# 組込DBMSにおける空間索引技術の検討

林　秀樹† 谷崎　正明† 木村　耕治†† 梶山　尚紀††† 入江　正憲†††

† 株式会社 日立製作所 中央研究所　〒185–8601 東京都国分寺市東恋ヶ窪 1–280
†† 株式会社 日立製作所 ソフトウェア事業部　〒244–8555 横浜市戸塚区戸塚町 5030
††† 日立ソフトウェアエンジニアリング株式会社　〒244–8555 横浜市戸塚区戸塚町 5030
E-mail: †{hideki.hayashi.xu,masaaki.tanizaki.tj,kohji.kimura.zn}@hitachi.com,
††{h-kajiyama,m-irie}@hitachisoft.jp

**あらまし**　本報告では，車載情報端末を適用先とする組込 DBMS において，形状が線や面の空間オブジェクトを対象とした空間索引技術を提案する．この提案に際し，車載情報端末でディスク容量の制約が厳しいと考え，空間索引の格納に必要なディスク容量の削減を課題とした．そこで，空間オブジェクトの管理で優位とされる PMR 四分木を参考にして，索引木を提案する．シミュレーション実験の結果より，提案索引木は，従来索引木よりディスク容量が小さくなることを確認した．

**キーワード**　組込みデータベース，空間データベース

Hideki HAYASHI†, Masaaki TANIZAKI†, Kohji KIMURA††, Hisanori KAJIYAMA†††, and Masanori IRIE†††

† Hitachi, Ltd., Central Research Laboratory　1–280 Higashi-koigakubo Kokubunji, Tokyo, 185–8601 Japan
†† Hitachi, Ltd., Software Division, 5030 Totsuka-cho, Totsuka, Yokohama, 244–8555 Japan
††† Hitachi, Software Engineering Co., Ltd.,　5030 Totsuka-cho, Totsuka, Yokohama, 244–8555 Japan
E-mail: †{hideki.hayashi.xu,masaaki.tanizaki.tj,kohji.kimura.zn}@hitachi.com,
††{h-kajiyama,m-irie}@hitachisoft.jp

## 1. はじめに

近年，モバイル端末に搭載可能な数ギガバイトの小型のフラッシュメモリが登場している．また，大容量の HDD(Hard Disk Drive) を搭載した車載情報端末や情報家電が存在している．このように，組込機器で管理すべきデータ量が急増し，データの多様化が進んでいる．それに伴い，組込ソフトウェアに，高度なデータ管理機能が要求されている．組込ソフトウェアの開発行数は，大規模なシステムで，500 万行以上になる [1]．その約半分は，データ管理機能であると言われている．組込ソフトウェアの開発サイクルが短縮される現状では，開発工数の削減は急務である．この課題を解決するため，組込機器向けのデータベース管理システム (DBMS) が製品化されている [2]〜[4]．組込ソフトウェアの開発者が DBMS（以下，組込 DBMS と呼ぶ）を利用することで，データ管理機能の開発に要する工数を削減できる．

組込 DBMS の適用先は，件数が多く，多種類のデータを管理する端末となる．また，データの参照のみではなく，更新も必要とされる端末となる．その代表例は，大規模な地図データベースをもつ車載情報端末である．車載情報端末では，地図描画，施設検索，経路探索などが要素機能となり，地図データベースから所望の地図データを高速に検索することが要求される．また，車載情報端末に書き換え可能な HDD が搭載されたことや，車載情報端末と携帯電話網が接続可能になったことで，車載情報端末上での地図データの更新が重要視されてきている．

これまでに報告者らは，車載情報端末への組込 DBMS の導入を想定し，施設検索の支援を目的とした空間検索を実現した [5]．実現した空間検索は，形状が点の空間オブジェクトを検索対象とし，組込機器の処理時間とメモリ使用量の制約を考慮した範囲検索と k 最近傍検索である．範囲検索は，任意の領域に含まれる空間オブジェクトを検索する．例えば,「現在地から 500 メートル以内のガソリンスタンド」という検索に用いられる．k 最近傍検索は，任意の地点から距離の近い上位 $k$ 件の空間オブジェクトを検索する．例えば,「現在地から距離の近い上位 10 件の駐車場」という検索に用いられる．報告者らは，(株) 日立製作所と日立ソフトウェアエンジニアリング (株) で共同開発した組込 DBMS 製品 Entier [2] に空間検索を実装し，その有効性を確認した．

文献 [5] では，空間検索の処理時間を短縮するため，四分木という空間索引を採用している．四分木では，空間オブジェクトの存在密度が高い領域ほど細かい矩形（分割矩形）に分割し，分割矩形とその矩形内の空間オブジェクトを対応付ける．空間オブジェクトは，自身を包含する分割矩形に対応付けられる．四分木を用いることで，空間検索で指定する場所付近の分割矩形に対応付けられた空間オブジェクトのみ参照すれば良い．車載情報端末では，通常，空間オブジェクトはディスクに格納されている．参照する空間オブジェクトの件数が減れば，ディスクからメモリに読み込む処理（ディスクアクセス）の回数が減り，検索時間を短縮できる．この効果は，近年普及が進む PND(Personal Navigation Device) のようなフラッシュメモリドライブ搭載の車載情報端末でも有効である．

ここで，組込 DBMS の空間検索を用いて，施設検索と同様に重要な要素機能である地図描画を支援することを検討する．地図描画では，描画範囲と縮尺を条件に，地図データベースから空間オブジェクトを高速に検索する必要がある．例えば，現在地や目的地の周辺の地図を画面に描画することが挙げられる．この空間オブジェクトは，道路，鉄道，河川などの形状が線の空間オブジェクト，建物，公園，行政界などの形状が面の空間オブジェクトとなる．つまり，地図描画を支援するためには，空間的な広がりをもつ空間オブジェクトを検索対象とした範囲検索を実現する必要がある．この範囲検索は，矩形範囲を指定し，矩形範囲に交差する空間オブジェクトを取得する機能を指す．文献 [5] の空間検索では，線や面の空間オブジェクトを検索対象にできない．これは，四分木が，線や面の空間オブジェクトを管理できる設計になっていないためである．

そこで，本報告では，形状が線や面の空間オブジェクトを管理可能な組込 DBMS における空間索引を提案する．この提案に際し，まず，車載情報端末では，ディスク容量の制約が厳しいと考え，空間索引の格納に必要なディスク容量の削減を課題とする．空間索引の格納に必要なディスク容量は，分割矩形と空間オブジェクトの対応関係を木構造で管理する索引木のデータ構造に依存する．そこで，これまでと同様に空間索引に四分木を採用し，従来索引木よりディスク容量の小さい索引木を提案する．提案索引木は，空間オブジェクトの管理で優れた性能をもつ PMR 四分木の索引木に基づく．また，範囲検索を高速化するため，提案索引木の効率的な探索アルゴリズムを考案する．そして，シミュレーション実験により，提案索引木が従来索引木に比べて，索引木の格納に必要なディスク容量と，範囲検索の処理に伴うディスクアクセス回数の面で，優位であることを確認する．

本報告の構成は以下の通りである．2 章で四分木について説明する．3 章で従来索引木について説明し，4 章で提案索引木について説明する．5 章で性能評価の結果と考察を示す．最後に 6 章で本報告のまとめを述べる．

## 2. 四分木による空間オブジェクトの管理

本章では，索引木に依存しない四分木による空間オブジェクトの管理について述べる．まず，四分木を用いた空間の分割方法について述べる．次に，索引木内の空間オブジェクトの表現方法について説明する．また，四分木を用いた範囲検索の処理について説明する．

### 2.1 空間の分割方法

四分木では，分割矩形と空間オブジェクトを対応付け，矩形範囲に交差する分割矩形の空間オブジェクトのみを参照することで，検索時間を短縮する．実世界では，空間オブジェクトの密度が場所により異なるが，矩形範囲の場所によって検索時間に大きな差が出ないことが要求される．この課題に対し，四分木では，分割矩形に対応付けられる空間オブジェクトの件数が分割矩形の間で大きな差にならないように，平面空間を分割する．また，空間オブジェクトは，自身に交差する分割矩形に対応付けられる．そのため，分割矩形が交差可能な空間オブジェクトの最大の件数を決めておき，その件数を超えた分割矩形は，等しい面積で四つの矩形に分割される．

### 2.2 索引木内の空間オブジェクトの表現方法

索引木では，空間オブジェクトを外接矩形で管理する．外接矩形とは，空間オブジェクトを包含する最小の矩形を指す．外接矩形を用いる利点は，以下の通りである．

- 分割矩形と空間オブジェクトの対応付けに，分割矩形と空間オブジェクトの交差判定の処理が必要になる．空間オブジェクトの外接矩形で交差判定を行うことで，分割矩形と空間オブジェクトの正確な対応付けはできないが，交差判定の処理の負荷を軽減できる．
- 外接矩形は，通常，左下（最小）座標と右上（最大）座標からなる固定長で表現される．そのため，外接矩形の管理は，任意の個数の座標列で表現される実形状の管理に比べ，容易である．

### 2.3 範囲検索の処理

線や面の空間オブジェクトを対象とする範囲検索は，次の手順で処理される．矩形範囲に交差する空間オブジェクトは，概略交差判定と詳細交差判定の二段階の処理で取得される．

（1） 条件設定：矩形範囲を設定する．

（2） 概略交差判定：空間オブジェクトの単純な近似図形と矩形範囲との間で交差判定を行い，範囲検索の結果の候補を求める．具体的には，矩形範囲に交差する分割矩形を探し，その分割矩形に交差する空間オブジェクトの外接矩形と矩形範囲で交差判定を行う．この処理を効率化するため，分割矩形と空間オブジェクトの対応付けを木構造で管理する索引木を用いる．本処理は，後述の詳細交差判定の前に，候補となる空間オブジェクトを絞り込むことで，検索処理を高速化する意図がある．

（3） 詳細交差判定：概略交差判定で求めた結果の候補の中から，正確な結果となる空間オブジェクトを求める．具体的には，結果の候補となった空間オブジェクトの実形状と矩形範囲の交差を判定する．

## 3. 四分木の従来の索引木

本章では，線や面の空間オブジェクトを管理可能な四分木の従来の索引木について説明する．まず，文献 [5] の索引木に対し，線や面の空間オブジェクトも管理可能にした索引木について説明する．この索引木は，分割矩形の親子関係をポインタで接続するポインタベース四分木の一種である．本報告では，文

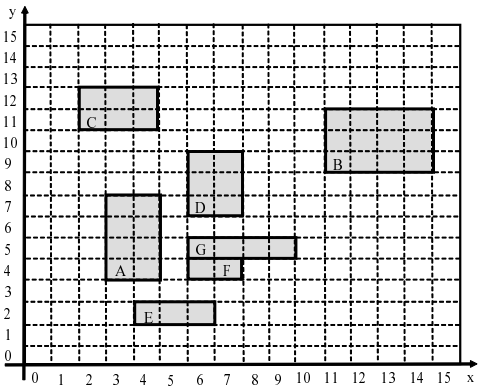


図 1　空間オブジェクトの配置例

表 1　空間オブジェクトの外接矩形

| 空間オブジェクト | 左下座標 | 右上座標 |
|---|---|---|
| A | (3,4) | (4,7) |
| B | (11,9) | (14,11) |
| C | (2,11) | (4,12) |
| D | (6,7) | (7,9) |
| E | (4,2) | (6,2) |
| F | (6,4) | (7,4) |
| G | (6,5) | (9,5) |

献 [5] の索引木を単にポインタベース四分木と呼ぶことにする．また，提案索引木のもとになる PMR 四分木について説明する．

### 3.1　ポインタベース四分木

まず，ポインタベース四分木の索引木のデータ構造について述べる．次に，範囲検索の処理に用いる索引木の探索アルゴリズムについて説明する．

#### 3.1.1　索引木のデータ構造

詳細な説明の前に，ポインタベース四分木を例示する．図 1 は，16×16 の空間における空間オブジェクトの配置例を示す．図中の灰色矩形は空間オブジェクトの外接矩形（表 1）を示す．図 1 では，破線で囲まれた 1×1 の区画を 1 座標としている．例えば，空間オブジェクト A の外接矩形の左下座標と右上座標はそれぞれ (3,4) と (4,7) となる．図 2 は，図 1 の空間オブジェクトをアルファベット順に挿入したときのポインタベース四分木による空間分割を示す．点線は，分割矩形で交差可能な空間オブジェクトの件数が最大件数を超える場合に実施される分割の境界線を示す．図 3 は，図 2 に対応するポインタベース四分木の索引木を示す．非リーフノードは，分割矩形と 1 対 1 の対応関係になっている．図中の非リーフノードの座標は，左が分割矩形の左下座標，右が分割矩形の右上座標を示す．一方，リーフノードには，空間オブジェクトの情報が格納されている．図 3 では，リーフノードに格納できる空間オブジェクトの情報を最大 3 個としている．これは，分割矩形が 3 件より多い空間オブジェクトと交差すると，四分割されることを意味する．

以下では，ポインタベース四分木を詳細に説明する．この索引木では，分割矩形の情報を含む非リーフノードと，空間オブジェクトの情報を含むリーフノードから構成される．非リーフノードには，分割矩形の範囲，下位の非リーフノードへのポインタが含まれる．下位の非リーフノードとは，自ノードに対応する分割矩形を四分割してできた分割矩形を指す．リーフノードは，分割矩形を含まない分割矩形と一対一の関係にある．リーフノードでは，空間オブジェクトの外接矩形，種別を示す属性情報，実データのディスク上の格納位置を示すポインタを要素とする．この索引木内の座標値は，文献 [5] の索引木に従い，32 ビットの整数型とする．

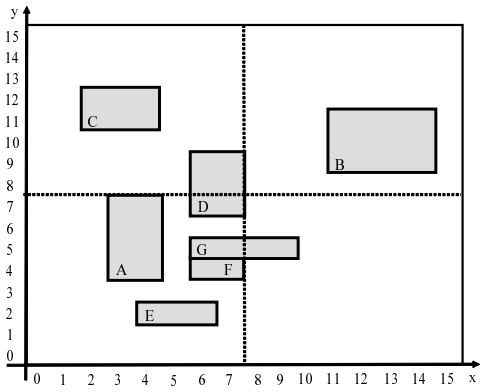


図 2　ポインタベース四分木による空間分割

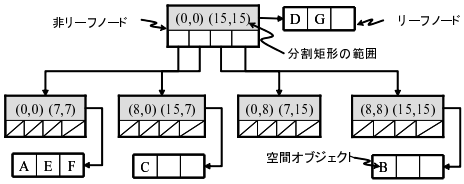


図 3　ポインタベース四分木の索引木

ここで，形状が線や面の空間オブジェクトは，複数の分割矩形に交差する可能性がある．この場合，空間オブジェクトの情報を，(1) 空間オブジェクトの外接矩形に交差する最下位の分割領域に対応する複数のリーフノードに格納する方法と，(2) 空間オブジェクトの外接矩形を内包する最小の分割矩形に対応した一つのリーフノードに格納する方法がある．(1) の方法は，空間オブジェクトの情報を索引木に重複して格納するため，(2) の方法より，ディスク容量が増大する．一方，(2) の方法では，矩形範囲に交差する最上位から最下位の全ての分割矩形で交差判定を行う必要があるため，範囲検索の処理時間が増大する．本研究では，ディスク容量の削減を課題としていることから，(1) の方法でリーフノードに空間オブジェクトの情報を格納する．

索引木の非リーフノードとリーフノードは，ページ単位でディスクに格納される．ページとは，データベースの処理において，ディスクとメモリの間で入出力処理を行う最小単位である．非リーフノード用のページには，分割矩形の親子関係を保つ複数段の非リーフノードの集合を 1 グループとし，複数のグループが，ページの容量の限り格納される．一方，リーフノード用のページには，一つのリーフノードが格納される．空間オブジェクトの挿入でリーフノードの容量を超えると，リーフノードに対応する分割矩形を四分割し，四つのリーフノードを新しく生成する．このように，分割矩形で交差可能な空間オブジェクトの最大件数は，ページのサイズで決定される．

#### 3.1.2　索引木の探索アルゴリズム

まず，矩形範囲に交差する分割矩形を求めるため，索引木の

ルートノードを始点とし，矩形範囲に交差する分割矩形に対応する非リーフノードを辿り，リーフノードまで到達する．非リーフノードは分割矩形の範囲をもつため，索引木の探索中に矩形範囲と分割矩形の交差判定を行える．この交差判定は，矩形範囲のいずれか一辺が分割矩形に交差することを確認する．次に，矩形範囲に交差する空間オブジェクトの外接矩形を求めるため，リーフノード内の空間オブジェクトの情報を参照し，矩形範囲と空間オブジェクトの交差関係を判定する．以上の処理を，矩形範囲に交差する全ての分割矩形に対応するリーフノードで実施することで，概略交差判定を実現できる．

### 3.2 PMR 四分木

PMR 四分木の索引木では，ディスク容量の削減のため，最下位の分割矩形のみを管理する．また，範囲検索の処理で効率的に分割矩形や空間オブジェクトを探索できるように，分割矩形を一意の整数値で符号化し，その整数値をキー値として，B 木などの一次元データの索引木で管理する．本節では，まず，分割矩形の符号化に用いるモートン符号値 [7] について説明する．次に，索引木のデータ構造について述べる．そして，範囲検索時の索引木の探索アルゴリズムについて説明する．

### 3.3 モートン符号値を用いた分割矩形の符号化

モートン符号値は，多次元の座標を一次元の整数値で表現するもので，空間の四隅のいずれかを始点とし，空間を Z 字で一筆書きに辿る順位に従う．本報告では，原点を始点として，x 方向を優先的に Z 字で辿る場合を想定する．以下に，範囲検索で有用なモートン符号値の特徴を示す．

- 空間内に任意の矩形を指定すると，矩形内の座標に対応するモートン符号値が，矩形の左下座標と右上座標に対応するモートン符号値の範囲に必ず含まれる．
- 空間内で近い距離にある座標間では，モートン符号値が近い値となる．

一つ目の特徴は，矩形範囲を一次元の範囲に変換できることを意味する．具体的には，矩形の左下座標に対応するモートン符号値を最小値，右上座標に対応するモートン符号値を最大値とする一次元の範囲となる．また，二つ目の特徴は，矩形範囲を一次元の範囲に変換したとき，一次元の範囲の幅が小さくなる傾向にあることを意味する．この特徴により，索引木の探索部分が狭くなり，範囲検索の高速化に寄与する．

PMR 四分木では，モートン符号値を用いて，分割矩形を一意に識別可能な整数値に変換する．具体的には，分割矩形の左下座標に対応するモートン符号値を用いる．以下に，座標を (x,y) とし，モートン符号値への変換手順を示す．

（1） x 座標と y 座標を 2 進数に変換する．

（2） x 座標と y 座標の各桁のビットを交互に並べる．

（3） (2) で並べたビット列を 10 進数に変換する．

例えば，座標が (7,10) の場合，手順 (1) で (x,y)=(0111,1010) になる．次に，手順 (2) で 10011101 となる．そして，手順 (3) で 10011101 が 157 となり，157 がモートン符号値となる．モートン符号値を用いる場合，モートン符号値の表現に必要なビット数を決める必要がある．本報告では，簡易な方法でモートン符号値のビット数を決めることにする．具体的には，空間を正方形とみなし，x 座標と y 座標のそれぞれの数値の表現に最低限必要なビット数の和とする．

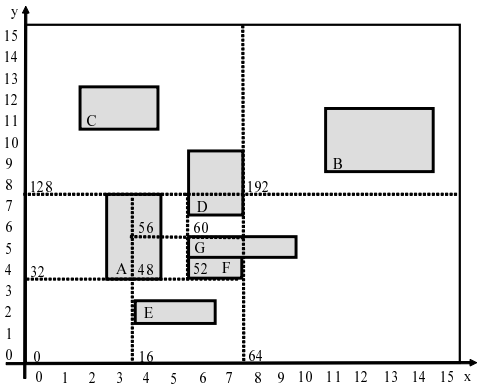


図 4 PMR 四分木による空間分割

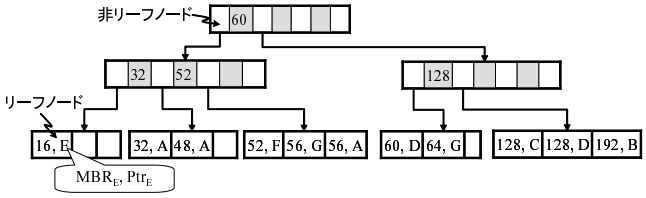


図 5 PMR 四分木の索引木

### 3.4 索引木のデータ構造

PMR 四分木では，範囲検索の処理で効率的に分割矩形や空間オブジェクトを探索できるように，分割矩形に対応するモートン符号値をキー値とし，一次元データを対象とする索引木を用いる．一次元データの索引木であれば制約はないが，文献 [6] では B 木を採用している．本報告でも，PMR 四分木の索引木に B 木を採用し，PMR 四分木の説明や性能評価を行う．

PMR 四分木の索引木は，分割矩形の情報をもつ非リーフノードと，空間オブジェクトの情報をもつリーフノードから構成される．非リーフノードとリーフノードは，それぞれ 1 ページの単位でディスクに格納される．非リーフノードには，i 個のキー値と i+1 個の下位ノードへのポインタが格納される．キー値は，分割矩形に対応するモートン符号値となる．下位ノードへのポインタは，下位ノードを含むページのディスク上の格納位置を示す．一方，リーフノードには，分割矩形に対応するキー値，空間オブジェクトの外接矩形，実データのディスク上の格納位置を示すポインタを要素として，リーフノードの容量の限り，複数の要素が格納される．

PMR 四分木の索引木を例示する．図 4 は，図 1 の空間オブジェクトをアルファベット順に挿入したときの PMR 四分木による空間分割を示す．分割矩形の左下座標の整数値は，分割矩形の符号値を示す．また，分割矩形が最大 3 件の空間オブジェクトに交差できるものとする．また，図 5 は，図 4 に対応する PMR 四分木の索引木を示す．図 5 では，非リーフノードに最大 3 個のキー値を，リーフノードに最大 3 個の空間オブジェクトの情報を格納できるものとしている．また，リーフノード内の空間オブジェクト E を指す吹き出しは，空間オブジェクトの情報を示す．吹き出し内の MBR(Minimum Bounding Rectangle) は外接矩形を，Ptr(Pointer) はポインタを意味する．

### 3.5 索引木の探索アルゴリズム

文献 [6] では，索引木の探索アルゴリズムについて詳述していないが，概ね以下の通りである．

（1） 矩形範囲を一次元の範囲に変換する．

（2） 索引木を探索し，一次元の範囲内のキー値をもつ空間オブジェクトを求める．

（3） (2) の空間オブジェクトと矩形範囲で交差判定を行う．

## 4. 提案技術

本章では，まず，組込 DBMS 向けの空間索引技術の提案にあたり，基本方針を示す．次に，PMR 四分木 [6] を参考にし，索引木のデータ構造を提案する．そして，この索引木を用いた範囲検索のアルゴリズムについて述べる．

### 4.1 課　　題

組込 DBMS における空間索引の実現で，下記を課題とする．

（1） 索引木のデータ構造の柔軟性：開発者が，組込機器のリソース，地図の特性，検索機能を考慮し，索引木のデータ構造を設定可能にする．例えば，座標値のデータ量を数値のデータ型を変更して設定可能にすることが挙げられる．この設定は，アプリケーション毎に要求される座標値の数値の解像度や精度に合わせて利用される．

（2） 索引木のディスク容量の削減：車載情報端末のディスクは，開発費の削減のため，低価格で小容量のディスクとなる．開発者は，地図データの管理に組込 DBMS を採用する際，組込 DBMS の採用前に比べ，ディスク容量が急増しないことを要求する．そのため，組込 DBMS の採用でディスク容量の増加の原因とされる索引木のディスク容量を削減することが必要になる．

本報告では，研究上の課題として，課題 (2) に焦点を当てる．課題 (1) は実装上の課題となるため，本報告での論点としない．

### 4.2 ディスク容量の削減のための基本方針

本報告では，以下の基本方針に基づき，四分木の索引木を提案する．

（1） 索引木に最下位の分割矩形（分割矩形を含まない分割矩形）の情報だけを格納する．

PMR 四分木と同様の方針である．この方針により，ポインタベース四分木と比較し，索引木内の分割矩形の情報を削減できる．分割矩形をモートン符号値による整数値で表現し，その整数値をキー値とした一次元の索引木で管理する．本報告では，一次元の索引木に B 木を用いる．

（2） 索引木のリーフノードにキー値を重複せずに格納する．

PMR 四分木では，リーフノードにおいて，一つの空間オブジェクトに対し，一つのキー値が付与される．そのため，ある分割矩形に交差する空間オブジェクトが複数存在する場合，同じキー値が重複してリーフノードに格納され，冗長になる．そこで，提案索引木では，リーフノード内のキー値の重複を解消する．具体的には，同じキー値をもつ（同じ分割矩形に交差する）空間オブジェクトの情報をまとめて別のノード（オブジェクトノードと呼ぶ）で管理する．オブジェクトノードには，オブジェクトの外接矩形とレコードへのポインタが格納される．

（3） 索引木内の空間オブジェクトの外接矩形を分割矩形内の相対座標で表現する．

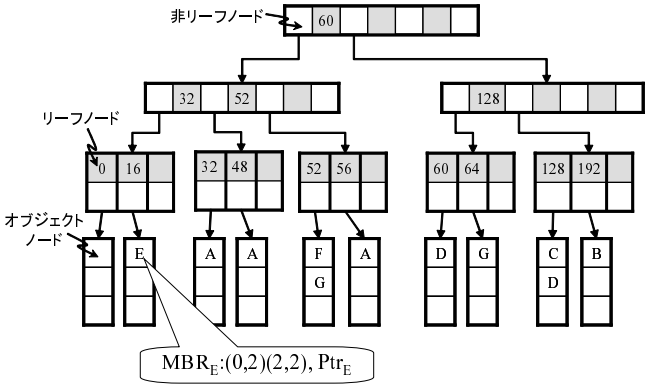


図 6　提案索引木

相対座標は，全空間の座標を表現する絶対座標に比べ，少ないビット数で表現できる．その結果，オブジェクトノードに格納可能な空間オブジェクトの件数が増えるため，索引木のオブジェクトノードの数が減り，索引木のディスク容量を削減できる．

### 4.3 索引木のデータ構造

提案索引木は，分割矩形を管理する非リーフノードとリーフノード，空間オブジェクトを管理するオブジェクトノードから構成される．非リーフノード，リーフノード，オブジェクトノードは，それぞれ 1 ページ固定長の単位でディスクに格納される．

非リーフノードには，i 個のキー値と i+1 個の下位ノードへのポインタが格納される．キー値は，分割矩形に対応するモートン符号値となる．下位ノードへのポインタは，下位ノードを含むページのディスク上の格納位置を示す．

リーフノードには，分割矩形に対応するキー値とオブジェクトノードへのポインタを要素とし，ページの容量の限り，複数の要素が格納される．オブジェクトノードへのポインタは，オブジェクトノードを格納しているページのディスク上の格納位置を示す．

オブジェクトノードには，空間オブジェクトの外接矩形，実データのディスク上の格納位置を示すポインタを要素として，ページの容量の限り，複数の要素が格納される．

図 6 は，図 1 の空間オブジェクトをアルファベット順に挿入した場合の提案索引木を示す．この図では，オブジェクトノードが最大 3 件の空間オブジェクトの情報を格納できるものと想定している．そのため，分割矩形は最大 3 件の空間オブジェクトに交差できる．この場合，空間分割は，図 4 と同様になる．オブジェクトノード内の空間オブジェクト E を指す吹き出しは，オブジェクトノードが空間オブジェクトの外接矩形とポインタの情報をもつこと示す．空間オブジェクト E の外接矩形は，分割矩形 16 の左下座標からの相対座標で表現され，左下座標 (0,2) と右上座標 (2,2) となる．

### 4.4 索引木の探索アルゴリズム

本節では，範囲検索の処理を実行したときの索引木の探索アルゴリズムについて説明する．本アルゴリズムの手順は以下の通りである．

（1） 矩形範囲を一次元の範囲に変換する．

（2） 索引木を探索し，一次元の範囲内のキー値をもつ空間

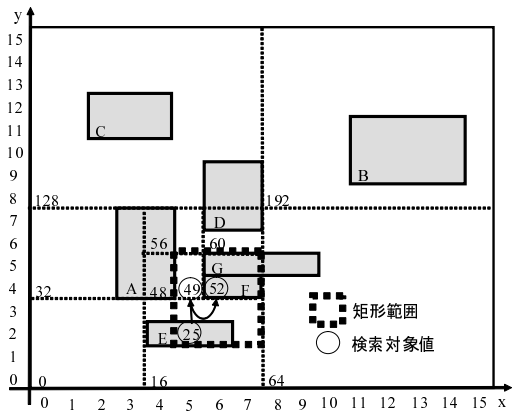


図 7　提案索引木による分割矩形の探索

オブジェクトを求める．その際，索引木のオブジェクトノードまで探索するたびに，矩形範囲内の座標で，かつまだ未探索の座標の中で，最小のモートン符号値を次の検索対象のキー値とする．

（3） 手順 (2) で求めた空間オブジェクトと矩形範囲で交差判定を行う．

図 7 は，範囲検索で提案索引木を用いる場合の分割矩形の探索を示す．太い点線で示す矩形範囲に対し，検索対象のキー値が 25，49，52 の順に更新され，矩形範囲に交差する空間オブジェクトを求める．

### 4.5　車載情報端末の地図描画のための拡張機能

提案索引木を用いた範囲検索は，主に車載情報端末の地図描画に利用される．車載情報端末の地図描画では，縮尺を指定した描画や，特定の種類の空間オブジェクトの描画が要求される．これらの機能では，矩形範囲に交差する空間オブジェクトの中で，指定する縮尺や種別などの属性情報に合致する空間オブジェクトを検索する必要がある．そこで，提案索引木のリーフノードで，空間オブジェクトに属性情報を付与することを拡張機能とする．この拡張機能により，提案索引木の探索で指定する属性情報に合致する空間オブジェクトを区別できる．

### 4.6　従来索引木との比較

表 2 に，提案索引木と従来索引木の比較を示す．提案索引木は，PMR 四分木と比較して，分割矩形の情報を重複して格納しない点，空間オブジェクトの外接矩形を相対座標で表現する点で，ディスク容量を削減できる．また，提案索引木は，ポインタベース四分木に比べ，空間オブジェクトを交差する全ての分割矩形と対応付ける点で，ディスク容量が増えるが，最下位の分割矩形のみ格納する点，分割矩形をモートン符号値で表現する点，空間オブジェクトを相対座標で表現する点で，全体ではディスク容量を削減できる．

## 5.　性 能 評 価

本章では，シミュレーション実験による提案索引木の性能評価の結果を示し，考察をまとめる．

### 5.1　目　　的

提案索引木が，索引木の格納に必要なディスク容量と，範囲検索に要するディスクアクセス回数の面で，従来索引木より優位であることを確認する．そのため，本研究成果の適用先となる車載情報端末の地図データに近いモデルでシミュレーション実験を行う．

### 5.2　実 験 環 境

本シミュレータの機能は，下記の通りである．

- 索引木の作成：空間オブジェクトを挿入すると，メモリ上に提案索引木，ポインタベース四分木，PMR 四分木の索引木のいずれかを作成する．
- 範囲検索の処理：矩形範囲を指定すると，索引木を用いて，矩形範囲に交差する空間オブジェクトを取得する．
- 評価項目の測定：索引木のディスク容量や範囲検索の処理に必要なディスクアクセス回数などを測定する．

評価項目は，下記の二つである．

- 索引木の格納に必要なディスク容量：索引木のノードをページ単位に格納する場合に必要なディスクの容量
- ディスクアクセス回数：範囲検索の処理で索引木のノードを格納しているページを，メモリからディスクに読み込む回数

シミュレータの評価対象となる索引木は，提案索引木，PMR 四分木の索引木，ポインタベース四分木の索引木である．提案索引木は，4.2 節の各基本方針の効果を示すため，二つに分類する．提案索引木 1 は基本方針 (1)(2) を適用した索引木，提案索引木 2 は基本方針 (1)(2)(3) を適用した索引木となる．基本方針 (1) のみを適用する索引木は，PMR 四分木に相当する．

本実験では，以下の空間オブジェクトの集合を用いた．

（1） 国土地理院の数値地図 2500（東京）の空間オブジェクトの集合

本地図データは，道路，街区，建物などの空間オブジェクトを含む．数値地図 2500 は車載情報端末の地図データではないが，実世界の地図データであるため，本実験に用いる地図データとして有用と判断した．なお，本実験では，日本の平面直角座標系で 9 系に属する東京の空間オブジェクトのみを利用する．東京の空間オブジェクトの大部分は 9 系に属する．

（2） ランダムな空間オブジェクトの集合

車載情報端末の地図データベースの規模を想定し，空間を日本の国土とし，その空間内に数千万件の空間データが存在する環境で実験を行った．空間オブジェクトは空間内に一様ランダムに配置される．空間オブジェクトの面積は，数値地図 2500 の東京の空間オブジェクトの平均面積に相当する正方形とした．

### 5.3　実 験 結 果

本節では，まず，実世界の地図データを用いた評価として，数値地図 2500 の東京の空間オブジェクを用いた実験の結果を示す．次に，車載情報端末の地図データベースの規模を想定した評価として，ランダムな空間オブジェクトの集合を用いた実験の結果を示す．

#### 5.3.1　実環境の空間オブジェクトを用いた実験

本実験では，表 3 に記載の数値地図 2500 の東京の空間オブジェクトの集合を用いた．表 3 は，日本の平面直角座標系で 9 系に含まれる東京の空間オブジェクトに関する．

表 6 に，本実験のパラメータの設定を示す．表内の最外郭矩形は，本実験で用いる空間オブジェクトの解像度に合わせて 1 メートルの単位とし，全ての空間オブジェクトを包含する矩形

表 2　索引木の比較

| | 提案索引木 | PMR 四分木 | ポインタベース四分木 |
|---|---|---|---|
| 空間オブジェクトを対応付ける分割矩形 | 交差する全分割矩形 | 交差する全分割矩形 | 包含する最小の分割矩形 |
| 格納する分割矩形 | 最下位の分割矩形のみ | 最下位の分割矩形のみ | 全ての分割矩形 |
| 分割矩形の表現 | モートン符号値 | モートン符号値 | 左下座標と右下座標 |
| 分割矩形の情報の重複 | なし | あり | なし |
| 空間オブジェクトの外接矩形の表現 | 相対座標 | 記載なし | 絶対座標 |

表 3　数値地図 2500 の東京の空間オブジェクトの構成

| 種別 | 件数 | 割合 [%] | 平均面積 [$m^2$] |
|---|---|---|---|
| 街区 | 203,612 | 22.7 | 7,698 |
| 内水面 | 3,155 | 0.3 | 192,733 |
| 道路 | 648,972 | 72.3 | 1,578 |
| 市町村界 | 4,597 | 0.5 | 518,118 |
| 建物 | 19,439 | 2.2 | 2,015 |
| 丁目 | 14,695 | 1.6 | 213,662 |
| 場地 | 3,660 | 0.4 | 80,456 |
| 合計 | 898,130 | 100 | |

表 4　実環境の空間オブジェクトを用いた実験のパラメータ設定

| パラメータ名 | 値 |
|---|---|
| 最外郭矩形 | 524,288[m] × 524,288[m] |
| 空間オブジェクトの総数 | 898,130 件 |
| ページのサイズ | 4,096[バイト] |
| ポインタのサイズ | 8[バイト] |
| キー値のサイズ | 38[ビット](=19 × 2) |
| 矩形範囲 | 300[m]，500[m]，1,000[m]，2,000[m]<br>5,000[m] 四方の正方形 |

表 5　索引木のディスク容量［バイト］

| | PMR | Proposed1 | Proposed2 | Pointer |
|---|---|---|---|---|
| 非リーフ | 4,096 | 4,096 | 4,096 | 245,760 |
| リーフ | 60,960,768 | 48,381,952 | 29,220,864 | 62,025,728 |
| 合計 | 60,964,864 | 48,386,048 | 29,224,960 | 62,271,488 |
| 合計の相対値 | 0.98 | 0.78 | 0.47 | 1 |

の範囲を 2 の累乗で設定した．これは，キー値のビット数を x 座標と y 座標を表現するのに最低限必要なビット数としたためである．524,288[m] は 219 となるため，キー値は 38 ビット（= 19 × 2）で表現できる．

表 5 は，数値地図 2500 の東京の空間オブジェクトを挿入した場合の索引木のディスク容量を示す．表内の “PMR” は PMR 四分木の索引木，“Proposed1(2)” は提案索引木 1(2)，“Pointer” ポインタベース四分木の結果を示す．以降，これらの表記を表やグラフ内の索引木の区別に用いる．表 5 の結果から，提案索引木 2 のディスク容量が最小になり，ポインタベース四分木の 47%に抑制できることが分かる．これは，4.2 節のディスク容量の削減のための基本方針の効果を示す．また，PMR 四分木，提案索引木 1，提案索引木 2 を比較すると，4.2 節の三つの基本方針の中で，空間オブジェクトの外接矩形を相対座標で表現する基本方針 (3) の効果が大きいことがわかる．

図 8 は，矩形範囲の大きさを変化させたときのディスクアクセス回数の結果を示す．図 8 の結果から，提案索引木 2 のディスクアクセス回数が最小であることがわかる．これは，提案索引木 2 のディスク容量が最小であるため，ディスクからメモリ

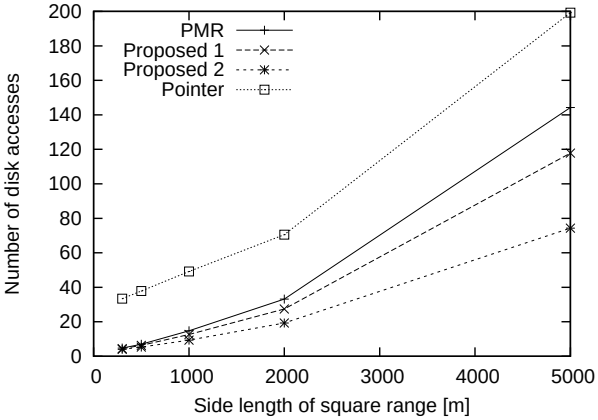


図 8　矩形範囲 1 辺の長さとディスクアクセス回数

表 6　実環境の空間オブジェクトを用いた実験のパラメータ設定

| パラメータ名 | 値 |
|---|---|
| 最外郭矩形 | 134,217,728 [ミリ秒] 四方の正方形 |
| 空間オブジェクトの総数 | 100 万件，500 万件，1,000 万件，<br>1,500 万件，2,000 万件 |
| ページのサイズ | 4,096[バイト] |
| ポインタのサイズ | 8[バイト] |
| キー値のサイズ | 54[ビット](=27 × 2) |
| 矩形範囲 | 300[m]，500[m]，1,000[m]，2,000[m]<br>5,000[m] 四方の正方形 |

に読み込むページの数が少なくなることを示す．また，各索引木ともに，矩形範囲が大きくなると，ディスクアクセス回数が増加することがわかる．これは，矩形範囲に交差する分割矩形の数が増え，分割矩形と空間オブジェクトを格納しているページの数が増えるためである．

**5.3.2**　一様ランダムな空間オブジェクトを用いた実験

本実験では，車載情報端末における地図データベースを想定し，日本の国土に相当する空間内に配置させた空間オブジェクトを用いる．空間オブジェクトは，数値地図 2500 の東京の空間オブジェクトの特性に従い，表 5 の割合で各空間オブジェクトを発生させ，表 5 の平均面積と同じ面積の正方形とし，空間内にランダムに配置させた．

表 6 に，本実験のパラメータの設定を示す．日本の国土は，東経 122 度から 154 度，北緯 20 度から 46 度に囲まれた領域に存在する．最外郭矩形は，その領域を包含し，1 辺が 2 の累乗で表現できる最小の矩形として，ミリ秒単位で設定した．キー値のビット数は，前節の実験と同様に設定し，134,217,728 は 2 の 27 乗であることから，54 ビット（= 27 × 2）とした．本実験で変化させるパラメータは，索引木の性能に影響を与える空間オブジェクトの総数と矩形範囲とした．

図 9 と図 10 に，空間オブジェクトの総数を変化させたとき

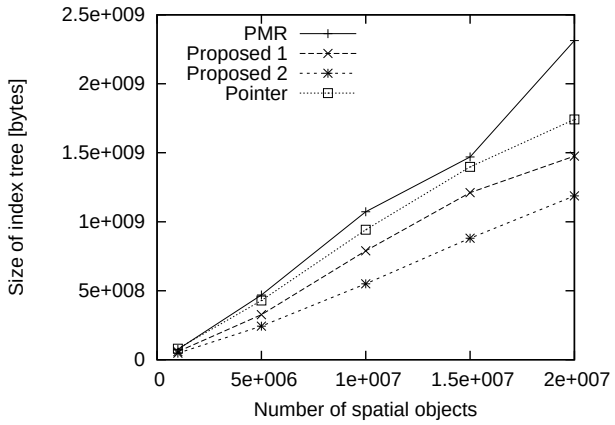


図 9　空間オブジェクトの件数と索引木のディスク容量

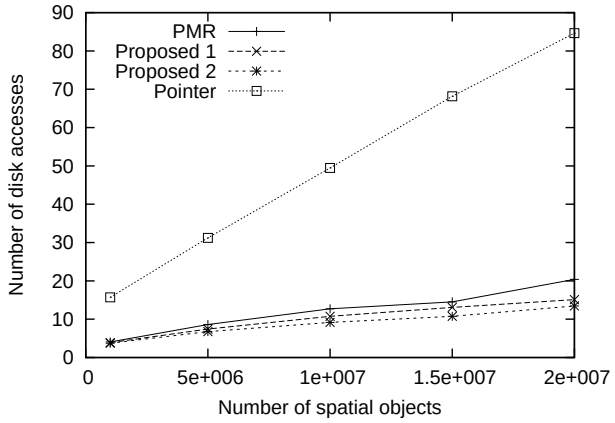


図 10　空間オブジェクトの件数とディスクアクセス回数

の索引木のサイズとディスクアクセス回数の性能を示す．本実験では，矩形範囲を 1 辺が 1,000[m] 四方の正方形とした．

図 9 の結果より，空間オブジェクトの総数が多い環境でも，提案索引木 2 の索引木のサイズが最小であることがわかる．これは，オブジェクトの外接矩形を相対座標で表現している効果が大きいことを意味する．また，PMR 四分木の索引木のディスク容量が，ポインタベース四分木より大きくなることがわかる．これは，分割矩形の符号を示すキー値のサイズが大きく（54 ビット），各オブジェクトに対し，キー値が付与されると，ディスク容量が大きくなるためである．

図 10 の結果より，提案索引木 2 のディスクアクセス回数が最小であることがわかる．これは，提案索引木 2 のディスク容量が最小であることに起因する．一方で，ポインタベース四分木は，空間オブジェクトの総数が増えるほど，ディスクアクセス回数が急増することがわかる．これは，分割矩形の境界線に重なる空間オブジェクトを上位の非リーフノードで管理するため，索引木の探索で上位から下位の非リーフノードで，リーフノードへのディスクアクセスが発生するためである．

図 11 に，矩形範囲の大きさを変化させたときのディスクアクセス回数の結果を示す．本実験では，空間オブジェクトの総数を 1,000 万件とした．図 11 の結果から，提案索引木 2 のディスクアクセス回数が最少であることがわかる．これは，提案索引木 2 では，各空間オブジェクトの外接矩形を相対座標で表現することで，索引木の格納に必要なディスク容量が最小になるためである．一方，ポインタベース四分木のディスクアクセス回数が最大であることがわかる．これは，矩形範囲に交差する上位から下位の分割矩形の非リーフノードに接続するリーフノードにアクセスするためである．

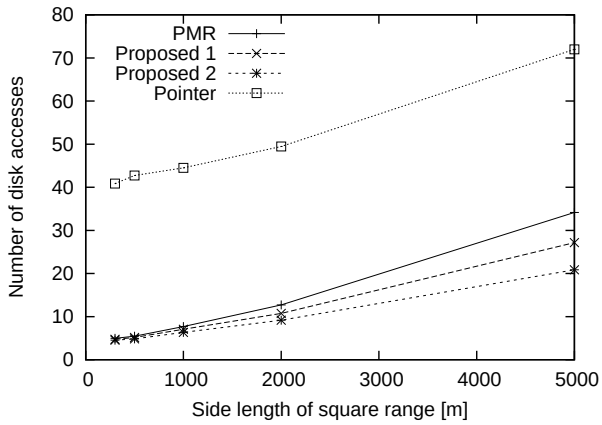


図 11　矩形範囲 1 辺の長さとディスクアクセス回数

## 6. ま と め

本報告では，車載情報端末を適用先とする組込 DBMS において，形状が線や面の空間オブジェクトを対象とした空間索引技術を提案した．提案技術は，空間オブジェクトの管理で優位とされる PMR 四分木に基づいた索引木である．シミュレーション実験により，提案索引木は従来索引木よりディスク容量とディスクアクセス回数の面で有効であることを確認した．実世界の地図データを用いた評価では，提案索引木は，文献 [5] の索引木の 47%にディスク容量を削減できることを確認した．

今後の課題は，空間オブジェクトの実データのフェッチ処理も含めた方式検討及び性能評価が挙げられる．地図描画では，索引木の探索後，詳細交差判定で空間オブジェクトの実データをディスクからメモリに読み込むフェッチ処理が入る．実データのフェッチ処理を高速化するため，空間的に近い空間オブジェクトをディスク上の近くに配置する工夫を行う必要もある．実データのフェッチ処理も含め，性能を評価する必要がある．

## 文　　献

[1] 経済産業省, “2008 年版組込みソフトウェア産業実態調査報告書 –プロジェクト責任者向け調査–” (2008 年 7 月).

[2] 日立 組み込みデータベース Entier, http://www.hitachi.co.jp/Prod/comp/soft1/ Entier/

[3] Oracle Database Lite, http://www.oracle.co.jp/database/Lite_Edition.html.

[4] IBM DB2 Everyplace, http://www-06.ibm.com/jp/software/data/db2/everyplace/.

[5] H. Hayashi, D. Ito, M. Tanizaki, K. Kimura, H. Kajiyama, “Dual-heap kNN: k-nearest neighbor search for spatial data retrieval in embedded DBMS,” Proc. of ACM GIS’08, pp. 357-366 (Nov. 2008).

[6] G.R. Hjaltason, H. Samet, “Speeding up construction of PMR quadtree-based spatial indexes,” VLDB J. vol. 11, no. 2, pp. 109-137 (2002).

[7] G.M. Morton, “A computer oriented geodetic data base and a new technique in file sequencing,” IBM Technical Report (1966).