対話型意思決定支援システムの試作
ー国道と人口メッシュを例としてー

s98406-C
荒木　俊太郎
計算機科学講座
田中研究室

平成 14 年 3 月 1 日

目 次

第1章　序論

従来のＤＢ技術-ＲＤＢは文字、数値データしか扱えず、写真，地図などの静止画や動画，音声などといったマルチメディア情報を扱おうとすると画像内容を表すキーワードを付ける必要があるなどの制約や検索速度に問題があった。例えば「こんな感じの画像」といった画像そのものを条件にした検索はできない。人間の普段取得している情報の８割は文書以外の情報であり、このマルチメディア情報の取り扱いに対する需要は高まる一方である。この問題を解決するためにオブジェクト指向のＲＤＢ、「ＯＲＤＢ」が開発され、本研究ではこの「ＯＲＤＢ」を用いてどんな用途に利用できるかをシステムの試作を構築するという形で研究を行なった。

1.1　ORDBについて

オブジェクトリレーショナルＤＢＭＳは、完全にこれまでのリレーショナルＤＢＭＳの延長線上に位置しており、リレーショナルＤＢＭＳのユーザに完全なオブジェクト指向の環境を提供するものである。例として、オブジェクト指向環境の「クラス」を ORDBMS のユーザ定義型と読み換えるのみで、あとはそのまま「メソッド」、アクセス制御の「 private or public 」、型継承関係などが、現在開発中の SQL-99 で提供されている（下記参照)。このため、ＲＤＢＭＳにおいて別々のＤＢに格納していた汎用データと空間データをＯＲＤＢＭＳでは同一のＤＢに格納することが可能となる。

1.1.1　SQL-99によるユーザデータ定義型の例

(x,y) 座標から構成される「点データ型」の場合（ユーザ定義型とはいっても，想定しているのはエンドユーザではなく，アプリケーション開発者である）

```
1) CREATE TYPE ST_Point
2)    UNDER ST_Geometry
      AS (
3)       ST_PrivateX DOUBLE PRECISION DEFAULT NULL,
         ST_PrivateY DOUBLE PRECISION DEFAULT NULL
   )
4) INSTANTIABLE
5)  NOT FINAL

6)  METHOD ST_X()
      RETURNS DOUBLE PRECISION
```

```
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    RETURNS NULL ON NULL INPUT,

  ...
```

１） ST_Point というデータ型を定義する
２） ST_Point は ST_Geometry 型（論理的な整合性を取るための実体の無い型）
　　を継承する
３） ST_Point を倍精度の x,y 座標値から構成する
４） インスタンスを持つ．つまり実体がある.（ST_Geometry は NOT INSTANTIABLE）
５） さらに継承可能

６） ST_Point 内の X 座標を返すメソッドを，ユーザ定義型内部で定義する

1.1.2 ＯＲＤＢＭＳの必要性

従来のＲＤＢＭＳでは、2.2.1 で述べたように属性情報と空間データは別々のＤＢに格納していた。しかし、ＲＤＢＭＳでも工夫すればＯＲＤＢＭＳと同様にこれらのデータを同一ＤＢに格納することができるのではないだろうか？

以下に、ある区画内においての人口総数をＲＤＢＭＳに格納した場合を例として示す。ＤＢに格納する要素は次のものとし、空間情報については座標点Ｘ，Ｙを４点用いて作られた四角形を１区画として扱うものとする。

・識別ＩＤ
・人口総数
・空間情報（ＸＹ座標 × ４点）

まず次のようなものが挙げられる。

表 1.1: 横長にした場合

識別ＩＤ	人口総数	Ｘ１	Ｙ１	Ｘ２	Ｙ２	Ｘ３	Ｙ３	Ｘ４	Ｙ４
１	１００	１０	１０	１０	２０	２０	２０	２０	１０
２	２００	２０	２０	２０	３０	３０	３０	３０	２０
３	３００	３０	３０	３０	４０	４０	４０	４０	３０
４	４００	４０	４０	４０	５０	５０	５０	５０	４０
５	５００	５０	５０	５０	６０	６０	６０	６０	５０
・	・	・	・	・	・	・	・	・	・
・	・	・	・	・	・	・	・	・	・
・	・	・	・	・	・	・	・	・	・

表 2.2 のように横長にした場合、例のような４点で構成される単純な図形ならまだしも、複雑な図形になってくると、ＸＹ座標の数が多くなるにつれ表のカラム数も多くなるという欠点がある。

表 1.2: 縦長にした場合

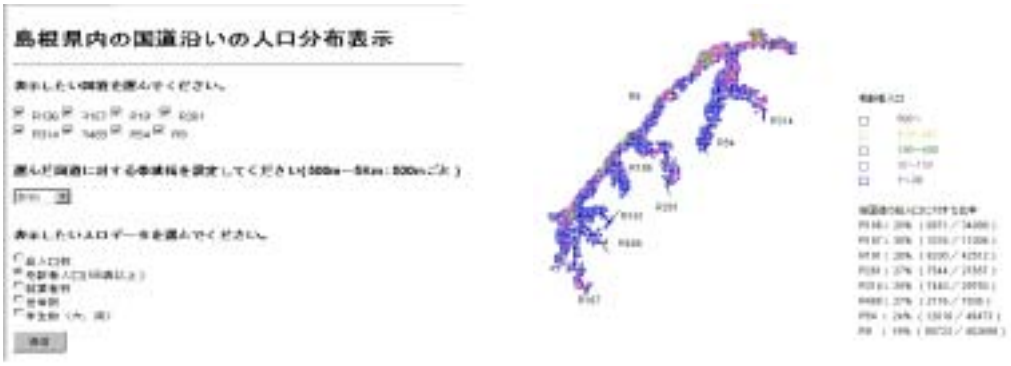
識別ＩＤ	人口総数	Ｘ	Ｙ
１	１００	１０	１０
１	１００	１０	２０
１	１００	２０	２０
１	１００	２０	１０
２	２００	２０	２０
２	２００	２０	３０
・	・	・	・
・	・	・	・
・	・	・	・

表 2.3 のように縦長にした場合も横長のものと同様に、ＸＹ座標の数が多くなるにつれ行数も増えてしまうだけでなく冗長性が生じてしまうという欠点がある。

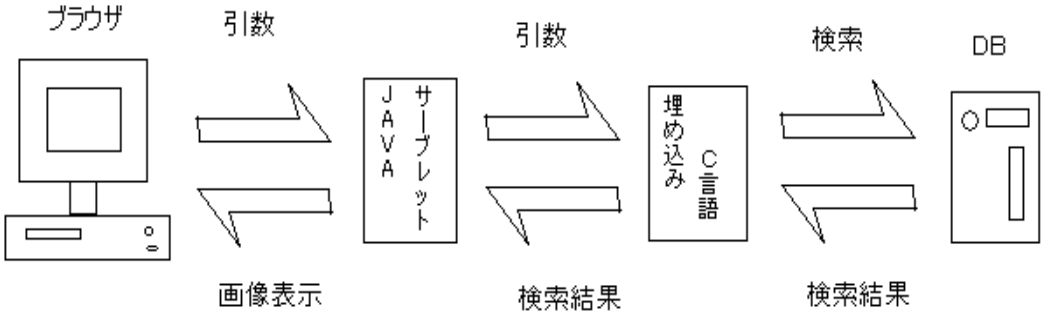
これらのことからＲＤＢＭＳは文字、数値以外のデータを扱うには不向きであることがわかる。よって空間情報を扱うにはＯＲＤＢＭＳが適しており、ＯＲＤＢＭＳはこれからのマルチメディア時代において必要不可欠なものになると思われる。

1.2　本研究のシステムの概要

本研究では「島根県内の国道沿いの左右帯域幅 $x$km($0.5 \leq x \leq 5$: 500m 毎) 内に含まれる選択された人口データに対する 1km × 1km の基準地域メッシュの表示」という検索を Web 上で利用するという仕様でシステムの試作を構築した。以下は実行画面である。



ユーザーはブラウザ上で表示したい国道をチェックボックスで選択し、(複数可) その国道に対する左右帯域幅を指定する。この左右帯域幅の範囲は 500m から 5km まで 500m 毎の値を選択ボックスから１つ選んでもらう。そして知りたい人口データの種類をラジオボタンで 1 種類選んでもらい、送信ボタンを押すとそれらの引数は Web の窓口である JAVA サーブレット、DB の操作に必要な SQL 言語が埋め込んであるＣ言語 (UAP と呼ぶ) と渡り、この引数をもとにしてＳＱＬ言語でＤＢに接続、検索を行なう。ＤＢからの検索結果は再びＵＡＰに引き渡され、その後 JAVA サーブレットで値を処理し、ブラウザに画像を表示する。



次の章から上図の各プロセスについて具体的な説明を展開していく。

第2章　使用したデータの概要

まず元としたデータの概要について述べる。作成したデータは国道座標値データ、人口データで取得元のデータはそれぞれ (財) 日本地図センター刊行のデジタルマップ「JMC マップ」、(財) 統計情報研究開発センター刊行の「平成７年度国勢調査に関する統計」である。なお全てのデータの生成はＣ言語プログラムによって行なった。

2.1　地域メッシュコード について

まず「JMC マップ」、「平成７年度国勢調査に関する地域メッシュ統計」に共通する規定、地域メッシュコードについて説明する。地域メッシュコードは、「標準地域メッシュ・システム」（昭和４８年行政管理庁告示第１４３号「統計に用いる標準地域メッシュ及び標準地域メッシュコード」）に基づくもので、一定の経度、緯度によって地域を網の目状に区画したときの位置を示すコードである。

地域別に情報を表示する方法（メッシュ法）は、統計データの表示をはじめとして、地形、自然環境、行政地域、道路・鉄道、公共施設、文化財などの位置・範囲等を数値化して表示するなど、多方面で利用されている。

2.1.1　地域区画区分

地域メッシュには、主に第１次地域区画、第２次地域区画、第３次地域区画（基準地域メッシュ）の３つの区画がある。（第４次地域区画は使用しないため説明を省略する）

第１次地域区画（１次メッシュ）

メッシュコードは４桁で表す。
約８０×８０ｋｍ

●上２桁：南端緯度　×　１．５（ただし、分の単位も含む）
●下２桁：西端経度の下２桁
【例】　南端緯度　３６度　００分　　西端経度　１３８度　　の場合
　　上２桁：３６×１．５＝５４　　下２桁：３８　→　５４３８

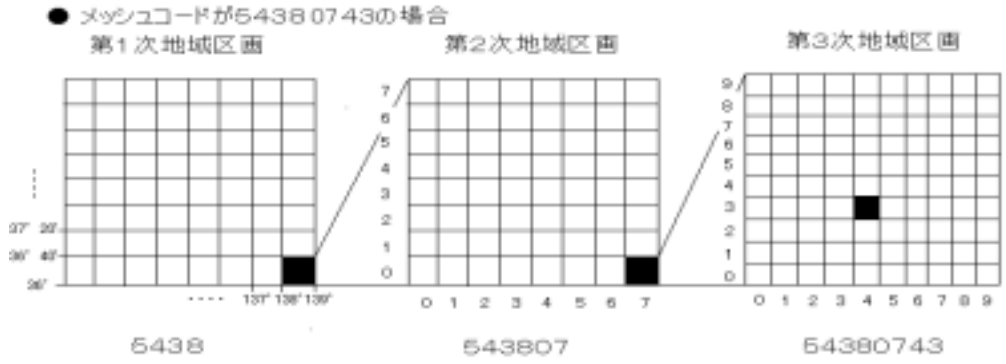
第２次地域区画（２次メッシュ）

第１次地域区画を　８×８等分　したもので、メッシュコードは６桁で表す。
約１０×１０ｋｍ
●上４桁：第１次地域区画のメッシュコード
●５桁目：第１次地域区画を縦に８等分し、南から０～７の番号を付け、これをそれぞれの区画を示す数字とする。
●６桁目：第１次地域区画を横に８等分し、西から０～７の番号を付け、これをそれぞれの区画を示す数字とする。
【例】　５４３８０７

第３次地域区画（３次メッシュ）

基準地域メッシュともいい、第２次地域区画を　１０×１０等分　したもので、メッシュコードは８桁で表す。
約１×１ｋｍ

●上６桁：第２次地域区画のメッシュコード
●７桁目：第２次地域区画を縦に１０等分し、南から０～９の番号を付け、これをそれぞれの区画を示す数字とする。
●８桁目：第２次地域区画を横に１０等分し、西から０～９の番号を付け、これをそれぞれの区画を示す数字とする。
【例】　５４３８０７３４



2.2　国道データ

ＪＭＣマップは日本地図センター刊行のデジタルマップで、日本全国を２０万～５０万分の１程度の縮尺に対応する数値地図情報である。行政界・道路・鉄道・河川・海岸線が数値化された線データと、行政名・自然地名等の点データから構成されており、広い範囲を概観するのに適した地図の骨格をなす情報を収めたデータである。なお、データの座標値は第２次地域メッシュ区画単位で記録されており、各座標値は左下を（0,0）、右上を（10000,10000）とする正規化座標となっている。

2.2.1 データの記録方式

1. データの概要

データファイルは１次メッシュ単位になっており、以下の図のような構造になっている。

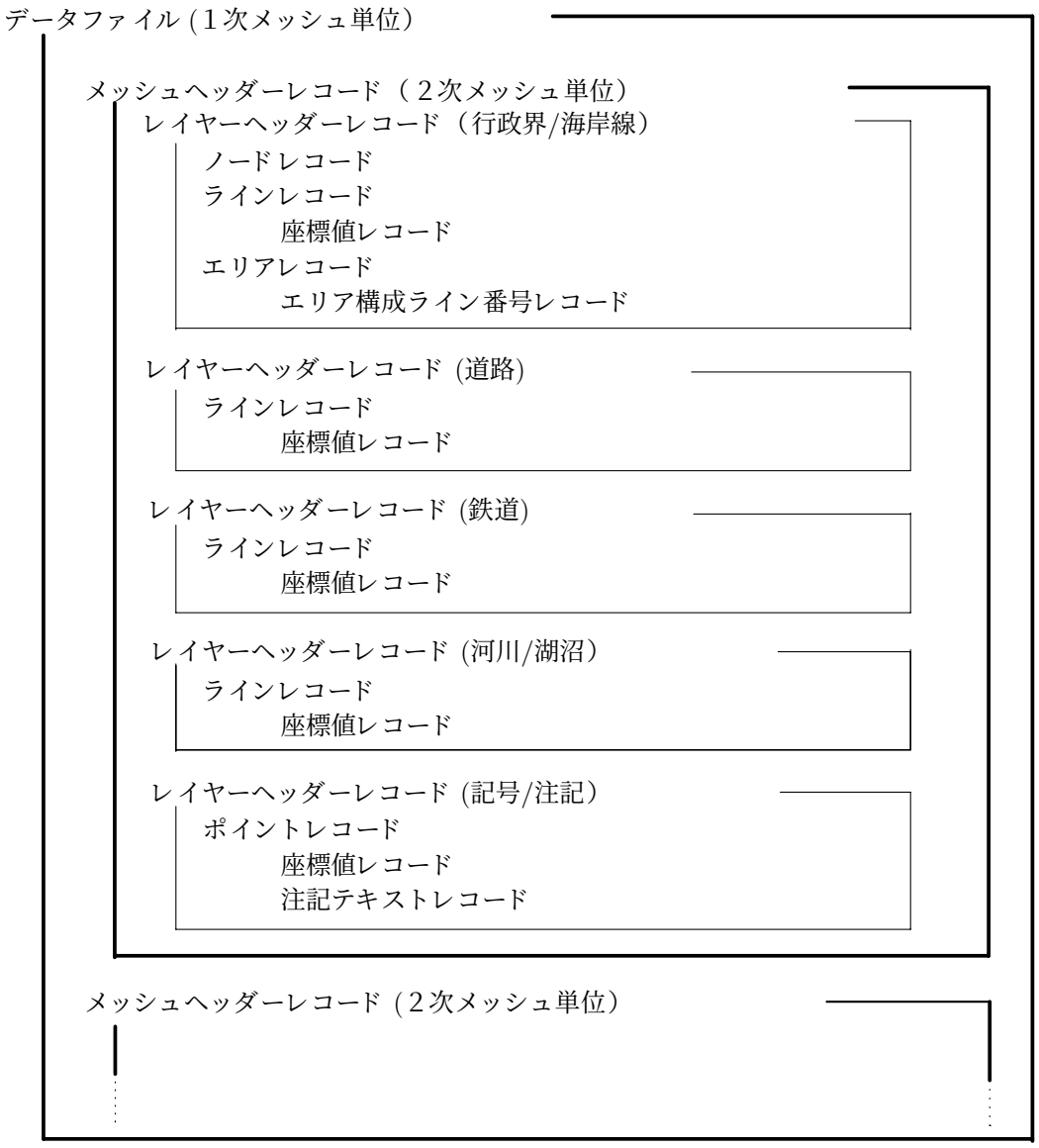


図 2.1: データファイルの概要

次にそれぞれのレコードの構成を示す。

1．メッシュ・ヘッダー・レコード

項　　目	開始	終了	形式*1)	内　容
レコードタイプ	1	2	Ａ２	”M”を記録
２次メッシュコード	3	8	Ｉ６	２次メッシュコード
図名	9	２８	Ｎ１０	当該２万５千分１地形図名
レイヤー総数	２９	３１	Ｉ３	当２次メッシュに含まれるレイヤー総数
ノード総数	３２	３６	Ｉ５	当２次メッシュに含まれるノード総数
ライン総数	３７	４１	Ｉ５	当２次メッシュに含まれるライン総数
エリア総数	４２	４６	Ｉ５	当２次メッシュに含まれるエリア総数
ポイント総数	４７	５１	Ｉ５	当２次メッシュに含まれるポイント総数
レコード総数	５２	５６	Ｉ５	当ヘッダーレコードを除いた当２次メッシュに含まれるレコード総数
空白	５７	７２	１６Ｘ	

例 ： M　303650 沖ノ鳥島　2　3　3　1　19

2. レイヤー・ヘッダー・レコード

項　　目	開始	終了	形式*1)	内　容
レコードタイプ	1	2	Ａ２	”Ｈ１”:構造化が行われていないレイヤー ”Ｈ２”:構造化が行われているレイヤー
レイヤーコード	3	4	Ｉ２	レイヤーの内容を表す １:行政界海岸線 ２:道路 ３:鉄道 ５:河川湖沼 ７:記号注記
ノード総数	5	9	Ｉ５	当レイヤーに含まれるノード総数
ライン総数	１０	１４	Ｉ５	当レイヤーに含まれるライン総数
エリア総数	１５	１９	Ｉ５	当レイヤーに含まれるエリア総数
ポイント総数	２０	２４	Ｉ５	当レイヤーに含まれるポイント総数
レコード総数	２５	２９	Ｉ５	当ヘッダーレコードを除いた 当レイヤー内のレコード総数
空白	３０	３０	ＩＸ	
レコード総数	３１	３４	Ｉ４	当レイヤーの最初の作成年月 西暦年の下２桁と月２桁
空白	３５	３５	ＩＸ	
レコード総数	３６	３９	Ｉ４	当レイヤーのデータ更新年月 西暦年の下２桁と月２桁
空白	４０	７２	３３Ｘ	

例：

```
H2    1     3     3     3     0    15    9410    9503
```

3．ライン (項目)・座標値レコード

- ラインレコード

項　　目	開始	終了	形式*1)	内　容
レコードタイプ	1	2	Ａ２	”Ｌ”を記録
レイヤーコード	3	4	Ｉ２	1:行政界海岸線 2:道路 3:鉄道 5:河川湖沼
データ項目コード	5	6	Ｉ２	下のライン項目コード 表参照
ライン一連番号	7	１１	Ｉ５	レイヤー内で当ラインが何番目に 位置するかを示す一連番号
ライン種別コード	１２	１７	Ｉ６	ライン種別コード 表参照
始点ノード番号	１８	２２	Ｉ５	
始点接続情報	２３	２３	Ｉ１	0:図葉内ノード 1:隣接図葉に接続する図郭線上のノード 2:隣接図葉に接続しない 図郭線上のノード
終点ノード番号	２４	２８	Ｉ５	
終点接続情報	２９	２９	Ｉ１	始点接続情報と同じコード
左側 行政コード	３０	３４	Ｉ５	ラインの向きに対して左側の行政コード *1)
右側 行政コード	３５	３９	Ｉ５	ラインの向きに対して右側の行政コード *1)
座標点の数	４０	４５	Ｉ６	当ラインを構成するＸＹ座標点の数 (始終点ノードを含む)
空白	４６	７２	Ｘ２７	

例：

```
L     1   5     1     0     10     101342199999     5     0     0
```

- ライン項目レコード

レイヤー	コード	データ項目
行政界海岸線 1	1	都府県境
	2	北海道の支庁界
	3	都市特別区界
	4	町村指定都市の区界
	5	海岸線
	9	図郭線
道路 2	1	高速道路及び自動車専用道
	2	一般国道
	3	主要地方道
	4	一般都道府県道
鉄道 3	1	ＪＲ
	2	公営鉄道
	3	民営鉄道
	9	未設定
河川湖沼 5	1	河川流路
	2	湖沼の水涯線

- ライン種別コード

レイヤー	コード	データ項目
行政界海岸線 1	0	確定境界線
	1	仮説境界線（陸部）
	2	仮説境界線（水部）
	9	図郭線
道路 2	0	地上
	1	地下・トンネル
鉄道 3	0	地上
	1	地下・トンネル
河川湖沼 5	0	通常の河川
	1	湖沼内の河川

- 座標値レコード

項　　目	開始	終了	形式	内　　容
１点目のＸ座標	１	６	Ｉ５	ラインレコードに続くレコードで、ラインを構成する座標点の数NだけＸＹ座標地のペアが記録される。座標値レコードの数は、　(Ｎ－１)／７＋１ である (余りは切り捨て)。最後のレコードに空きが生じる場合０が埋められる。
１点目のＸ座標	６	１０	Ｉ５	
１点目のＸ座標	１１	１５	Ｉ５	
１点目のＸ座標	１６	２０	Ｉ５	
⋮				
１点目のＸ座標	６１	６５	Ｉ５	
１点目のＸ座標	６６	７０	Ｉ５	
空　白	７１	７２	Ｘ２	

例：

```
5776  443  5786  443  5785  456  5775  456  5776  443
```

- エリアレコード

例：

```
A      113421        1   5780      450      1
```

- エリア構成ライン番号レコード

例：

```
-1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

- ポイント・レコード

例：

```
P   7   7        1   5780      451   0   1
```

- 注記テキストレコード

例：

```
01   4   5780   451                  00      沖ノ鳥島
```

2.2.2　データファイルの内容

データファイルの一部を以下に記す。

```
M 513100 小月                    5    8   53    5    1  215
H2 1    8   12    5    0  103 9410 9503
N  1 1    1    0   83 1 3    1    6  -12    0    0    0    0    0    0
N  1 1    2   45    0 1 3   -1   -6    7    0    0    0    0    0    0
N  1 1    3  657    0 1 3    2   -7    8    0    0    0    0    0    0
N  1 2    4 5617 4283 0 3   -2    4    5    0    0    0    0    0    0
N  1 1    5 1983    0 1 3    3   -9   10    0    0    0    0    0    0
N  1 1    6 1964    0 1 3   -3   -8    9    0    0    0    0    0    0
N  1 1    7 9913    0 1 3   -4  -10   11    0    0    0    0    0    0
N  1 1    8 876410000 1 3   -5  -11   12    0    0    0    0    0    0
L  1 5    1     0    11    213520199999     5 0  0
    0   83    2   78   56   45   43    2   45    0
L  1 5    2     0    31    403520199999   135 0  0
  657    0  659    5  306  274  367  398  352  409  376  463  261  553
  287  620  218  677  290  826  325  796  451 1066  416 1090  536 1347
  497 1374  506 1393  415 1463  463 1577  446 1588  472 1647  422 1704
  727 2189  853 2056  600 1630  649 1462 1264 1011 1881 2018 1659 2519
 1193 2793 1165 2856 1193 2880 1215 2891 1286 2893 1317 2904 1336 2920
 1366 2959 1371 2974 1377 2971 1387 2964 1402 2956 1418 2943 1425 2956
 1429 2964 1426 2972 1419 2977 1412 2982 1404 2988 1396 3002 1432 3079
 1441 3098 1458 3125 1430 3149 1528 3295 1648 3419 1702 3470 1672 3511
                                  .
                                  .
                                  .
```

これらのデータのうち、必要なものはレコード「M」に記されている島根県を示す第２次地域メッシュコードとレコード「Ｌ」の行の道路を表す「２」、さらに国道を表す「２」、つまり、「L 2 2」を示していす個所である。そして次の行から記されている座標値は左から（X座標１［5バイト］，Ｙ座標１［5バイト］)、(Ｘ座標２［5バイト］，Ｙ座標２［5バイト］)、…　の順に１行に最大７個記録されている。この個所をプログラムにて取得し、島根県内の各国道ごとの座標値データファイルを作成した。
なお、詳細についてはＪMCマップ CD-ROM 付属の説明書を参照せよ。

2.3　人口データ

2.3.1　データの記録方式

１：仕様

●レコード形式：　　固定長ブロック
●レコード長：　　２５３４バイト
●ブロックサイズ：　　１７７３８バイト
●文字コード：　　ＥＢＣＤＩＣ
●データのソート順：　　地域メッシュコードの昇順
●データの属性：　　キャラクター形式

２：コードの説明

●編成区分：
「３」… 平成７年国勢調査第１次・２次・３次基本集計及び従業地・通学地集計に係る編成であることを示す。

●地域メッシュ区画区分:
「２」… 第２次地域メッシュ区画の数値であることを示す。当該第２次地域メッシュ区画内の第３次地域メッシュ区画データの合算値である。
「３」… 第３次地域メッシュ区画の数値であることを示す。当該第３次地域メッシュ区画が人口集中地区地域である場合は、当該第４次地域メッシュ区画データの合算値である。
「４」… 人口集中地区地域における第４次地域メッシュ区画の数値であることを示す。

●市区町村数：
「９９」… 当該地域メッシュ区画に同定された市区町村数が９以上となる場合を示す。

●秘匿・合算識別符号：
「Ｘ」… 秘匿対象地域メッシュであることを示す。
「@」… 合算地域メッシュであることを示す。
「Ｚ」… ２次メッシュについて、ひとつの他の２次メッシュに合算される場合の秘匿地域メッシュであることを示す
「W」… ２次メッシュについて、複数の他の２次メッシュに合算される場合の秘匿地域メッシュであることを示す。

●秘匿措置について
一つの地域メッシュに表彰される世帯総数又は人口総数が極めて少ない場合、秘匿措置として当該地域メッシュの数値は単独では表彰せず、近接する地域メッシュの数値に合算した上で表彰している。ただし、人口総数（総数、男、女）、世帯総数（総数、一般世帯、施設等の世帯）、年齢（３区分）別人口（総数、男、女）、性比及び年齢（３区分）別人口の割合（総数、男、女）については、集計したままの原数値と、秘匿措置を講じた数値の両方を、表章している。データは秘匿措置を行なったデータをとり、人数 (世帯数) が「X」の個所は 0 として取得した。

2.3.2　データファイルの内容

データは以下のように地域メッシュコードの昇順にならんでいる。

表 2.1: データの記録順序

・ ・ 第１次地域メッシュ区画データ（５２３２）
第２次地域メッシュ区画データ（５２３２００）
第３次地域メッシュ区画データ（５２３２０００）
第４次地域メッシュ区画データ（５２３２０００１） ・ ・ 第４次地域メッシュ区画データ（５２３２０００４）
第３次地域メッシュ区画データ（５２３２０００１）
第４次地域メッシュ区画データ（５２３２０００１１） ・ ・ 第４次地域メッシュ区画データ（５２３２０００１４）
第３次地域メッシュ区画データ（５２３２０００２） ・ ・ 第３次地域メッシュ区画データ（５２３２００９９）
第２次地域メッシュ区画データ（５２３２０１）
第３次地域メッシュ区画データ（５２３２０１００） ・ ・ 第３次地域メッシュ区画データ（５２３２９９９９）
第４次地域メッシュ区画データ（５２３２９９９９１） ・ ・ 第４次地域メッシュ区画データ（５２３２９９９９４）
第１次地域メッシュ区画データ（５２３３）
第２次地域メッシュ区画データ（５２３３００）
第３次地域メッシュ区画データ（５２３３０００） ・ ・

次にデータのレコードを以下に記す。
（１１３バイト目以降は「就業者数」、「学生数」等の人口データ）

表 2.2: データレコード

項目	開始バイト数	終了バイト数
調査名コード	1	2
調査年	3	6
ブランク	7	7
編成区分	8	8
ブランク	9	9
地域メッシュコード	１０	１８
ブランク	１９	１９
地域メッシュ区画区分	２０	２０
ブランク	２１	２２
第２次地域メッシュ区画の 1/25,000 地形図名 JIS　漢字１２文字	２３	４６
ブランク	４７	４８
当該地域メッシュ区画に同定された 都道府県・市区町村コード	４９	９０
ブランク	９１	９１
秘匿・合算識別符号	９２	９２
ブランク	９３	９３
合算先 地域メッシュコード	９４	１０２
ブランク	１０３	１０５
人口総数 ・ ・	１０６ ・ ・	１１２ ・ ・

【実際のデータ例】

```
                                                                （人口総数）…
Z51995 3 53330293   3  母里          0132206338      …        338 …
Z51995 3 533310     2  松江          043230632201    …     135374 …
Z51995 3 53331000   3  松江          0132306         …        451 …
Z51995 3 53331003   3  松江          0132201         …         68 …
Z51995 3 53331004   3  松江          0132201         …        214 …
Z51995 3 53331007   3  松江          023220132305    …        547 …
Z51995 3 533310072  4  松江          023220132305    …        421 …
```

2.3.3　地域メッシュポリゴン

ブラウザで人口分布を視覚的に表示させるために人口データと共にその位置の 1km × 1km の四角形ポリゴンの座標値を生成する。なお、このセクションでは第２章 2.1「地域メッシュコードについて」の内容をふまえながら進めていく。

●作成方法

座標値系は前述した国道座標値を基準とし、取得した人口データのメッシュコードより座標値を求めていく。国道座標値は第２次地域メッシュ区画単位の左下を（0,0)、右上を（10000,10000）とする正規化座標となっており人口データは第３次地域メッシュ区画単位で取得したので 1km × 1km の人口メッシュポリゴンの座標値は ((0,0),(0,1000),(1000,1000),(1000,0)) からなる四角形となる。また、島根県の位置する第１次地域メッシュ区画は隠岐を除くと、

５３３１	５３３２	５３３３
５２３１	５２３２	５２３３
５１３１	５１３２	５１３３

と第１次地域メッシュ区画が３×３の正方形の形になっている。ここで一番左下を (0,0) とし、各区画単位での絶対座標値をメッシュコードから求める

・第１次地域メッシュ区画単位の場合
上図より５○３１の「○」の部分が y 成分、５１３○の「○」部分が x 成分を表している。よって、求める座標値は
(○-1) × (10000 × 8)

・第２次地域メッシュ区画単位の場合
同様に５１３１○３の「○」の部分が y 成分、５１３１２○の「○」の部分が x 成分を表しているので求める座標値は
○× 10000

・第３次地域メッシュ区画単位の場合
第２次地域メッシュを１０等分したものなので、求める座標値は
○× 1000

よって求めるポリゴンの座標値は、第１次地域メッシュ区画単位の x 成分を x1, 第２次を x2, 第３次を x3 とすると、
x = (x1-1) × (10000 × 8) + (x2 × 10000) + (x3 × 1000)

y 成分についても同様。よって求めるポリゴンの座標値は、
Polygon() = ((x,y),(x,y+1000),(x+1000,y+1000),(x+1000,y))

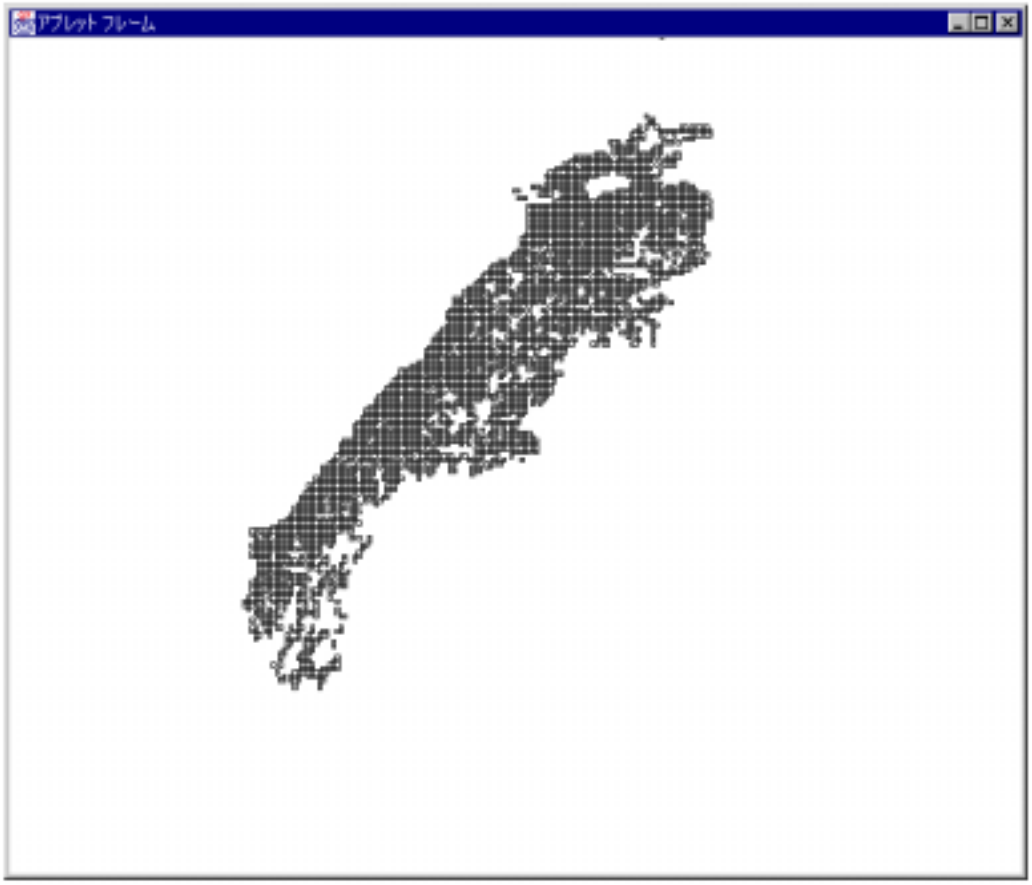
の式で求められる。

EX. 例として 52314365 のメッシュポリゴン座標値を求める。

(y1,y2,y3) = (2,4,6)
(x1,x2,x3) = (1,3,5)
y = (2-1) × (10000 × 8) + (4 × 10000) + (6 × 1000) = 126000
x = (1-1) × (10000 × 8) + (3 × 10000) + (5 × 1000) = 35000

よって、求める座標値は
Polygon() = ((35000,126000),(35000,127000),(36000,128000),(36000,126000)) となる。

これらの方法で求めた島根県の人口分布ポリゴンを以下に示す。



DB には約４０００弱の各々のポリゴン座標値に属性データとして各人口データ (総人口、就業者数、等) を付加し、格納する。

第3章　DBの操作

使用したDBはHITACHIソフトウェアの最新版「HiRDB ver 6] とHiRDBで空間検索システムを構築できるようにするプラグイン製品の最新版「HiRDB Spatial Searc Plug-in 03」(以降、空間検索プラグインと略記する) を用いた。この空間検索プラグインの最新版には新機能として、ある線分に対する左右帯域幅を発生させてその中に含まれる図形を検索するというものが追加されたので本研究ではこの空間検索プラグインの新機能を用いてどのような用途のシステムを構築できるかを考えた。

3.1　データの格納

生成した国道座標値データ、人口データ、基準地域メッシュのポリゴン座標値データをDBに格納するのだが、１つ１つの座標値、また4000弱ある人口データをSQL言語でわざわざ格納するのは効率が悪すぎるので空間検索プラグイン特有の一括登録コマンド「pdload」を使って表に登録を行う。

3.1.1　コマンド形式

コマンド形式は表に格納するデータを「文字列データ」か「バイナリデータ」であるかによって変化するが今回は「文字列データ」の場合で表への一括格納を行ったのでそのコマンド形式を示す。

コマンド形式：
pdload -c 列構成情報ファイル名　テーブル名　制御情報ファイル名

-c　…　文字列データで格納するときのオプションを示す。
列構成情報ファイル　…　格納する表に対する列名の構成情報ファイル。
テーブル名　…　格納の対象となる表
制御情報ファイル　…　格納するデータファイルのパス名を指定するファイル

●使用例
１．格納する表の作成

国道データを格納する表の場合を述べる。まず以下のSQL文で表を作成する。

```
CREATE TABLE KOKUDOU(ID INTEGER,NAME VARCHAR,GEO_CLMN GEOMETORY);
```

ID　…　国道のID
NAME　…　国道の名前
GEO_CLMN　…　座標値データ

２．列構成情報ファイルの作成

ファイル名は info.txt で内容を以下に示す。

```
ID
NAME
GEO_CLMN,func=(GeomFromText,param=varchar,param=integer,param=integer)
```

つまり、次に示すデータファイルの中のデータを格納する列を指定するファイルである。

３．データファイルの作成

ファイル名は KOKUDOU __ DATA.csv。内容を以下に示す。

```
1,R186,"LINESTRING( 97576 -90000,97302 -90260,97213 -90458, …
2,R187,"LINESTRING( 75809 -40000,75755 -40325,75631 -40523, …
                                .
                                .
8,R9,"LINESTRING( 60421 -50000,60570 -50386,61473 -51420, …
```

各列に格納するデータは「,」で区切られており、１つめの区切りは ID、２つめは国道名を格納する列 NAME、３つめが座標値を格納する列 GEO __ CLMN とそれぞれなっている。

4. 制御情報ファイルの作成

ファイル名は control.text。内容は以下のようになっており、データファイルのパス名を示す内容となっている。

```
source C:￥LINE_PDDATA￥KOKUDOU_DATA.csv
```

５．コマンドの実行

DOS プロンプトで各ファイルが保存してあるフォルダに移動し、コマンドを実行する。

```
C:￥LINE_PDDATA>pdload -c info.txt KOKUDOU control.txt
```

●以下にデータが格納された国道と人口の表のイメージ図を示す。

ID	NAME	GEO-CLMN
1	R186	__/
2	R187	「 ¯
・	・	・
・	・	・
3	R9	/ ¯

MESH-CODE	JINKOU		STUDENT	GEO-CLMN
51313688	1	…	0	□
51513698	8	…	0	□
・	・	…	・	・
・	・	…	・	・
54333300	87	…	4	□

3.2 データの検索、抽出

3.2.1 UAP について

次にデータを格納した表より JAVA サーブレットからの引数をもとにした検索を行う過程を示す。(JAVA サーブレットから UAP への値の受け渡し方法は第 4 章　4.3「UAP への値の受け渡し」を参照のこと。) データの検索や抽出は UAP と呼ばれる埋め込み型プログラミングによって行う。これは DB を操作する言語、SQL 言語を C 言語、もしくは COBOL 言語に直接記述し DB とのやりとりを行う方式である。本研究では C 言語を用いて国道座標値を返す UAP(以降国道 UAP)、地域メッシュポリゴンの座標値と人口データを返す UAP(以降人口 UAP) をそれぞれ作成した。

3.2.2 アルゴリズムの説明

作成した UAP の大まかなアルゴ リズムを以下に示し、以降詳細に説明をする。
JAVA サーブレットからの引数を main 関数の標準入力として受け取る。
得た引数をもとに SQL 文を生成する。
SQL 文をもとに検索を行い、検索結果を標準出力として出力する。

データの受け取りに関して

データの受け取りは main 関数の標準入力から行う。標準入力に得る main 関数の引数は main(int argc,char **argv[]) とし、引数には国道名 (複数の可能性有り)、人口データ名、左右帯域幅と最低でも 3 つの引数が入力される。よって argv[] の各要素には以下のような値が入力される。

●国道 UAP の場合：

JAVA からの UAP 実行形式の例：KOKUDOU-UAP.exe R261 R314 R54 R9
argv[1]〜argv[argc-1] ･･･　国道名 ･･･ R261 R314 R54 R9

●人口 UAP の場合：

JAVA からの UAP 実行形式の例：JINKOU-UAP.exe R261 R314 R54 R9 JINKOU 3000
argv[1]〜argv[argc-3] ･･･　国道名 ･･･ R261 R314 R54 R9
argv[argc-2] ･･･　人口データ名 ･･･ JINKOU
arvcargc-1] ･･･ 帯域幅 ･･･ 3000

国道 UAP では argv をそのまま SQL 文生成の引数として用いるが、人口 UAP では国道の線分を構成する座標に対する帯域幅と人口データが検索条件となるのでソースでも引数を異なる変数に格納する。

●国道 UAP のソース：

```
int main(int argc, char **argv[])
{
int l;
for(l = 1; l <= argc-1; l++){
Kokudou[l] = argv[l];
}
                .
```

●人口 UAP のソース：

```
char **Kokudou[],**Jinkou[],**Buffer[];
int main(int argc, char **argv[])
{
int l;
for(l = 1;l <= argc-3;l++){
  Kokudou[l] = argv[l];
}
Jinkou[0] = argv[argc-2];
Buffer[0] = argv[argc-1];
              .
```

SQL 文の生成

●国道 UAP の場合：

SQL 文を格納する以下のような構造体を定義する。

```
struct{
long Size;
char meirei[10000];
}CMD_KOKUDOU
```

CMD_KOKUDOU.meirei[] には SQL の命令文、Size には配列のサイズ数を入れる。以下はソース。

```
strcat(CMD_KOKUDOU.meirei,"SELECT AsText(GEO_CLMN,1) FROM KOKUDOU WHERE NAME='");
strcat(CMD_KOKUDOU.meirei,Kokudou[i]);
strcat(CMD_KOKUDOU.meirei,"' WITHOUT LOCK NOWAIT ");
CMD_KOKUDOU.Size = sizeof(CMD_KOKUDOU.meirei);
```

strcat で CMD_KOKUDOU.meirei に命令文を入れていく。ここでの命令は「列 NAME が
Kokudou[i] であることを満たす (WHERE NAME='a[i]')GEO_CLMN 列のデータを検索し、文字列で
出力 (AsText(GEO_LMN)」
という命令である。つまり、指定した国道名の座標値を検索する命令である。

●人口 UAP の場合：

人口 UAP の場合は検索するコマンドが、3つある。「選んだ国道座標値の検索 (上記のコマンドと同じ)」、「国道座標値に対する左右帯域幅 $x$km に含まれる基準地域メッシュポリゴンの座標値の検索」、「選択された人口データの検索」。
ソース:

```
strcat(CMD_KOKUDOU.meirei,"SELECT AsText(GEO_CLMN,1) FROM KOKUDOU WHERE NAME='");
strcat(CMD_KOKUDOU.meirei,Kokudou[i]);  // ····· 国道名が入った配列
strcat(CMD_KOKUDOU.meirei,"' WITHOUT LOCK NOWAIT ");
CMD_KOKUDOU.Size = sizeof(CMD_KOKUDOU.meirei);
                    .
                    .
strcat(CMD_JINKOU.meirei,"SELECT AsText(GEO_CLMN,1) FROM JINDATA WHERE
                   INTERSECTSIN(GEO_CLMN,RegionBuffer(GeomFromText('LineString(");
strcat(CMD_JINKOU.meirei,Zahyouti[j]); // ····· 国道座標値が入った配列
strcat(CMD_JINKOU.meirei,")',0,1),");
strcat(CMD_JINKOU.meirei,Buffer[0]);  //  ····· 左右帯域幅が入った配列
strcat(CMD_JINKOU.meirei,")) IS TRUE WITHOUT LOCK NOWAIT");
CMD_JINKOU.Size = sizeof(CMD_JINKOU.meirei);
```

この検索は
「国道の線分に対する　　(GeomFromText('LineString(Zahyouti [j])',0,1)···(i))
帯域幅の大きさ Buffer[0]　(RegionBuffer((i),Buffer[0]) ··· (ii))
に含まれるポリゴンの検索」(INTERSECTSIN(GEO_CLMN,(ii))) である。

```
                    .
                    .
strcat(CMD_JINDATA.meirei,"SELECT ");
strcat(CMD_JINDATA.meirei,Jinkou[0]);     // ····· 人口データが入った配列
strcat(CMD_JINDATA.meirei," FROM JINDATA WHERE
                     INTERSECTISN(GEO_CLMN,RegionBuffer(GeomFromText('LineString(");
strcat(CMD_JINDATA.meirei,Zahyou[j]);     // ····· 国道座標値が入った配列
strcat(CMD_JINDATA.meirei,")',0,1),");
strcat(CMD_JINDATA.meirei,Buffer[0]);     // ····· 左右帯域幅の大きさが入った配列
strcat(CMD_JINDATA.meirei,")) IS TRUE WITHOUT LOCK NOWAIT");
CMD_JINDATA.Size = sizeof(CMD_JINKOU.meirei);
```

この検索は上記の検索条件と同じであるが、取り出す値は Jinkou[0] で指定した人口データの列に対する検索結果である。

検索結果の出力

検索結果の出力は printf 文による標準出力で行う。JAVA サーブレット側ではこの出力結果を受け取り、値を画像等に加工する。以下に出力結果を示す。

```
国道 UAP：          人口 UAP：
97576 -90000       93000 -100000
97302 -90260       93000 -101000
97213 -90458       94000 -101000
96988 -90592       94000 -100000
96892 -91062       J101E              … 人口数
97086 -91101
       ・           94000 -100000
       ・           94000 -101000
90280 -106408      95000 -101000
90039 -106469      95000 -100000
       ・           J24E               … 人口数
       ・                 ・
       ・                 ・
```

第4章　JAVA サーブレットの操作

4.1　JAVA サーブレットとは

4.1.1　役割

Web 上でのサーバーとのやりとりの手法に JAVA サーブレットと CGI プログラムがある。これらが Web サーバ上で動くことによってクライアント (ブラウザ等) から来たリクエストをデータベースやその他のアプリケーションに伝えたり、逆にＤＢからの検索結果をクライアントに送ったりする。つまり、サーブレットや CGI プログラムは Web と一般アプリケーションやＤＢとのミドルウェアといえる。

4.1.2　なぜサーブレットか

JAVA サーブレットは従来のＣＧＩ技術に比べて、優位な点が多い。以下にその点をあげる。

1. 効率的

   従来の CGI では HTTP リクエストがくるたびに新たなプロセスが動き出し、小さな CGI プログラムの場合には実行時間のほとんどがプロセスの始動で占められる。よってリクエストごとにオペレーティングシステムのプロセスをつくって動かすという重いオーバーヘッドが存在するが、サーブレットの場合だと JAVA 仮想マシンが唯一のプロセスとして常に動いており、個々のリクエストはスレッドによって処理されるので効率てきである。

2. 強力

   通常の CGI には難しい、または不可能な機能をサーブレットはサポートしている。例えばサーブレットは Web サーバと直接対話できるが、CGI ではそのために固有の API を使わなければならない。Web サーバと直接対話できると、例えば相対 URL をパス名に翻訳する処理等が簡単になる。

3. 可搬性

   サーブレットは JAVA 言語で書くので JAVA の標準 API を使用する。よってプラットホーム非依存なのでどんなサーバ上でも動く。

4. 開発しやすい

   サーブレットは高級な基礎的機能を豊富にもっている。例えば HTML のフォームデータの構文解析、HTTP ヘッダの読み取りと設定、クッキーの処理、セッションの管理等。

この他にも安全面や経済的である等の利点がある。これらの面から本研究では Web 上でのやりとりに JAVA サーブレットを用いることにした。

4.2 HTMLからの引数の受け取り

ユーザーが項目を選択し、ブラウザの送信ボタンが押されてから指定した引数がサーブレットに渡されるまでの過程を示す。なお、説明文中の [ ] はプログラム中でのインスタンス名、または変数名を示す。

ユーザーがフォームデータ (国道名、左右帯域幅、人口データ) を設定する。各フォームデータはフィールド名に設定した属性になる。
(例:国道名 = Kokudou, 左右帯域幅 = Buffer, 人口データ = jindata)
送信ボタンを押すと設定したフォームデータが URL に添付され、doGet メソッドに渡される。
doGet メソッドの引数、HttpServletRquest[req] の getParameter() メソッド、1つのフィールドに複数のフォームデータがある場合は getParameterValue() メソッドを用いて各フォームデータを変数に格納する。

HTML 文

```
<INPUT TYPE="checkbox" NAME="Kokudou" VALUE="R186" >R186 |
<INPUT TYPE="checkbox" NAME="Kokudou" VALUE="R187" >R187 |→ Kokudou フィールド
<INPUT TYPE="checkbox" NAME="Kokudou" VALUE="R191" >R191 |
                          .
<SELECT NAME="Buffer">                                    |
<OPTION VALUE="500">500m                                  |→ Buffer フィールド
<OPTION VALUE="1000">1Km                                  |
<OPTION VALUE="1500">1.5Km                                |
                          .
<INPUT TYPE="radio" NAME="Jinkou_data" VALUE="jinkou" CHECKED>総人口数
<INPUT TYPE="radio" NAME="Jinkou_data" VALUE="worker">就業者数
                          .
                          .
```

このとき選択された個所の VALUE の文字列が各フィールドのフォームデータとして doGet メソッドに渡される。(,)

```
public void doGet(HttpServletRequest req,
    HttpServletResponse res)
                      .
                      .
  String K_Paramater[] = req.getParameterValues("Kokudou");
  String J_Paramater   = req.getParameter("Jinkou_data");
  String Buffer        = req.getParameter("Buffer");
                      .
                      .
```

単数のフォームデータなら getParameter で、複数のフォームデータがあれば配列の各要素に格納される。( )

4.3 引数のUAPへの引渡し、検索結果の受け取り

次はHTMLから送られてきた引数のＵＡＰへの引渡しとＵＡＰから受けとった検索結果を受け取る方法を説明する。ＵＡＰからの検索結果の受け取りはJAVAの外部プログラムを実行する一連のメソッド操作を利用した。ここではその手法について説明する。なお[　]の中は下の例題プログラム中に設定したインスタンス名または変数名を示す。

java.lang.Runtimeクラス[rt]を呼び出し現在の実行環境を表すオブジェクトを取得する。
Runtimeオブジェクト[rt]のexecメソッドを実行したいコマンドを引数に呼び出す。ここではＤＢから国道座標値、地域メッシュポリゴンの座標値、人口数を返すUAPの実行ファイルを指定する。また引数としてHTMLから受け取った値を設定する。またexecメソッドの呼び出しによってコマンドの実行結果を受け取るProcessオブジェクト[prcs]が呼び出される。
Processの入力ストリーム[prcs.getInputStream]をInputStreamReader[isr]につなぐ。
入力ストリーム(isr)から1文字ずつ読み込む。

```
import java.io.*;
   public class OutZahyou{
     public sttic void main(String arg[]){
       String K_NAME = "R9 jikou 3000";
       // HTMLからの引数(例えば国道をＲ９, 人口データを総人口, 帯域幅を3Kmとする)
       try{
        Runtime rt = Runtime.getRuntime();
        //
        Process prcs =rt.exec("C:\GEO_KENSAKU\Debug\Out_Kokudou.exe " + K_NAME);
        //
        InputStreamReader isr =
           new InputStreamReader( prcs.getInputStream() );
        //
        char str[] = new   char[5];
        while(isr.read(str[],0,1)
          System.out.println(str[0]);
        //
       }
       catch(IOException ioe) System.out.println(ioe);       //  例外処理
    } }
```

実行結果 :

```
10345 -34895
10959 -37383
11200 -33455
      ・
      ・
```

つまり、ここでの操作はUAPのprintf文で出力された文字(検索結果)をJAVAが受け取り、国道の座標値として画像表示のための引数とすることである。

4.4 画像表示

UAP から得た座標値をもとに画像を生成し、ブラウザに表示するまでの過程を以下に示す。なお [ ] はプログラム中でのインスタンス名、変数名を示す。

java.awt.image.BufferedImage クラス [image] を呼び出し、そこから得られた Graphics[g] の描画メソッド getGraphics() を使い描画対象を作成する。

作成する描画対象は

1．国道を描画する線分。
メソッドは drawPolyLine(int x[],int y[],int NumberOfPoints)。
x[],y[] には線分を構成する点列を格納し、NumberOfPoints にはその点の数をいれる。

2．基準地域メッシュを描画するポリゴン。
メソッドは drawPolygon(int x[],int y[],int NumberOfPoints)。
x[],y[] にはポリゴンを形成する点列を格納し NumberOfPoints にはその点の数をいれる。

また、人口数に応じた色付けを if 文+setColor(Color) で行なう。

3．各人口データの範囲を文字列で描画。
メソッドは drawString(String str)。
str に描画したい文字列を格納する。

doGet メソッドの引数、HttpServletResponse[res] の setContentType() メソッドでブラウザに返す値の形式を設定する。ここでは JPEG 形式で返すので setContentType("imge/jpeg") とする。

ブラウザのストリームに出力形式でつなぐために ServletOutputStream クラス [out] を doGet メソッドの引数、HttpServletResponse[res] の getOrtputStream() として行なう。
そして生成した画像を JPEG 形式で表示するためには BufferdImage での画像情報は内部的なフォーマットで管理されているので JPEG 形式にエンコードする必要がある。
そのメソッドは JPEGImageEncoder クラス [encoder] の encode() 。

次にプログラムを示す。

```
public doGet(HttpServletReqest req,HttpServletResponce res)
                              .

      BufferedImage image = new BufferdImage(800,700,BufferedImage.TYPE_INT_RGB)
          Graphics g = image.getGraphics();
          //
          g.drawPolyLine(Kokudou_x[],Kokudou_y[],Kokudou_Num_Points);

          if(Jinkou < 0 & jinkou >100) g.setColor(SystemColor.blue); if ··· ;
          g.drawPolygou(Jinkou_x[],Jinkou_y[],Jinkou_Num_Points);

          g.drawString("人口",600,600); g.drawString("0〜100",600,620); ···
          //
          res.setColorType("image/jpeg");
          //
          ServletOutputStream out = res.getOutputStream();
          JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(out);
          encoder.encode(image);
          //
                               .
                               .
```

第5章　本研究からの展望

今回作成したシステムの試作で基本的なものを作成してみたが、これらの空間検索技術と DB と Web とのやりとりを用いて DB から得られた人口データを加工した数値、例えば「国道９号線沿いの左右帯域幅１ km の総人口に対する就業者数比率」だとか「国道 54 号線沿いの左右帯域幅 2km の総人口に対する 65 歳以上の老齢者比率」等の表示、またこれらのデータのヒストグラフ、円グラフ等を表示できる機能を付加すれば様々な視点からのデータをブラウザ上で誰でも取得することができ、意思決定支援に役立てることができると思う。
また残念であったのが問題点として「ある１点から半径〜km 内にある空間データ」という比較的簡単な検索ならば検索時間は１秒もかからないのだが今回主とした「ある線分の左右帯域幅〜km 内にある空間データ」という帯域幅に関する検索はデータが大きい線分であると最大で２０分弱かかるという欠点があった。考えられる理由は前者の検索がある１点から検索範囲を発生させているのと後者は線分を構成する点１つ１つに対して検索範囲を発生させ、後に検索を行っているのではないかと考えた。この考えに至る理由として試しに国道を構成する点１つ１つに対し円検索を行ったものと左右帯域幅で検索を行ったものとの時間を比較したら結果が返ってくるまでそれほど変わらない検索時間であった。ただ前者の検索であると人口データが重なって検索結果が返ってきてしまうが後者だと左右帯域幅を生成してから検索を行っているようなので人口数が重ならず、正確なデータを得ることができる。
この問題が解決されればスムーズにデータ検証を行うことが可能となるので更なる技術の発展に期待したい。

引用・参考文献

[1]　総務庁統計局　刊行
　　平成７年国勢調査地域メッシュコード　　（平成７年）

[2]　（財）日本地図センター　刊行
　　ＪＭＣマップ　CD-ROM 添付ファイル　　（平成１０年）

[3]　日立製作所ソフトウェア事業
　　「HiRDB Spatial Search Plug-in Version ３」（平成１３年）

[4]　日立製作所ソフトウェア事業部
　　「HiRDB Version5.0　コマンドリファレンス」（平成１１年）

[5] Michael Stonebreaker 著
「オブジェクトリレーショナル DBMSs」
太田佳伸　訳
株式会社ビー・エヌ・エヌ
International Thomson Publishing Japan （平成８年）

[6] 株式会社ピアソン・エデュケーション
ＳＱＬ９２／９９標準リファレンスブック

[7] Peter van der Linden 著
「ジャスト JAVA2 -オブジェクト指向プログラミングと JAVA
中田秀基 訳
株式会社アスキー （平成１２年）

[8] Windows プログラミング愛好会 著
「Java500 の技
株式会社技術評論社 （平成１３年）

[9] Marty Hall 著
「コア・サーブレット ＆ JSP -JAVA サーバ技術による Web 開発-」
岩谷 宏 訳
ソフトバンク パブリッシング株式会社 （平成１３年）

謝辞

本研究において最後まで暖かく丁寧な指導、また助言をしてくださった田中章司郎先生に深く感謝の意をここに表します。そして、共に研究をしてきた田中研究室の仲間、たくさんのアドバイスを頂いた先輩方にここに感謝の意を表します。