データベースシステム 128－13
（２００２．７．１７）

質問シートによるXMLデータ検索

北川文夫
岡山理科大学
kitagawa@mis.ous.ac.jp

概要

我々は、野外調査で得られたデータを地図上に表示させるためのシステムを研究開発してきた。このシステムは調査データが階層化したツリー構造をしていることを利用して、XMLのデータ定義 (DTD とタグのスケルトン) を内部で生成してデータスキーマとして用いている。このスキーマに従ってシステムが生成する入力インタフェースを質問シートと呼ぶ。質問シートでは、データの入力から修正、削除、検索が行なえる。質問シートは入力データをスキーマのテンプレートに従ってタグ付けされ XML 文書形式になりシステムに渡る。また、蓄積されるデータはリレーショナルデータベースのテーブル群に格納される。本稿では、質問シートからの検索、特に RDB 上のテーブルに対する検索の SQL 生成等に関して報告する。

Querying XML Data using a Query Sheet

Fumio KITAGAWA
Okayama University of Science

abstract

We have been researching and developping a system, which can draw some markers on a map according to each data of field work. This system adopts XML(DTD and tag-skeleton) as data schema, because each data of field work has a tree structure. According to this schema, the system generate a window having attribute name and text box associated to each tag. We call this window as a query sheet. By using query sheet, we can do, data entry, data update, data delete, and data query. On the other hand, this system adopts a RDB as a strage of data. Then we show how to map the tree structure to RDB, and haw to convert the query of query sheet to SQL.

1　はじめに

我々は、様々な野外調査データに対して、同一の地図上に調査地点を表示するシステムを開発してきた。ここで扱う野外調査データとは、植生調査や考古学調査などの調査データである。これらの調査地点を表示することに加え、大域的に調べられたメッシュデータとも組み合わせられれば、より発展的なデータの解析が行なえると考えられるので、本システムの対象データに加える。

野外調査データは、研究分野や調査者の学派などによっても調査する項目が異なることがある。従って、システムにはじめから調査項目が決められていることは好ましくないので、項目を登録できるようにしてある。野外調査の項目は単純な階層構造をしているので、そのまま XML のツリー構造で表現可能である。また、XML で構造を定義することにより、RDB へのテーブル分割やインタフェース画面の生成が容易に行なえるので都合がよい。システムに蓄積するデータ構造を XML のまま行なうと、検索に時間がかかるなどの不利な点も考えられたので、データを RDB にすることにした。

データの検索は、データ項目の構造を定義した XML から生成される質問シートと呼ぶデータ入力用ウインドウから行なう。質問シートはデータ項目名に対してテキストボックスが対応したウインドウであり、入力データは、XML のタグスケルトンの該当個所に入力された形で処理される。この XML 文書から、RDB を検索するための SQL を生成し、実行される。この質問シートは複数枚を組み合わせて論理演算ができるようになっている。

本稿では、複数の質問シートからの検索方法の考察を中心に、プロトタイプの試作と合わせて報告する。

2　システム概要と質問シート

質問シートを説明する前に、システムの中で質問シートの位置付けが分かるようにシステム概要を解説する。

2.1　システム概要

システムのデータ処理の流れを図 1 に示す。ユーザが設定したデータ項目は DTD と XML スケルトンに変換され、定義情報として DB に登録される。同時に、定義情報に従ったテーブル群を DB に作成する。登録された定義情報を基に質問シートが動的に生成され、データの入力や削除、検索に利用される。

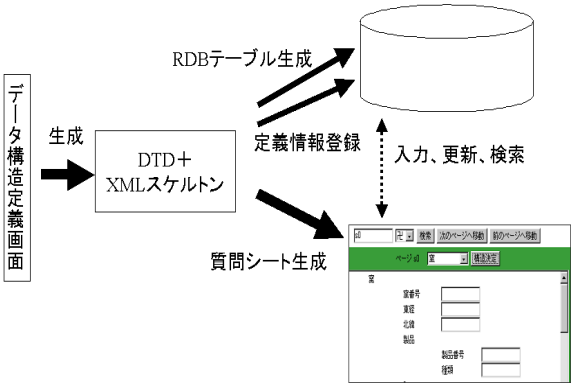

図 1 データ処理の流れ

ここでは主にデータ項目作成と定義情報とテーブル生成について説明する。本システムでは、データの項目と構造 (階層順) はユーザが入力する。具体的にはインデント付けが可能な構造定義画面によりフラットな項目か、子要素なのかをユーザが作成できるようになっている。(図 2 参照)

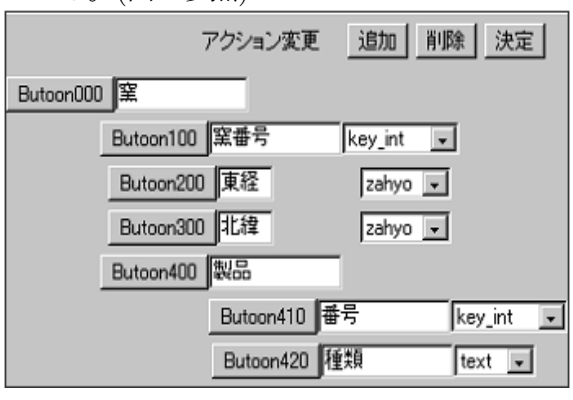

図 2 データ構造登録画面

ここの先頭のテキストボックスは定義情報自身の名称を入力する。例えば「窯」や「遺跡」などが入る。２番目は各調査地点の識別名が入る。例えば遺跡名称や発掘地点の番号

などである。続く東経と北緯はあらかじめ項目として入っている。以上の４項目は必須項目で初めから表示されている。調査者が必要とする項目はButton000をクリックし次に追加ボタンを押せば、Button400がButton300の下に表示され項目名とその属性(文字か数値かなど)が書き込める。同様にどこかの項目に子要素を作るときはその項目のボタンを押して追加を押せばよい。

このようにして登録されたデータ構造定義は、DTD付きのタグスケルトンとして登録される。(図3参照)

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<!DOCTYPE 窯 [
<!ELEMENT 窯 (窯番号, 東経, 北緯, 製品)>
<!ELEMENT 窯番号 (#PCDATA)>
<!ATTLIST 窯番号 datatype CDATA #FIXED "key_int">
<!ELEMENT 東経 (#PCDATA)>
<!ATTLIST 東経 datatype CDATA #FIXED "text">
<!ELEMENT 北緯 (#PCDATA)>
<!ATTLIST 北緯 datatype CDATA #FIXED "text">
<!ELEMENT 製品 (番号, 種類)>
<!ELEMENT 番号 (#PCDATA)>
<!ATTLIST 番号 datatype CDATA #FIXED "key_int">
<!ELEMENT 種類 (#PCDATA)>
<!ATTLIST 種類 datatype CDATA #FIXED "text">
]>
<窯>
  <窯番号></窯番号>
  <東経></東経>
  <北緯></北緯>
  <製品>
    <番号></番号>
    <種類></種類>
  </製品>
</窯>
```

図3 DTDとタグスケルトン

このデータ項目定義情報が登録されると、この情報に従ってRDBのテーブルが作成される。例えば上記のタグスケルトンからは、

```
窯 (窯番号    integer primary key,
    東経      text,
    北緯      text)

製品 (番号    integer primary key,
      種類    text,
      窯番号  integer)
```

の2つのテーブルが生成される。ここでのデータタイプは項目定義時に指定してDTDの属性タグに書かれたものが使われる。またタグスケルトンからのテーブル生成規則は子要素の有る無しで判定している。このようにテーブルを分割するのは、自然に第3正規形を満たすテーブル群にできるためである。分割したテーブル群に対して、元の関連(XMLで示された木構造)を維持するために、テーブル間の関連情報を持たせる。1つは、階層が下位のテーブルが上位のテーブルのキー属性を持つことである。SQLのCREATE文で外部キー制約を付けることが望ましいが、現状では上位のキー属性を持つところまでである。こうすることで全ての木のインスタンスが構成できる。しかし、いちいちタグスケルトンをスキャンして結合属性を調べるのは効率が悪いので、結合属性情報をSQLの結合条件式の形で保存しておく。例えば、上記の例に対しては、

窯.窯番号=製品.窯番号

という情報を保存しておく。この結合条件とタグスケルトンと定義名称の3情報が定義テーブルに保存される。

定義(名称，XML，結合属性)

なお、テーブル名、属性名はデータ項目登録時やデータ検索時の名前の混乱を避けるために、上記の例とは多少異なり、次の規則で命名することにしている。

テーブル名 テーブルは子ノードを持つノードが1つのテーブルを生成するので、そのノード名をルートからのフルパスで表現する。ただし、ノードの区切りは“/”では無くて“_”であり、ルートノードの先頭には適用しない。

属性名称 テーブル名称と同様に“_”で区切ったフルパス名を付ける。また、上位テーブルとの結合属性はフルパスではないノード名と上位のキー属性とを“_”で連結した名称にしている。

先の例では次に示す名称になる。(データタイプは省略する)

窯 (窯_窯番号,
　窯_東経,
　窯_北緯)

窯_製品 (窯_製品_番号,
　　窯_製品_種類,
　　製品_窯_窯番号)

2.2　質問シート

以上のように、データ項目の設定からタグスケルトンと RBD のテーブルが生成されると、これらの項目に対応したデータ入力や更新、検索を行なうためのインタフェースが必要になる。これは、ユーザが入力した項目の階層構造で表示するのが好ましい。従って、タグスケルトンの階層構造に応じてインデントを付けたテキストボックスにノード名のラベルを付けて表示する。これを質問シートと呼び、システムがタグスケルトンから自動生成する。質問シートは、テキストボックスを変数と見なしたとき、タグスケルトンに値として変数が入ったものと同等と考えることができる。ユーザから見ても、最初に設定したデータ項目の階層構造がそのまま使われているように見える。

質問シートは、全項目を 1 つのシートで表示するものと、階層ごとに別々のシートにするものの 2 タイプある。データの検索では全項目のシートを、入力・更新では階層ごとの複数シートを使う。それは、入力・更新では特定のノードに繰返し項目があることが考えられるためで、それらを分けたシートにした方が便利と考えられるからである。検索での条件式ではそのようなことは必要ないと考えられる。

(1) 階層ごとの質問シート

データの入力や更新などに用いる質問シート。階層ごとといっても、各層がばらばらに表示したのでは分かりにくいし、指定方法も難しくなる。そこで上位層には下位層を開くためのボタンを配置してある。このようにすることでルートノード (= ユーザが入力したデータ項目定義名) から順に下位のシートを開いてゆくことができる。一つのシートは 1 つのテーブルに対応しているので、この単位で入力や更新が行なわれる。ただし、結合属性は自動的に上位シートに入力したキー属性が使われる。図 4 は上が上位層で「製品」右のボタンを押すと、下位層に対応した下のシートが表示する。

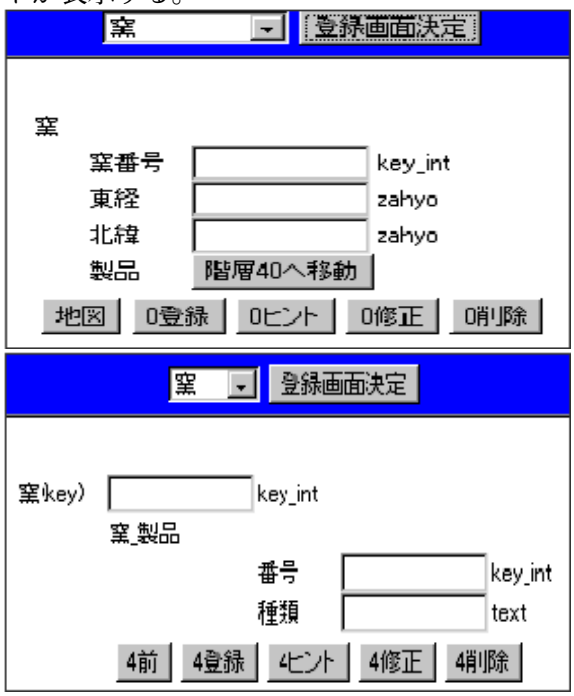

図 4 階層ごとの質問シート

(2) 全項目の質問シート

この質問シートは、ユーザが定義した項目全てが、階層に従ってインデントされて表示される。XML のタグスケルトンそのものを見出しラベルとテキストボックスにして表示したものである。このシートでは、データ検索条件が指定できるようにテキストボックスに書き込める文字が工夫されている。各テキストボックスに入力できるものは比較演算子と文字列の組み合わせである。

$(\theta \quad value)$

θ ::= = | < | <= | > | >= | *like*

value ::= 文字列 | #項目名

なお比較演算子の = は省略可能である。このような比較演算子と文字列を記入することで、各データ項目の検索条件を記述することができる。また、特殊文字として “#” を使うことで他の項目とも比較演算が行なえる。#に続け

て項目名を書くと、その項目と書き込んだテキストボックスの項目の比較を行なうことができる。1 つのシート内での条件指定は全て AND でつながれたものとして解釈する。OR 等の AND 以外の論理演算の指定は次節で説明する。図 5 にこの質問シートの表示例を示す。

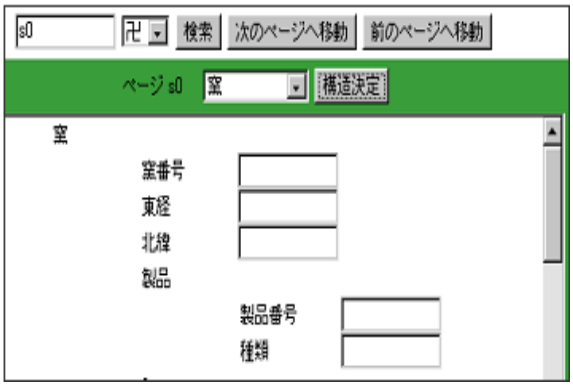

図 5 全項目の質問シート

3　質問シートによる検索

ここでは、これまでに説明してきたシステム上で、質問シートからどのように検索を行なうかを検討する。階層ごとの質問シートは各シートに対してテーブルが対応しているため、入力や削除更新は比較的簡単に SQL が生成できるので、説明は省略する。全項目質問シートつまり検索のための質問シートについては、前節で説明したように条件式と文字列が各データ項目に対して指定できるので、単純な質問はシート 1 枚で実行できる。集約関数等も必要な場合が出てくることも考えられるが、本システムの検索結果は地図上の点として表示することを優先的に考えているため、現時点ではサポートする予定はない。

検索結果

ここで、検索結果についての説明をしておく。このシステムでは、地図上へのマーカー表示を主眼にしているので、検索結果は最小限位置情報つまり (東経, 北緯) だけが得られれば良い。この位置情報を元に、システムは指定地図上に結果地点をマークする。しかし、こうすると複数のデータ項目を組み合わせた検索で区別ができないことや、結果の地図から各地点の入力データを調べられないなどの不便なことが考えられるので、ルートノード直下のキー属性も検索結果に含める。従って (キー属性, 東経, 北緯) への射影が結果として得られる。先の例で製品の種類に「すり鉢」が指定された場合を例に取ると、質問シートからは次の XML 文書が生成される。

```
<窯>
  <窯番号></窯番号>
  <東経></東経>
  <北緯></北緯>
  <製品>
    <番号></番号>
    <種類>すり鉢</種類>
  </製品>
</窯>
```

この XML 文書から値の入ったタグだけを取り出し、条件式を生成する。検索結果は、先に示した 3 フィールドに射影されるので、SQL は次のように生成される。テーブル名とフィールド名は 2.1 節で説明した命名規則による。他の項目にも値が入力されれば、AND で条件が付け加えられていく。また、階層の上下のテーブル分割に対する結合属性はデータ項目定義時に自動的に保存されたものを利用する。結合属性は次の SQL の最後の行に出てきている。

```
SELECT DISTINCT 窯_窯番号, 窯_東経, 窯_北緯
FROM   窯, 窯_製品
WHERE  窯_製品_種類 = "すり鉢" AND
       窯_窯番号 = 製品_窯_窯番号
```

結果は配列に入れられ、地図の画像サイズと周辺の緯度経度から結果のマーカーを描く位置を計算し表示する。この計算結果を使い、地図上でマーカーをクリックすると、その地点の情報をポップアップ表示する。実行例を図 6 に示す。

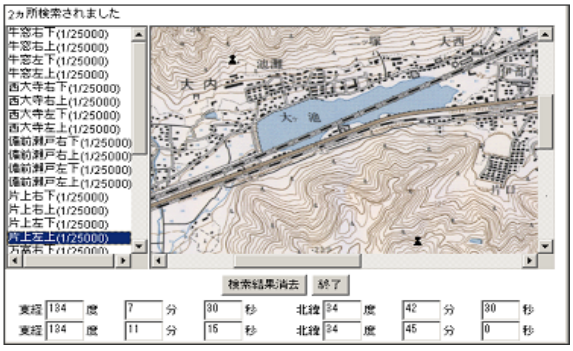

図 6 検索結果表示画面

ここで、元々のデータモデルがXMLで記述された階層構造を持つことを考えると、この検索はXMLの文書検索と考えることができる。XQuery風に書けば、次のような質問になる。

```
for $x in  root/窯
where $x/製品/種類="すり鉢"
return $x
```

本システムでは、このような構文には対応しておらず、上記のSQLを生成する。検索の効率としてツリーのスキャンをしなくて良く、条件が書かれたサブテーブルのみの検索が行なえるので高速な検索が期待できる。

本システムでは、更に複雑な条件での検索ができるように、複数のシートに条件を書いてシートを組み合わせることができる。図5を例に操作を説明すれば、まず、2番目のシートを開くには図5の上部に表示された「次ページへ移動」ボタンを押す、そこで条件を入力し、図5の左上のテキストボックスの記入領域には、シート名が開いた順に $S0, S1, \cdots$ とつけられてゆくので、論理演算子を書き込めばよい。

以降このような複数のシートを組み合わせた検索について考察する。シートの組み合わせには次の3種類が考えられるので、それぞれについて検討していく。ここで同一構造と呼んでいるものはデータ定義構造での同一性なので、実体は同一のテーブル群を扱うという意味である。構造が同じで異なるテーブルではないことをおことわりしておく。

1. 同一構造複数検索
2. 異種構造複数検索
3. メッシュ組み合わせ検索

関連研究として既存のDRBテーブル群にXMLビューを定義して、その上でXQueryを実行するシステムがある[9]。実行結果にはTaggerというタグ付け機構を用意して、XML文書で結果が得られるので、RDBをXMLでうまくラップできている。

3.1 同一構造複数検索

一枚の質問シートでは複数項目に対して入力した条件がANDでつながれて実行される。より複雑な条件を記述するためにはシートを複数利用する。

演算子はAND, OR, NOTが利用できる。これらの演算は検索対象テーブルが同じものの結果を論理演算するものであるので、集合演算としては、INTERSECT, UNION, EXCEPTと同等である。SQLでは2つのSELECT文をこれらの集合演算子で連結した演算を行なえるが、本システムでは、検索の効率から、これらの条件をwhereの条件にまとめてしまい、演算子で個々のシートからの条件を連結したSQLを生成する。これは、同一テーブル群に対しての検索条件なのでwhereの条件にまとめられるということである。

これを、XMLの文書検索という点からみると、図7のように見ることができる。ここで条件Aの部分木と条件Bの部分木を集合演算した結果の木ができる。これらは条件Aと条件Bを論理演算した条件の部分木ということができる。

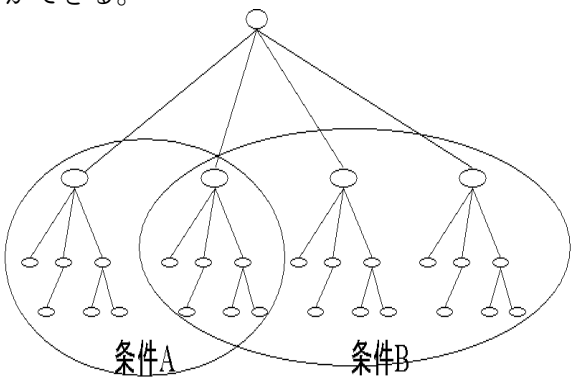

図7 部分木の集合演算

3.2 異種構造複数検索

ここでいう異種構造とは、調査者が異なるとか調査した年代が異なるなどで、調査対象は同じでも調査項目が異なる場合も含む。もちろん、遺跡と植生などという分野が異なる場合もある。これらの論理演算又は集合演算にどのような意味があるかを考察する。まず、前節の同一構造複数検索の場合と異なるのは、

別々の条件で選択した結果の集合演算を、一つの検索式の選択条件の論理演算に置き換えられないことがあげられる。これは、木の構造が異なるので、それぞれの木に対して検索条件を書かなければならないからである。では、個々の選択条件で得られた結果を集合演算すると、結果はどうなるであろうか。UNION の場合、それぞれの結果から得られる林に共通の親があるとして、構造的に見れば 2 種類の構造の部分木を持つ木が生成される。それでは、INTERSECT や EXCEPT は意味をもつ演算になるだろうか。この演算は普通に行なったら木の構造が異なるので意味有る演算にはならない。しかし、我々のシステムは検索結果を地図上に表わすことを目的としている。従って、検索条件に適合する木から、東経と北緯のデータだけを取り出して、共通集合や差集合などを求めることには意味が出てくる。例えば、遺跡と窯の両者が複合している場合などである。この演算には多少の注意が必要である。それは、位置情報の精度を度数で 1 秒にすると日本付近の地図では経度緯度とも 1/25000 の地図では約 1mm 強に相当する。従って、許容範囲をある程度 (2,3 秒) 設定して判定することが必要である。

ここで、UNION の場合の木の検索結果の概念を図 7 に示す。ここで示された木は意味あるものとは考えにくいかもしれない。しかし、この中の位置情報とその位置の示すキー属性だけに注目し、それが地図上に表示された場合を考えると、調査データの分布の関係がわかりやすくなると考えられる。

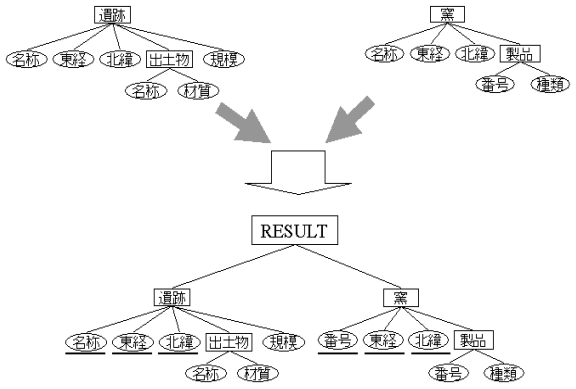

図 8 異種構造の部分木の集合演算の概念

実際の検索結果は、次の属性を持つようにしている。

構造名	key 値	東経	北緯
遺跡	作山古墳	134,24,30	34,35,15
窯	123	135,10,43	34,20,51
⋮	⋮	⋮	⋮

このような結果のテーブルを得ることで、地図上のマーカーをテーブルことに変えることや、地図上のマーカーのクリックに対応してポップアップで情報を表示できるようになる。

3.3 メッシュ組み合わせ検索

メッシュデータはある図面領域を経度方向に m 等分、緯度方向に n 等分したもので、これらの分割された地域の代表値が対応付けられたものである。代表値には標高や降水量、気温など大域的な観測で得られるものが多い。これらは元の図面の周辺の経度緯度と分割数 m と n が分かっているので、ある場所がどのメッシュに入るかは簡単に計算で求められる。

例えば、1/25000 地形図の縦横 10 等分した領域は標準メッシュと呼ばれ、国勢調査などの調査領域に利用されている。この領域には緯度経度から計算できるメッシュ番号が振られているので、ある地点がどのメッシュに含まれるかは東経と北緯から簡単に計算できる。

調査データとメッシュデータとを組み合わせた検索をする場合は、メッシュデータの性質から AND 演算のみ考えればよい。AND を実現する方法として、(1) 前の 2 つの検索のようにそれぞれのデータに対して別々に検索をして、条件に適合した調査データとメッシュ領域とを逐一付き合わせて判定するものと、(2) 調査データの検索結果のそれぞれに対してその位置を含むメッシュの値を判定させるものとが考えられる。

調査データとメッシュデータの検索結果の数の割合によって (1) と (2) の優位性が変わってくる。調査データ、メッシュデータに対してそれぞれ次のようにおいて考える。

$$\text{データ総数} \quad : \quad F_t, \quad M_t$$
$$\text{選択レコード数} \quad : \quad F_s, \quad M_s$$

(1) の場合、検索条件でそれぞれを検索するのに $F_t + M_t$ の比較演算がかかり、これに調査データの選択結果の数 F_s だけ、結果メッシュと比較すればよいので、合計 $F_t + M_t + F_s$ の検索コストと考えられる。

(2) の場合、調査データの条件検索に F_t かかり、結果の数 F_s それぞれにメッシュのデータを検索するのに $F_s \times M_t$ となる。合計は $F_t + F_s \times M_t$ となり (1) よりもコストが大きくなる。しかし、調査データの東経北緯からメッシュ番号が計算で求められるので、メッシュ番号にインデックスが貼ってあれば、この式の M_t を 1 と見ることができるので、合計で $F_t + F_s$ となるので (1) よりも低いコストで結果が得られる。

従って、本システムでは (2) の手順で検索を行なうことにする。

3.4 検索結果の XML 出力

本システムでは、野外調査の位置を地図上に表示することを主な目的にして設計しているが、有用な多くの野外調査データが蓄積されることを考えると、条件検索の結果をレコード型や XML 型で取り出すことも必要になることが考えられる。データの保存先が RDB なので、レコード型で返すのは問題ない。また、XML 型では、構造定義情報のタグスケルトンに結果の値を入れて返すだけでよいので、簡単に実現できる。

この場合の検索言語であるが、質問シートに結果取り出しフラグをつけたものを提案する。これだと、質問シートの条件の記述は以上見てきたとおりのものがそのまま利用でき、結果を得たい項目にチェックするだけで検索ができる。検索シートを SQL に変換するには、フラグのついた項目に射影する演算を実行すればよいので、今のシステムの仕組みを利用できる。例えば、次の SQL が生成される。

```
SELECT DISTINCT フラグ項目列
FROM テーブル列
WHERE 質問シートの条件群 AND
      テーブル結合条件
```

4 終わりに

野外調査データを地図上に表示するだけで、地域的な特性や性質が見えてくることがある。そのような発見を支援するシステムの設計とプロトタイプの開発をした。データモデルを XML の DTD とタグスケルトンで行なうことで、データ入力や検索のインタフェース：質問シートを構造的に自動生成することができた。この報告では、質問シートによる検索、特に複数の質問シートの組合せ検索を中心に考察した。検索結果を地図上に表示するというシステムの性質から、検索するデータ構造の組み合わせにより、内部で生成する SQL の検索式が変わること、またメッシュデータと組み合わせた検索ではメッシュデータの性質に着目して、より効率よい検索手順を考察した。

今後の予定としては、メッシュ検索と XML 出力の実装を行い、野外調査データの新しい発見に役立てたいと考えている。

参考文献

[1] 久保幸夫, 厳網林, 「地理情報科学の新展開」, 日科技連 (1996)

[2] 高木悟, 松本一則, “地図情報を利用した情報検索”, 情報処理 Vol.41, No.4, 357-362 (2000)

[3] 松本圭司, 他, “地図データベースへの応用を考えた多次元データアクセス:ビット埋め込み R 木”, 情処研報 Vol.2000, No.10, 169-176 (2000)

[4] G-XML, http://gisclh01.dpc.or.jp/gxml/contents/

[5] 国土地理院, http://www.gsi.go. jp/

[6] 吉川正俊, 志村壮是, 植村俊亮, “オブジェクト関係データベースを用いた XML 文書の格納と検索”, 情報処理学会論文誌, Vol.40, No. SIG6(TOD3), 115-131 (1999)

[7] 北川文夫, “野外調査のためのワークベンチ”, 情処研報 DB122-24, pp.185-190 (2000)

[8] 藤田幹則, 北川文夫, “XML によるスキーマ定義を用いた情報システム”, 信学技報 DE2001-4, pp.25-31 (2001)

[9] J. Shanmugasundaram et al., “Querying XML Views of Relational Data”, 27th VLDB, 261-270 (2001)