データベース・システム 74－3
（１９８９．１１．１７）

地図データベース検索高速化のためのファイルインデックス方式

田中　章司郎　　池田　秀人

地理情報システム（GIS: Geographical Information System）においては、従来の文字数値型のデータを扱う一方で、対応する地図データ、例えば都道府県や島のような領域（面）、鉄道網、道路のような線、工場の位置のような点を同時に表す必要がある。

文字数値からのデータの検索は、従来のB-tree、ハッシングなどの方法によって、大量のデータを高速に検索することが可能である。GISでは、これらの機能に加えて、地図をはじめとする図形（点、線、面）をキーとしたデータの検索に対しても、効率よく応答できることが必要となる。

本報告では、地理情報システムの要求分析、定義・格納・検索方法等を検討し、新たにファイルインデックス方式を提案する。

FILE INDEX METHOD FOR MAP DATABASES

Shojiro TANAKA　　Hideto IKEDA

Hiroshima University

1-1-89 Higashidenda-machi, Nakaku, Hiroshima 730　Japan

Geographical Information Systems(GISs) require to represent map data like municipal jurisdiction boundary and islands as area, railroad webs and highways as line, and factory locations as point, which correspond to conventional character/numeric data.

B-tree or hashing method in practice make it possible to retrieve character/numeric data very fast already. Yet geometric keys, of point,line, and area in map data query should provide speedy response in GISs as well as the above conventional facility of one-dimensional domain.

In this research paper, we consider a new file index method for geometric data through examination of GIS requirements, its data definition,storage, and the query features.

１．はじめに

地図及びそれに付随する数値・文字データをデータベースとして格納し、データの検索・表示・更新及び解析等を行うシステムを地理情報システムと名付けると、それは国土数値情報、大気汚染・気象のモニタデータ、人工衛星データ、国勢調査・農林業センサス等の地図と関連して取られている統計データ、などの広範囲の情報の管理や利用に供することのできる汎用的なシステムと位置付けることができる。

都市化による人口動態の急激な変化、砂漠化・熱帯雨林の消失、酸性雨、食糧生産量の偏在など、単なる数値の羅列では容易に把握すらできない問題が急速に顕在化する一方、上記のような大量のデータが急ピッチで集められている。

さまざまな行政的計画策定において、大量で多様なデータを、判り易く具体的に表示する必要性は、今後ますます増大の一途を辿ることが予想され［5］、学術的な諸分野においても、集積されたそれぞれの知見を地図空間上に重ね合せて結集し、発見的・学際的な解析を押し進めることを可能にすることが重要である。また企業においては、地域ごとのきめこまかなマーケティングを展開し、より緻密な意志決定を支援する。

高速道路沿い左右１ｋｍのガソリンスタンド、行政域中の対応するメッシュ、汚染海域をカバーする管理主体（自治体）等の地図座標をキーとする統計数値データの高速検索は、現在の汎用データベースシステムでは困難である。GISは、点、線、面の地図データと文字数値データが全く別々のDBMSで管理され、アプリケーションの機能の割には道具が大きかったり、データがアプリケーションに従属していて、データそのものの汎用的なデータ独立性が確保されていないなどの問題を残しているのが現状である。また、ある点からの道程と距離を予め計算してデータを作成する方法などがよく用いられるが、この場合、他の交通経路が新設された時には、すべて見直して再作成しなくてはならなくなり、更新に負荷がかかるなどの問題もある。地図データとして多目的に集中して管理するには不適当な方法である。

一方、そのような点を改善した、地図を中心とする図形データの構成法として、Quad tree または Linear Quadtreeが現在注目されている［3,1,6］。しかし、(a)斜線に対して階段状の近似であり、精度が高くなると矩形の数が増え、島嶼部などの自然に発生する複雑な図形に対する記述は、簡潔であるとはいえない、(b)ファイル作成時には比較的大掛りな前処理を必要として、効率が良くない、(c)高精度の複雑な図形の検索は全マッチングに近くなるので、大量データに対する高速検索が困難である、などの短所がある。

我々は、このような諸点を踏まえて、図形をキーとした図形データ検索を高速にするファイルインデックス方式を提案し、図形の定義、格納、検索の諸点に対して検討する。

２．ファイル構成法

地理情報システムの図形データの検索には、(a)文字・数値から図形を検索する(b)図形から文字・数値を検索する(c)図形から図形を検索する、のそれぞれがあり、(c)にはさらに包含、重なりの場合がある。また、数値の平均等と同様に、図形の距離・面積の計算も図形の扱いに必要である。これらの要求事項を分析することは、良いファイル構成の必要条件を提示する。

２．１　地理情報システムの要求事項

地理情報システムとして、次のような要求を満足することが必要である。

(1)データ要素の定義：文字、数値はもとより、点、線、面が客観的に、かつ曖昧性なく定義できなくてはならない。とりわけ図形自体を属性として扱い、図形データが独立している定義が望まれる。

(2)データの格納：必要なファイル容量が少なく、前処理時間も短かいものが望ましい。図形を格納する方法（ベクトル法、ラスタ法）は、対象とする図形に応じた正確な近似である必要がある。

(3)データの更新：文字、数値データ型は一般のDBSと同様であるが、地図を中心とする図形データの場合、更新の機会は、道路が新設されたり、町村が市に合併された時などで、さほど多くはない。しかし、共有DBでは、いち早く少ない労力で更新ができる必要がある。

(4)データ検索：文字・数値 -> 図形、図形 -> 文字・数値、図形 -> 図形の検索が容易にかつ高速に出来なくてはならない。複雑な図形同士の包含関係、重なり関係等の判定を伴う検索は、対象となるデータが通常大量であるため、特に高速で処理されることが必要とされ、またその際の主記憶容量等の資源を大量に消費するものであってはならない。

(5)データ操作：数値データの平均値のように、図形データに対しても、その長さ、面積、近隣図形との距離などが計算できなくてはならない。

(6)データ表示：検索の結果のビューなどは文字・数値、ないしグラフィック装置等で表示できるものでなければならない。

２．２　図形の定義

既に我々は、SQLの拡張(XSQL:EXtended SQL)を提案し［8］、言語の拡張を行った。前節の(1)、とりわけ図形をひとつのデータ要素として定義し、ひとまとまりの定義域（ここでは２次元に限定する）が存在していれば、図形表現が理解し易く簡潔になり、演算子集合にも対応する一様性を導くことになる。XSQLのDBスキーマ定義は、図１の例のように点、線、面が図形属性のデータ型として各々定義されている（下線部）。POINT, LINE, AREAの精度、座標系等は、標準値として予め導入者が決定しているものとする。

```
CREATE TABLE  HOUSE      ( HOUSE#         INT(5)    NOT NULL,
                           LOCATION       POINT,
                           NUM_PERSON     INT(2),
                           NM_HOUSHOLDR   CHAR(20),
                           ENGAGEMENT     CHAR(10),
                           SPOUSE         CHAR(1),
                           NUM_CHILDREN   INT(2),
                           CITY#          INT(3)    NOT NULL,
                           UNIQUE         (CITY#,HOUSE#) )
CREATE TABLE  RAILROAD   ( RAILROAD#       INT(3)    NOT NULL,
```

```
                        NAME_RAIL       CHAR(30),
                        TYPE_ORBIT      CHAR(7),
                        NUM_STATION     INT(3),
                        BEGIN_STATION   CHAR(20),
                        END_STATION     CHAR(20),
                        RUN_ORBIT       LINE,
                        PLACE_STATN     POINT,
                        UNIQUE          (RAILROAD#) )
CREATE TABLE  CITY      ( CITY#         INT(3)   NOT NULL,
                        NAME_CITY       CHAR(30) NOT NULL,
                        FRINGE_CITY     AREA,
                        SEGMENT         CHAR(3),
                        POPULATION      DEC(9),
                        PLACE_CITYHALL  POINT,
                        UNIQUE          (CITY#) )
CREATE TABLE  IMAGE_XS  ( XS            IMAGE(3000,3000,256,3),
                        COVER           AREA(NEAREST_NEIGHBOR),
                        DATE            CHAR(8),
                        K_J             CHAR(7),
                        VIEW_ANGL       CHAR(5),
                        CLD_COV         DEC(4) )
```

図１　XSQLによるDBスキーマ

XSQLでは、画像データ（主に衛星画像）と面データを別々に扱った。しかし地図や地形データに対してはよいものの、光沢を持った物体等(Geometrical Data)に対しては有効ではなく、画像データ型は面データ型のオプションとして扱った方がより広い意味で図形データの整合性をもたせることができる。AREA属性は故に、面領域内にfillerがある場合（主にRGB各256階調画素）と、fillerが無い場合がある。従って、図１のIMAGE_XSのDBスキーマ［7］は、図１－１のように書き直した方が良い。境界は標準値として長方形を前提としているものとする。

```
CREATE TABLE  IMAGE_XS  ( XS        AREA( IMAGE(3000,3000,256,3),
                                          NEAREST_NEIGHBOR ) )
```

図１－１　修正XSQLによる衛星画像のDBスキーマ

図１のIMAGE_XSの撮影日付(DATE)、軌道ID(K_J)、俯角(VIEW_ANGL)、雲量(CLD_COV)等はDBスキーマの内容というよりも、DD/Dの内容であるので、図１では省略した。

２．２．１　検索の構文

前節の基底表を用いて、例えば「人口が２万人以上の市の領域に該当するスポット衛

星画像を求めよ」という検索をした場合、XSQLでは、図２のビューのような短かく簡潔な表現で目的が達成できる。標準SQLのDMLでは困難であるので、親言語PL/Iと埋込SQLを用いると、９５ステップを超える手間を要した［8］。

```
CREATE VIEW    DENSE ( CITY# , XS )
    AS SELECT  XS
       FROM    CITY , IMAGE_XS
       WHERE   INTERSECTION(FRINGE_CITY,XS) NOT = φ
       AND     POPULATION >= 20000
```

図２　XSQL-DMLの例

２．３　格納法

次にXSQLの外部仕様に従って、次にその内部仕様を検討する。図１に対して、具体的に図３のような例を設定する。簡単のため、全てのX,Y 座標値は１から１０までの整数にしてある。図中の逆三角形は世帯、国鉄の線は鉄道、その線上の四角形は駅、黒丸は行政領域境界、黒い四角形はそのうちのデータの始点、そして楕円は市庁舎をそれぞれ表している。尚、filler付き面データは、本報告では割愛して別の機会に譲る。

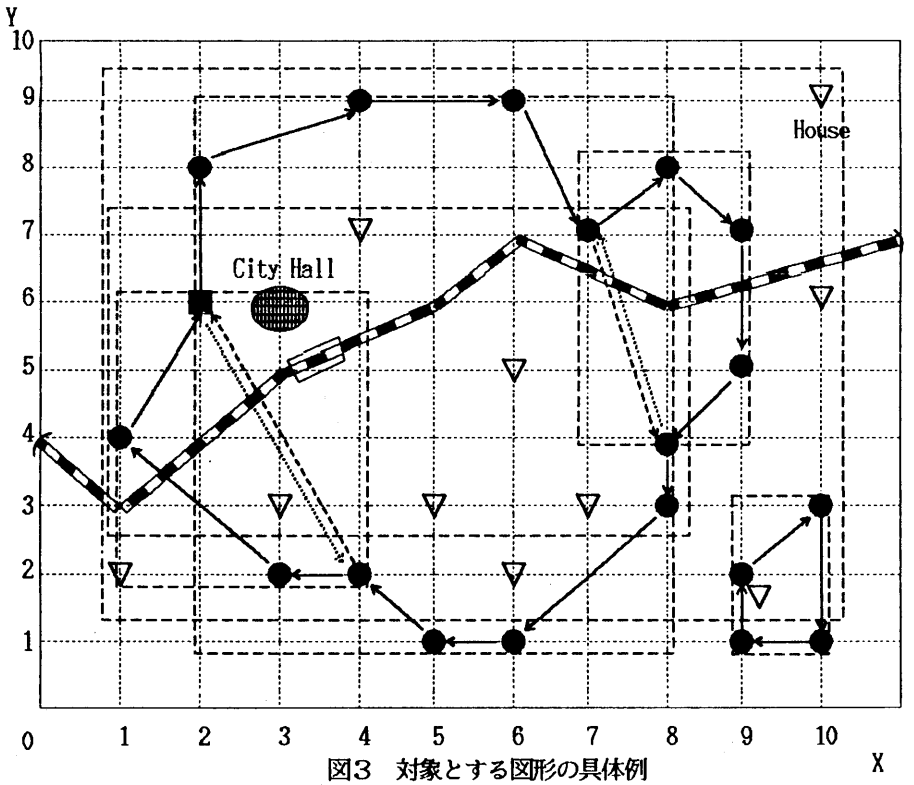

図３　対象とする図形の具体例

XSQLにおける点、線、面の各値は、RDB表中で可変長の不可分値(atomic value)として取り扱われる。それは格納しようとする図形データが、面の画像オプションを除くとディジタイザ等で取得された、レコード長の一定でないファイルであり、かつ多くの場合、データ容量が膨大であって少しでも冗長さを取り除く必要があるからである。

各図形データ型別座標情報採取プロセスを以下に論ずる。それらの中間的データは、それぞれ図４に示される。座標・セグメント以外の、図形ID等の付随した情報は適切に採取されるものとする。

２．３．１　点情報の採取

図１の世帯(HOUSE)DBスキーマの中で点として定義されている(LOCATION)フィールド値が図３の各逆三角形である。点のX,Y 座標を採取する時はその順序は特に必要とせずに、項目毎漏れの無い様にすればよい。

２．３．２　線情報の採取

線の座標採取には順序が必要である。最も X座標の小さな点、つまり図形の最も西に近い端点から順に線に沿って道なりに採取する。たまたま両端点が一致した場合にはY座標の大きな点の座標から始める。閉ループ状の線は最も X座標の小さなもの、つまり図形の最も西端から時計回りに座標を採取する。その最も小さなX座標に対応するY座標が複数ある場合は、そのうちYの値が一番大きいものを選んで開始する。

２．３．３　面情報の採取

図３のCITYのDBスキーマに相当する領域は、島のような飛び地をもつから、先にどちらかから始めて、

(i)最も X座標の小さなもの、つまり図形の最も西端から時計回りに境界座標を採取する（SEGMENT変数を１にセットする：親領域）。最も小さなX座標に対応するY座標が複数ある場合は、そのうちYの値が一番大きいものを選ぶ。飛び地のない時のSEGMENTフィールド値は０（ゼロ）である。

(ii)次に残りの飛び地（子領域）の閉ループ座標を同様に採取しながら、対象の子領域がひとつ増える毎にレコードを次に移しSEGMENT変数に１をたして、子領域がなくなるまで繰返す。

ここまでの手順で採取されたデータは図４となる。

(iii)次に、凸包でないその親・子領域については、それぞれ隣接する凸包に分割する。これは、凸包内の任意の２点を結ぶ直線はその境界とは接しないため、境界座標の内側を埋める画素等の配置が容易となり、またこのような凸包の性質が良く研究されていて［4,9］、第２．５節の包含関係、重なり関係等の領域探索問題のアルゴリズムが既にほぼ確立されているためである。

(iv)最も X座標の小さなもの、つまり図形の最も西端から時計方向に座標点を順次ひとつずつ進んで境界座標を巡り、最初にその座標点を中心とする内角が180度を越える点を始点とする。最も小さなX座標に対応するY座標が複数ある場合は、そのYの値が一番大きいものを選ぶ。

(v)同様の手順により次の凸でない部分は少なくとも一つ存在するから、その場合、その点に印をつけ、その点を中心として、ひとつずつ次の点となす角度を比較し、最初にその内角が180度以下となる点を選んでそこからまたひとつずつ進んで内角を比較し、再び(iv)の始点に戻って他の凸でない部分が無くなるまで繰返す（親凸包）。

(vi)上記(v)で印を付けた点から、分岐した残りの座標に(v)の手続きを繰返して、元々のふたつの凸でない点の座標を取り込んで、子凸包を作成する。これを子凸包がなくなるまで繰返す。

LOCATION	(1,2),(3,3),(5,3),(6,2),(7,3),(10,6), (10,9),(4,7),(6,5),(9,2) |	<HOUSE DBスキーマ>
RUN_ORBIT	(1,3),(3,5),(5,6),(6,7),(8,6) |	<RAILROAD DBスキーマ>
PLACE_STATN	(3,5) |	
FRINGE_CITY	(2,6),(2,8),(4,9),(6,9),(7,7),(8,8), (9,7),(9,5),(8,4),(8,3),(6,1),(5,1), (4,2),(3,2),(1,4) | (9,2),(10,3),(10,1),(9,1) |	<CITY DBスキーマ>
PLACE_CITYHALL	(3,6) |	

図4　格納の中間ファイル

2．3．4　超長方形(Super Rectangular)の生成とレコードヘッダ作成

それぞれの点、線、凸包群を、

(i)それらに内接する座標格子に平行な超長方形(SR: Super Rectangular)で囲み、

(ii)その左上（北西）と右下（南東）の座標点を各単位図形のヘッダとして、図形座標レコードの頭に書き込む。

この手順に従って、図4を具体的に格納した形態が図5である。

LOCATION	(1,9),(10,2)	(1,2),(3,3),(5,3),(6,2),(7,3),(10,6), (10,9),(4,7),(6,5),(9,2) |
RUN_ORBIT	(0,7),(8,3)	(1,3),(3,5),(5,6),(6,7),(8,6) |
PLACE_STATN	(3,5),(3,5)	(3,5) |
FRINGE_CITY	(2,9),(8,1)	(2,6),(2,8),(4,9),(6,9),(7,7),(8,4) (8,3),(6,1),(5,1),(4,2) |

	(7,8),(9,4)	(7,7),(8,8),(9,7),(9,5),(8,4) |
	(1,6),(4,2)	(4,2),(3,2),(1,4),(2,6) |
	(9,3),(10,1)	(9,2),(10,3),(10,1),(9,1) |
PLACE_CITYHALL	(3,6),(3,6)	(3,6) |

図5　点、線、面の内部スキーマ［7］部分と Super Rectangular

２．４　索引付け

格納された図形ファイルの見出し部分(SR header)を基に、索引集合を作成する。B-tree［2,6］を使用する理由は、超長方形の対象領域の比較が中心であり、高速判定の目的に適しているためである。

図５の中間のカラムのSR部分を、(X,Y),(X',Y')と表すと、先に Xの範囲から絞りこむ意味で、X -> X' -> Y -> Y'の順に木を作成する。

この２段階の図形検索構造によって、まずワールド座標上の範囲外の図形を排除し、候補を絞りこむことで、SRが全く無い場合等と比べて、大幅に検索速度が向上することが予想される。

２．５　図形間の関係の判定

図形間の関係（包含、重なり）を整理すると、図６のように表すことができる。どの場合も最初にSuper rectangularによって、対象範囲内外の判定がなされ、範囲外である場合、直ちに包含、重なり関係は排除される。SR範囲内である場合には、以下の基本的な判定基準を用いることができる。

	POINT	LINE	AREA
POINT	f_1 =	-	-
LINE	f_2 $\ni$	f_4　g_1 $\supseteq$, $\cap$	-
AREA	f_3 $\ni$	f_5　g_2 $\supset$, $\cap$	f_6　g_3 $\supseteq$, $\cap$

図６　点、線、面相互の参照可能な関係

(f_1-f_6は包含関係、g_1-g_3は重なり関係を表す。)

２．５．１　包含関係

$$f_2(p,L) = \begin{cases} 1 & p \in L \\ 0 & p \in L, \quad \text{where } L \text{ is a direct line.} \end{cases}$$

$$f'_2(p,L) = \begin{cases} 1 & p \in L \Leftrightarrow f_2(p,L_i)=1 \text{ for some } L_i, \\ 0 & \text{otherwise} \qquad \text{where } L_i=L_1+L_2+ \ldots +L_n \end{cases}$$

$$f_3(p,A) = \begin{cases} 1 & p \in A \\ 0 & p \in A, \quad \text{where } A \text{ is a convex set.} \end{cases}$$

$$f_4(L_1,L_2) = \begin{cases} 1 & L_1 \subseteq L_2 \Leftrightarrow f'_2(p,L_2)=1 \text{ for all } p \in p(L_1), \\ 0 & \text{otherwise} \qquad \text{where } p(L_1)=\{p_1,p_2, \ldots ,p_n\} \end{cases}$$

$$f_5(L,A) = \begin{cases} 1 & L \subset A \Leftrightarrow f_3(p,A)=1 \text{ for all } p \in p(L), \\ 0 & \text{otherwise} \qquad \text{where } p(L)=\{p_1,p_2, \ldots ,p_n\} \end{cases}$$

$$f_6(A_1,A_2) = \begin{cases} 1 & A_1 \subset A_2 \Leftrightarrow f_3(p,A_2)=1 \text{ for all } p \in A_1 \\ 0 & \text{otherwise} \end{cases}$$

２．５．２　重なり関係

$$g_1(L_1,L_2) = \begin{cases} 1 & \exists p(p \in L_1 \text{ and } p \in L_2) \\ 0 & \text{otherwise} \end{cases}$$

$$g_2(L,A) = \begin{cases} 0 & g_1(L_1,L_2)=0 \text{ for all } L_2 \in L(A), \\ 1 & \text{otherwise} \qquad \text{where } n(L)=n\{p(L)\} \end{cases}$$

$$g_3(A_1,A_2) = \begin{cases} 1 & A_1 \cap A_2 \neq \phi \\ 0 & A_1 \cap A_2 = \phi \end{cases}$$

引用文献

[1] Burrough,P.A., Principles of Geographical Information Systems for Land Resources Assessment, Oxford University Press (1989)

[2] Date,C.J.; 藤原譲訳, データベース・システム概論 (第3版), 丸善 (1984)

[3] Gargantini,I., An Effective Way to Represent Quadtrees, Communications of ACM, 25(12) (1982) 905-910

[4] 伊理正夫・腰塚武志他, 計算幾何学と地理情報処理 (bit別冊), 共立出版 (1986).

[5] 国土庁計画・調整局, 地理情報システム, 大蔵省印刷局 (1986).

[6] 坂内正夫・大沢裕, 画像データベース, 昭晃堂 (1987)

[7] 佐藤英人, 統計データベースの設計と開発, オーム社 (1988)

[8] Tanaka,S. and Ikeda,H., An Integrated Data Model for Statistical and Geographical Data Management, Proceedings of World Conference on Information Processing and Communication (1989) 444-451.

[9] 鳥脇純一郎, 画像理解のためのディジタル画像処理 (II), 昭晃堂 (1988)