとられる．

図-2は線や面のように空間的な広がりを持つ対象をグリッドファイルで管理する場合を表している．またこの図では，線のMBRのみを表している．ここで，7の図形は，セル（X1, Y1）とセル（X1, Y2）にまたがっており，この図形はそれぞれのセルに登録される．

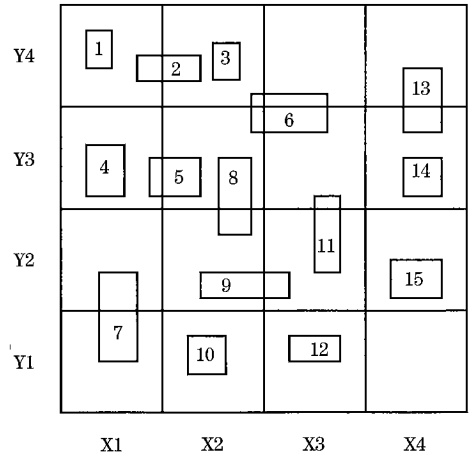


図-2　グリッドファイル

木構造による空間データ管理

前章で述べたグリッド法は，データの粗密があるとき，密にデータが存在する部分で検索性能が低下する問題がある．また，空間的広がりを持つ図形を管理する場合，1つの図形を重複してインデックスに登録しなければならないなどの問題がある．本章では，階層的な木構造によりこれらの問題を解消するデータ構造について述べる．

2^n分割木

階層的木構造で単純なものに2^n分割木[4]がある．これは，2次元平面では4分木となり，3次元空間では8分木となる．

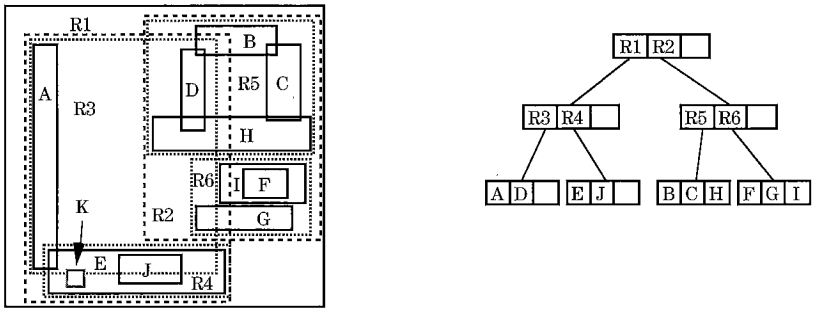
正方形の範囲に点データが分布している場合，その平面を各座標軸で2等分割することにより，4つの部分平面に分割する．もし，各部分平面に存在するデータの数があらかじめ定められた値（M，ページサイズ）を超える場合，その部分平面をさらに縦横2等分割し，4つの部分平面に分割する．この操作を，どの部分平面にもM個以下のデータとなるまで繰り返す．前章で述べたグリッドファイルと比較すると，データ分布の粗密が検索性能の劣化に及ぼす影響を軽減できる．

対象のデータが線や面のように広がりを持つデータの場合には，グリッドファイルと同様にまたがる部分平面に重複して登録する方法もとることができるが，後述のR-treeのように各ノードに対して，そのノードの下に存在しているすべての図形を内包するMBRを付与することにより，1つの図形は1回しか登録されないようにすることもできる．

k-d木

J. L. Bentleyにより提案されたデータ構造[5]である．各座標軸についてデータのソーティングを行い，そのメディアンの値でデータを2つのグループに分割する操作をベースとして，2分木でデータ管理を行う構造である．分割する座標軸については，$x \rightarrow y \rightarrow x \rightarrow y$ ...のように巡回的に変える方法や，分割された空間が正方形に近い形となる分割軸を選択する方法などが提案されている．

範囲検索を行う場合，指定される検索範囲に特別な傾向がない場合には，典型的な検索範囲として円や正方形に近い形が考えられる．したがって，分割された



(a) Rectangles and regions　　(b) R-tree managing rectangles

図-3　R-tree

空間の形については，正方形に近い形になるように分割軸を選択する方式の方が，検索時に訪れる葉ノードの数が少なくなり検索効率が良好となることが知られている[6]．

k-d木は，データに追加削除などの動きがない場合には完全バランス木を構成することができる．しかし，データの追加削除が発生する動的なデータに対しては，バランスが大きく劣化する場合があり，検索性能の低下も招く．

R-tree

k-d木が2分木構造を採用しているのに対し，R-tree[7]は，B木の考え方を多次元に拡張したものである．そのため，R-treeはすべての葉ノードのレベルが同じになる完全平衡木である．R-treeは，長方形データの管理用に考案され，その後，効率向上を目指したいくつかの改良がほどこされている．ここでは，最初に提案されたR-tree構造を紹介する．

R-treeは，長方形データを格納する葉と領域を管理する内部ノードから構成される．内部ノードは，Cm個以上，CM個以下の子ノードを持つ．葉は，Lm個以上LM個以下の長方形データを管理する．一般に，Cm，Lmは，それぞれ，CM，LMの30～40%に設定される．

図-3のように，R-treeの葉と内部ノードは，そのノード以下の部分木中に存在するすべてのデータを包含するMBRを管理領域とする．図ではCM，LMが共に3の場合を示している．あるノードvを根とする部分木にデータxを追加する際には，vの子ノードの中から，データxをその子ノードを根とする部分木に追加した場合に，管理領域の拡張が最小となる子ノードを選択し，その子ノードを根とする部分木にデータを追加する．

たとえば，図-3に長方形Kを追加する場合，Kは，R1に完全に含まれるので，R1の子ノードが選択される．さらに，KはR4に含まれるため，R4の子ノードが選択され，R4が管理する葉にデータが追加される．葉のデータ数がLM個を超えるときには，葉の分割が行われる．葉の分割では，2つの葉が作成され，LM+1個のデータが2つの葉に分けられる．ただし，両方の葉とも，Lm個以上のデータを含むものとする．

データの分割方式には，線形オーダ，2乗オーダアルゴリズムなどの方法が提案されているが，いずれにしても，領域の重なりが少なく，かつ，領域を正方形に近くなるように長方形データを二分する．内部ノードの分割も基本的には葉の場合と同じ方式を用いる．R-treeによる範囲検索では，検索範囲とノードの管理領域とが共通部分を持つすべてのノードを辿り，到達した葉内のデータから検索領域と重なりを持つものを求めることになる．

R-treeでは管理領域のオーバーラップを許している．そのため，データを追加する部分木の選択やノード分割のアルゴリズムにフレキシビリティがあり，いくつかの改良案が提案されている．また，効率の向上を目指し，分割時にデータを再投入する方式も提案されている[13]．しかし，R-treeで採用される追加アルゴリズムは，k-d木やMD木構造に比べ複雑であり，データ構造の構築に，時間を要する．また，最低ページ効率は，Lm/LMを保証するが，MD木（66.6%）やk-d木（50%）に比べ低い．最低ページ効率を上げた場合，領域分割の最適化が難しくなり，検索効率が低下することが知られている．実際の環境では，Lm/LM=0.3程度が適当とされている[12]．

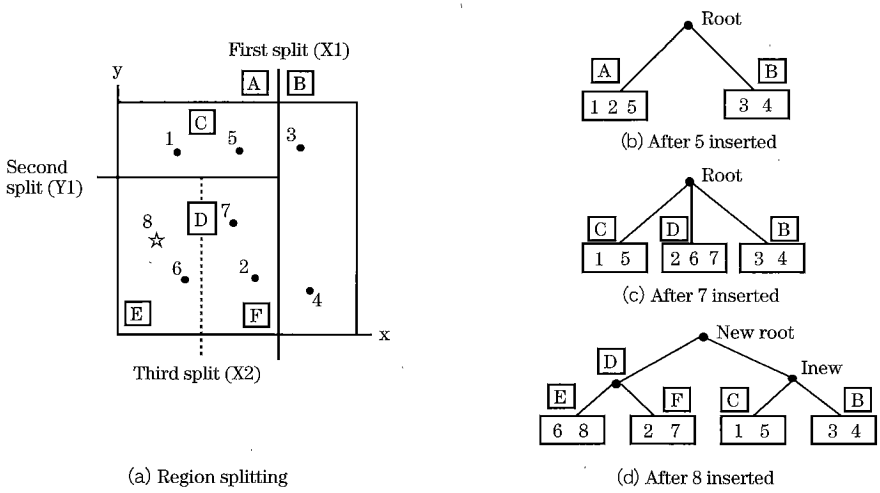


図-4　MD木

MD木

MD木（Multi-Dimensional tree）[10), 11)]は，メモリ効率と木のバランス性向上を図った多次元データ構造である．MD木は，1次元データの管理構造である2－3木を多次元に拡張したものであり，R-treeやkdb木[9)]と同等に完全平衡木が構成できる．さらに，MD木では，最悪時でも2/3（≒67%）のページ効率が保証できることを特長とする．シミュレーションによれば，MD木では，データの投入順序や分布によらずページ効率（メモリ効率）は80%以上となることが示されている．

MD木では，葉は分割された部分領域に対応し，内部ノードは，そのノード以下の部分木中のすべての領域を含むMBRを管理する．**図-4**にMD木の成長過程を示す．まず，図-4の（b）までの過程を説明する．データ1の投入により，最初の葉Lrが作成される．データ2，3は，Lrに格納される．データ4が投入されると，Lrが分割される．その結果，全領域は1回目の分割軸X1により左右に分割され，新しい葉が作成される．データ5の投入では，分割は起こらず（b）の木となる．さらにデータ6が投入されると（b）の左側の葉が分割され，分割軸Y1によりその領域（左半平面）は領域CとDに分けられる．その結果，（c）の3分木となる（ただし，（c）はその後，データ7が投入された状態）．さらに，データ8が投入されると，中央の葉（領域C）が分割軸X2により，領域DとEに分割される．引き続き，rootが分割され，新しいrootと内部ノードInewが作成され，（d）の木となる．

MD木では，葉の領域の重なりは認めていないが，ノードの領域の重なりは認めている．kdb木では内部ノードの分割が起こると，そのノード以下の子ノードに対しても，ノードの管理領域が分割軸と交差する場合には再帰的に分割が適用されるため，多数のノードの分割が行われることがある．その結果，kdb木のような無駄な領域分割は行われず，最低ページ効率も保証可能である．

MD木では，葉のデータ数が容量を超える場合には，B木と同様に兄弟の葉での動的な調整により，オーバーフローの吸収処理を行い，吸収が不可能な場合にのみ葉の分割が行われる．葉の分割手順は，k-d木と同様である．検索アルゴリズムは，他のデータ構造の場合と同様である．MD木で線分や長方形を管理する方式として，GBD木[8)]で適用された重心とMBRを利用する方式が応用可能である．

その他の木構造

kdb木[9)]はk-d木と同様な空間分割方式で，多分木に拡張したデータ構造である．Range木[15)]は範囲検索効率の向上が図られている．R-treeなどでの範囲検索効率が$O(N^{1/2}+k)$であるのに対して，Range木では$O(log^2N+k)$が保証されている（ただし，Nはデータ数，kは検索結果のデータ数）．

R-treeがデータの挿入削除と検索共にMBRを参照して行うのに対して，GBD木[8)]ではデータの挿入削除については領域式というビット表現を用いている．これにより，データの追加削除の効率を向上させている．

時空間データ管理構造

一般の空間データ管理においては，常に最新の状態のデータに対応したデータ構造のみが保持される．しかし，アプリケーションによっては，過去の状態のデータセットへのアクセスが必要になる．たとえば，都市空間データ管理においては，x年○月△日における上水配管を表示する，あるいは，ある時間範囲内に設置（廃棄）された設備を求めるといった検索が必要になる．このような時間範囲と空間範囲を指定した検索（以下，時空間検索と呼ぶ）を実現するためには，過去の状態のデータセットも保持する必要がある．

中村らは，MD木を用いた過去の任意の時刻でのデータ検索を可能とするデータ構造PMD木[16),17)]を開発している．PMD木では，データ追加，削除に伴う木構造の変化（差分）を管理するためにMD木を拡張した．PMD木では，各時刻における有効データの管理と任意時刻のデータ構造へのアクセスを可能にする．しかも，どの時点においても，データ構造は完全にバランスしており，さらに，MD木の良好な性質であるメモリ効率（最大データ容量に対する実データの割合）が66.6%以上という性質も保持している．データの追加のみのケースでは，データの時系列的な変化を保存するために必要な記憶容量は，最新の状態のみを保持するデータ構造に要するメモリ量をS(N)とした場合，kS(N)（kは定数で，k≦5/2，Nはデータ総数）であるという性質を持つ．

図-5はPMD木の構造を示している．PMD木の内部ノードは，最大4個の子ノードを持つ．ただし，最終の内部ノードは，最大3個の葉を管理する．PMD木の葉には，現在時刻で存在しているデータとすでに削除されているデータが存在する．以下，前者を現存データ，後者を削除データと呼ぶ．ノードは，過去の状態を保存した保存ノードと現在の状態を保持した現存ノードからなる．図-5中ではそれぞれ，黒塗り長方形（もしくは，円），白抜き長方形（もしくは，円）で表されている．各ノードは，そのノードが有効な期間の情報を持つ．ただし，－は現在時刻を表し，現在時刻は時刻とともに進行する．内部ノードは，初期状態（葉の総数が3以下のとき）を除き，2個以上最大4個の子ノードを持つ．ただし，子ノードの半数以上が保存ノードのとき，そのノードは保存ノードである．各時間間隔におけるPMD木の根は，根管理テーブルにより，存続期間とともに管理される．

PMD木へのデータ追加，およびノード分割は，MD木と同様であり，ノードの分割時の処理のみ異なる．

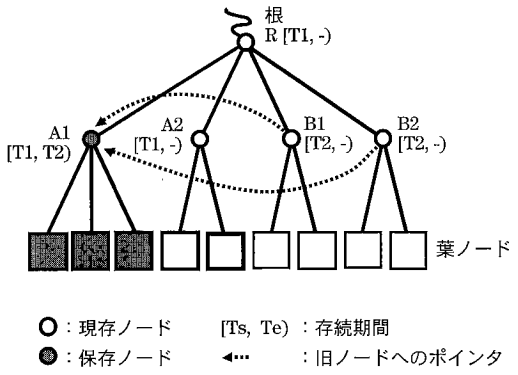


図-5　PMD木の構造

ノードvが分割されるとき，vは保存ノードとなる．新規に作成された2つのノードがvの親ノード(p)の子として追加される．もしpの子ノード数が4を超えるとき，pも分割され，保存ノードとなる．ただし，vが葉のときは，MD木の通常の分割を行い，pの子ノード数が4個になる場合，pが分割され，pとpの子ノードが保存ノードとなる．以上を繰り返すことにより，PMD木が構成される．

図-6にPMD木の構成例を示す．時刻T6で，右端のI4ノード以下，ノードE2にデータが追加されたとき，葉の分割が起こり，その結果，E2が分割され，新しくノードF1とF2が構成された様子を示す（E2は保存ノードとなっている）．PMD木の各ノードは有効時間を持ち，すべての時刻のMD木の状態が保持されている．PMD木による検索では，時刻を指定することでその時刻に有効であったMD木の根を探索し（一般に空間検索に比べ十分短い時間で実行できる），そのMD木により空間検索が行われる．すなわち，時刻と空間範囲を指定した検索が可能であり，しかも，検索に要する時間は，PMD木中のデータ総数には依存せず，その時刻に有効であったMD木の探索時間と同等であることから，高速な時空間検索が可能である．

空間データ構造の応用

空間情報システムが扱うデータは2次元または3次元の比較的低次元のデータである．空間情報は現実世界のディジタルデータであり，応用分野は広い．汎用的な数値地図としては，最近では1/2500程度の高い精度

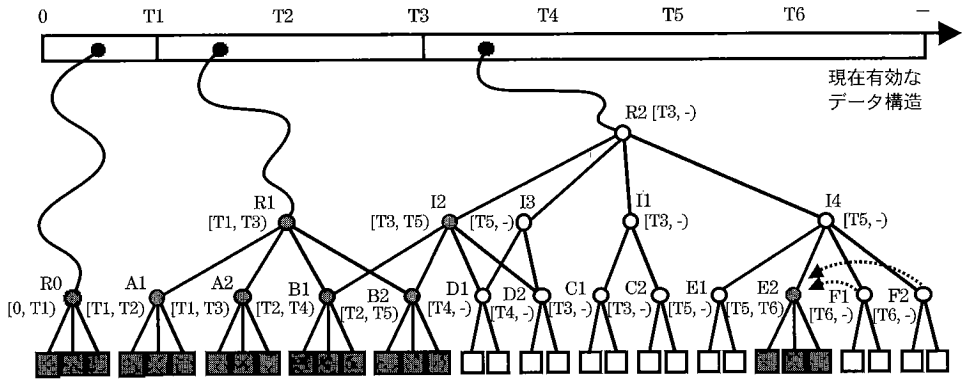


図-6　PMD木の構築例

を持つディジタルデータの整備が政府機関により進められている．また，ITS（高度交通システム），特にカーナビのためのデータとして道路網のディジタル地図も高精度化，詳細化してきている．さらに，住宅地図のディジタル化と保守も民間ベースで進められている．さまざまな対象の空間情報を活用できる環境が整いつつある．

これら整備が進められているディジタル地図の多くは，2次元平面上に存在する地物を2次元的にディジタル化したデータである．地形の標高については，DEM（digital elevation map）がメッシュデータとして整いつつあるが，建物，道路，橋，さらには地下街などを3次元的に捉えたデータの整備は非常にコストの高いものとなることから，地域を限定したものにとどまっている．航空写真による3次元計測や，レーダを用いた3次元計測技術の進歩は目覚しいが，物体に見えない場所が生じるオクルージョン（occlusion）の問題があり，さらなる研究を必要としている．

本稿で対象とした2～3次元のデータ管理では，グリッド法や各種階層的な木構造によるデータ管理構造で効率よいデータ管理を行うことができる．特に，最近のコンピュータの処理速度の向上により，空間データ管理については大きな障害はなくなった．

高さ情報に加えて，時間情報も空間情報には重要な情報である．土地の登記や水道や道路などの設備管理において，時空間情報管理が求められている．空間情報と時間情報とはデータの持つ性質，利用法についてまったく異質なものである感がある．時空間情報システムにおいて，時間をどのように管理すべきかについては，さらなる研究を要していると考える．

参考文献
1) Memers, M. N.: Fundamentals of Geographic Information Systems (second ed.), John Wiley & Sons (2000).
2) 坂内正夫, 大沢　裕: 画像データベース, 昭晃堂 (1987).
3) Burrough, P. A. and McDonnell, R. A.: Principles of Geographical Information Systems, Oxford (1998).
4) Finkel, R. A. and Bentley, J. L.: Quad Trees: A Data Structure for Retrieval on COmposite Keys, Acta Informatica, 4, pp.1-9 (1974).
5) Bentley, J. L.: Multidimensional Binary Search Trees Used for Associated Searching, Comm. of ACM, 18, pp.509-517 (1975).
6) 松山他: 空間的近接性に基づくファイル分割アルゴリズムの性能評価, 信学技報, IE81-14 (1981).
7) Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching, Proc. ACM SIGMOD Intl. Symp. on the Management of Data, pp.45-57 (1984).
8) 大沢　裕, 坂内正夫: 2種類の補助情報により検索と管理性能の向上を図った多次元データ構造の提案, 電子情報通信学会論文誌D-I, J74-D-I, 8, pp.467-475 (1991).
9) Robinson, J. T.: The K-D-B-tree: A Search Structure for Large Multidimensional Dynamic Indexes, Proc. ACM SIGMOD, pp.10-18 (1981).
10) 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), J71-D, 9, pp.1745-1752 (1988).
11) Nakamura, Y., Abe, S., Ohsawa, Y. and Sakauchi, M.: A Balanced Hierarchical Data Structure for Multidimensional Data with Efficient Dynamic Characteristics, IEEE Trans. KDE, Vol.5, No.4, pp.682-694 (1993).
12) Beckmann, N., Kriegel, H. P., Schneider, R. and Seeger, B.: The R*-tree: An Efficient and Robust Access Method for Points and Rectangles, Proc. SIGMOD, pp.322-331 (1990).
13) 中村, 阿部, 大沢, 坂内: 空間的広がりを持つ図形データのMD木による管理, 電子情報通信学会論文誌D-II, Vol.J73-D-Ⅱ, No.12, pp.1976-1984 (1990).
14) Bentley, J. L. and Friedman, J. H.: Data Structures for Range Searching, Computing Surveys, 11, 4, pp.397-409 (1979).
15) Bentley, J. L.: Decomposable Searching Problems, Inform. Process. Lett., Vol.8, No.5, pp.244-251 (1979).
16) Nakamura, Y.: PMD-TREE: A Balanced Hierarchical Data Structure for Spatio-Temporal Objects, Proc. 13th Int. Conf. Computers and Their Applications, pp.282-286 (1998).
17) 中村, 出木原: 時間属性をもった空間データの管理構造－PMD木－, 情報処理学会論文集, Vol.40, No.SIG 5 (TOD2), pp.54-68 (1999).
（平成13年8月10日受付）