

検索フォームにおける未入力変数を含む質問文の自動変換

小林 武彦 † 遠山 元道 ‡

† 慶應義塾大学大学院 理工学研究科 開放環境科学専攻

‡ 慶應義塾大学 理工学部 情報工学科

E-mail: † tk@db.ics.keio.ac.jp, ‡ toyama@ics.keio.ac.jp

本稿では、Webにおけるデータベース検索フォームをプログラミングを必要とせず、HTMLとSuperSQLの知識のみでデータベース検索フォームを容易に作成する方法を提案する。提案に基づき本稿では、SuperSQLクエリに対して新たにクエリ内変数の概念を導入する。従来手法では条件式を一つずつ4つのパートに分けてフォームから送信していたが、提案手法ではクエリにクエリ内変数を利用した条件式・条件句を記述する。その後フォームよりクエリを引き渡しクエリを解析する。解析後のクエリのクエリ内変数値をフォームより受け渡しクエリを完成させ、SuperSQLによってクエリを処理し、HTMLを出力する。しかし変数が未入力の場合も多々考えられるため、その際の質問文の変換アルゴリズムについても述べる。従来では検索フォームを作成するためにはプログラミング言語を用いる必要があり非常に多くの知識やスキルが必要で作成に多くの時間がかかってしまう場合や、またプログラミングは必要としないものの自由度が低い場合などがあったが、本稿の提案によって検索フォームを作成するうえで必要なスキルや知識、時間を削減することを目標とする。また本論文はDEWS2006で発表したSuperSQLのフォーム対応の発展版である。

キーワード：検索フォーム， SuperSQL， クエリ書き換え

Automatic conversion of not inputted variable included query in search form

Takehiko KOBAYASHI † Motomichi TOYAMA ‡

† School of Science for OPEN and Environmental Systems,
Faculty of Science and Technology, Keio University.

‡ Department of Information and Computer Science, Faculty of Science and Technology,
Keio University.

E-mail : † tk@db.ics.keio.ac.jp ‡ toyama@ics.keio.ac.jp

This paper we proposes the way to construct database search form by using only SuperSQL and HTML skills not using any programing skills. We implement query variable to SuperSQL query. The established mehod was that form send a each conditions which must separated into four parts. Therefore in new method, conditions are written in a query by using query variables. After getting a query from the form and construing the given query, replace query variable by value from the form to complete a query and output a HTML by using SuperSQL. However variable may be empty, so that query variable can not be replaced. So we propose traslation of query for empty variables. This paper is upgrading version of the paper which proposes "Form Interface of SuperSQL" in DEWS2006.

keyword : search form , SuperSQL , rewriting query

1 はじめに

近年 Web においてデータベースの内容を検索するフォームが数多く見受けられるようになった。航空機や新幹線などの検索やホテルなどの予約状況など、それらは多岐にわたる。このような検索フォームによってユーザはその都度窓口に出向いたり電話で問い合わせる必要がなくなり、移動の手間や時間などを削減できユーザの利便性が大きく向上している。また近年のインターネットの高速化により懸案事項ともいえたインターネットのアクセス速度も大きく改善され、ユーザはストレスなく検索を行うことが可能になっている。

しかしながらこのようなフォームを作成するためには HTML の知識のみならず、高度なプログラミングスキルが必要であり、作成できる人間は限られている。また高度ゆえに作成するためにかかる時間も長くなっている。

そこで以前、SuperSQL を利用したフォーム作成法を提案した [3]。SuperSQL [1, 2, 4] によってクエリから直接 HTML が出力できるため、クエリの作成を手助けするようなサブレットを作成すれば検索フォームが作成可能だと考えた。しかしながら従来のシステムでは多くの問題点があり、システムとして不十分であった。様々な制約があり、その自由度は低く、またエラー耐性も低かった。そこで本稿では前回提案したシステムの問題点を解決するために、システムの提案自体をやり直す。そこで新たにクエリ内変数の概念を導入する。またフォームから変数の値を受け取れなかった場合（値が未入力の場合）の解釈についてのアルゴリズムも提案する。

次に本稿の構成を述べる。セクション 2 で従来技術について述べる。セクション 3 で今回利用する SuperSQL について述べる。セクション 4 では今回の提案について述べる。

セクション 5 で実際のフォームの作成方法を述べ、セクション 6 で結論と今後の課題を述べる。

2 従来技術と問題点

Web においてデータベースを検索するフォームを作成する方法は主に PHP [5] や JAVA [6] などといったプログラミングを利用する必要がある。またその他の方法として SuperSQL を利用した方法 [3] などもある。

2.1 プログラミング

Web において検索フォームの内部ロジックを実装するためには PHP [5] や JAVA [6] などといったプログラミングスキルが必要となる。これらの言語は非常に高級で様々な複雑なロジックを実装することが可能であり、利用率も非常に高い。

しかしながらこれらの言語を用いて検索フォームを作成するためには複雑なロジックを記述できる反面、高度な知識とテクニックを必要とする。それらを会得するためには非常に多くの労力と時間を必要としてしまうため、作成できる人間は限られてしまっている。また作成にかかる時間も膨大になることが多い。

2.2 SuperSQL のフォーム対応

従来提案した SuperSQL のフォーム対応 [3] では多くの問題点が存在する。従来の方法としてはフォームの `<input>` タグの `hidden` タイプを利用し、`where` 句以前のクエリと設定ファイルを引き渡し、条件部を左辺、右辺、条件、値の型・AND/OR オペレーターの 4 つに分けて引き渡していた。そして通常、条件の右辺にフォームに入力された値が入る。フォームに値が入力されなかった場合にはその条件を利用することはなく、その条件はなかったことになる。この方法でまず問題になる場合が変数が二つ以上ある場合である。具体的には BETWEEN 句や IN 句などであるが、従来手法ではこれらを利用することはできなかった。そのため全てを不等号を基本とする簡単な式で表現する必要があった。二つ目の問題点は条件部を HTML に記述するため、スキーマ情報が不特定多数の人間に閲覧される可能性があるという点である。三つ目の問題点はラジオボタンやチェックボックスなどを利用するためには様々なテクニックを駆使する必要性があり、利便性が高いとは言いがたい。さらに四つ目の問題点はプログラミングと比べると記述量は劇的に減ったものの、それでも一つの条件のために引数が 4 つもあり、記述量が多いという点である。またさらに不等式以外の条件を利用するためにはある程度のテクニックが必要となってしまう点も問題点として挙げられる。

2.3 研究目的

2.2 で述べた問題点を解決しつつ、プログラミングを利用する場合に比べて Web における検索フォームの作成に必要な労力、時間、スキルを大幅に減らす方法として SuperSQL に対してクエリ内変数の

導入と未入力変数の解釈アルゴリズムを提案する。

3 SuperSQL

この章では本論文で改善を試みる SuperSQL について簡単に述べる。SuperSQL は関係データベースの出力結果を構造化し、多様なレイアウト表現を可能とする SQL の拡張言語であり、慶應義塾大学遠山研究室で開発されている [4][1]。そのクエリは SQL の SELECT 句を GENERATE< media >< TFE > の構文を持つ GENERATE 句で置き換えたものである。ここで < media > は出力媒体を示し、HTML、PDF などの指定ができる。また < TFE > はターゲットリストの拡張である Target Form Expression を表し、結合子、反復子などのレイアウト指定演算子を持つ一種の式である。

3.1 結合子

結合子はデータベースから得られたデータをどの方向 (次元) に結合するかを指定する演算子であり、以下の 3 種類がある。括弧内はクエリ中の演算子を示している。

- 水平結合子 (,)

データを横に結合して出力。

例: Name, Tel

name	tel
------	-----

- 垂直結合子 (!)

データを縦に結合して出力。

例: Name! Tel

name
tel

- 深度結合子 (%)

データを 3 次元方法へ結合。出力が HTML ならばリンクとなる。

例: Name % Tel

name

 →

tel

3.2 反復子

反復子は指定する方向に、データベースの値があるだけ繰り返して表示する。また反復子はただ構造を指定するだけでなく、そのネストの関係によって属性間の関連を指定できる。例えば

[科目名 ! , [学籍番号] ! , [評点] !

とした場合には各属性間に関連はなく、単に各々の一覧が表示されるだけである。一方、ネストを利用して

[科目名 ! [学籍番号 , 評点] !]

とした場合には、その科目毎に学籍番号と評点の一覧が表示されるといったように、属性間の関連が指定される。以下、その種類について述べる。

- 水平反復子 ([] ,)

データインスタンスがある限り、その属性のデータを横に繰り返し表示する。

例: [Name],

name1	name2	...	name10
-------	-------	-----	--------

- 垂直反復子 ([] !)

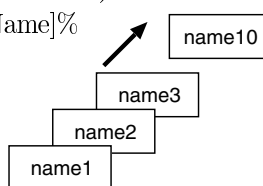
データインスタンスがある限り、その属性のデータを縦に繰り返し表示する。

例: [Name]!

name1
name2
...
name10

- 深度反復子 ([] %)

データインスタンスがある限り、その属性のデータを奥行き方向 (HTML ではリンク、PDF ではページ変換) に繰り返し表示する。

例: [Name]% 

3.3 装飾子

SuperSQL では関係データベースより抽出された情報に、文字サイズ、文字スタイル、横幅、文字色、背景、高さ、位置などの情報を付加できる。これらは装飾演算子 (@) によって指定する。

<属性名>@[<装飾指定>]

装飾指定は”装飾子の名称 = その内容”として指定する。複数指定するときは各々を”,”で区切る。

3.4 関数

SuperSQL ではいくつかの関数が用意されている。ここでは代表的な関数を 2 つ紹介する。

3.4.1 imagefile 関数

imagefile 関数を用いると画像を表示することが可能となる。引数には属性名、画像ファイルの存在するディレクトリにパスを指定する。

imagefile(id, path=”./pic”)

3.4.2 sinvoke 関数 (出力メディアが HTML の場合のみ)

sinvoke 関数は FOREACH 句と同時に用いる。これらを用いることで深度結合子と同様にリンクを生成することができる。

3.4.3 embed 関数

embed 関数を用いることでクエリを分割・合成することが可能になる。利用方法は別ファイルに保存されたクエリ、もしくはHTML ファイルを埋め込みたい箇所に embed 関数を記述する。

```
embed(file="./test.sql" where="ca.id=" att=ca.id)
```

4 提案手法

従来手法の問題点の総括として挙げられるのが1つの条件を利用するために4つの引数を受け渡さなければならないということである。この実装方法のため、フォームのソースが長くなってしまったり、利用できる条件式の自由度が低くなってしまっていた。またスキーマ情報が不特定多数の人間に見られてしまう可能性があった。そこで引き渡すクエリ自体に条件式を書いてしまう方法を考えた。そしてそのクエリを置いておくディレクトリをセキュアな状態にしておけば、スキーマ情報が漏れることは防げられると思われるからである。しかし当然フォームから受け渡した値を条件式に利用しなければ意味がない。そこでクエリ内変数という概念を導入し、その変数名でフォームからの値を待ち受け、取得し変数を実際の値に置き換えてクエリを完成させ、処理するような実装を SuperSQL を利用して行った。その際、値を取得できなかった場合の処理方法も合わせて提案する。

4.1 クエリ内変数

フォームから条件の値を与える場合、入力値を代入するための変数がクエリ内に必要になる。この変数をクエリ内変数と呼ぶ。クエリ内変数を含むクエリを「不完全な」クエリと呼ぶこととする。なぜなら不完全なクエリはそのままの状態では利用することが不可能であるためである。

クエリ内変数は\$マークで始まる任意の文字列である。クエリ内変数は現状では where 句のみで利用することができる。クエリ内変数はフォーム側のHTML の < input > タグの name にも利用する。それはクエリ処理側でフォームからの値を受け取る際にフォーム側とクエリ側の対応を取るためである。そのためフォーム作成時にはクエリとフォームとの対応関係を十分確認する必要がある。またクエリ内変数には記号 (" \$ ' ! % & < > [] { } , # ? *) を利用することはできない。

例：\$name, \$sex, \$age

4.2 クエリ解析

フォームからクエリのアドレスを受け取り、クエリを読み込む。このとき読み込んだクエリは不完全なクエリである。その例を以下に示す。

不完全なクエリ例

```
GENERATE HTML
...
FROM member m
WHERE m.name like '%$name%'
AND (m.name like '%take%'
AND m.age >= 20)
AND m.age BETWEEN $lower AND $upper
```

上記クエリは不完全なクエリである。なぜなら上記クエリにはクエリ内変数が存在するためである。そこでクエリ内変数を置き換えるための作業が必要になる。ここで入力値を以下のように仮定する。

表 1: 入力仮定値

クエリ内変数	入力値
\$name	なし
\$lower	22
\$upper	なし

表1のような入力があったとすると、先ほどの不完全なクエリの完成形は以下のようなクエリとなる。

完全なクエリ例

```
GENERATE HTML
...
FROM member m
WHERE (m.name like '%take%'
AND m.age >= 20)
AND m.age >= 22
```

クエリ内変数は where 句内でのみ利用可能としているため、実際に解析する箇所は where 句のみである。

クエリを容易に解析するためにまず、クエリを不等式および like 句のみにするため、BETWEEN 句

や IN 句などといった述語を書き換える必要がある。
その方法を次のセクションで述べる。

4.2.1 条件句書き換え : Query Converter

Query Converter は条件句を引数に取り、BETWEEN 句や IN 句を不等式によって置き換える。この処理を行うことで AND/OR は式の区切り文字として扱うことが可能となる。また 1 つの式に値や変数は 1 つとなり、条件句が「単純な」条件句にすることができる。またこの処理によって条件句が「式」と AND/OR オペレーターのみで定義することが可能になる。「式」の定義は AND/OR オペレーターによって区切られる不等式、IS NOT NULL 句、IS NULL 句、like 句のいずれかである。

BETWEEN 句は (属性名 >= 下限値 AND 属性名 <= 上限値) に書き換えられ、IN 句は (属性名 = 値1 OR 属性名 = 値2 OR … 属性名 = 値 N) に書き換えられる。

4.2 で挙げた例の変換後は以下ようになる。

変換後のクエリ例

```
GENERATE HTML
...
FROM member m
WHERE m.name like '%$name%'
AND (m.name like '%take%'
AND m.age >= 20)
AND m.age >= $lower
AND m.age <= $upper
```

ここで生成されたクエリは依然不完全なクエリである。

クエリ内変数を置き換えるためにはまず、式を取り出さなければならない。次のセクションで式を取り出し方法を述べる。

4.2.2 条件句解析 : Change Query

Change Query では条件句を引数に、条件句から式を抜き出す作業を行う。4.2.1 の作業を行うことで式の区切り文字は AND/OR と見なすことができる。しかしここで注意しなければならないのが、括弧の存在である。括弧の中は括弧の中だけで独立に真理値が決まらなければならない。そこで括弧の中身は式ではなく条件句と判断し、括弧の中身を引数に Change Query を呼び出すこととした。このような作業を行うことで、式の最下部からきちんと評価

を行うことが可能となっている。

このクラスでは AND/OR を区切り文字として認識するが、その際に右括弧の方が左括弧よりも多い場合には括弧内であると判断し、区切り文字と判別しない。その代わりに括弧内である、ということを示すフラグを立て、右括弧の数と左括弧の数が同じ場合で AND/OR を読み込んだ時に、括弧内フラグが立っていれば、それまで全てのパーツを条件句として、Change Query を再度利用する。もし立っていない場合は、式を引数に式の評価 4.2.3 を行う。

前述のクエリ例の場合、以下ようになる。

1. m.name like '%\$name%'
2. (m.name like '%take%' AND m.age >= 20
 1. m.name like '%name%'
 2. m.age >= 20
3. m.age >= \$lower
4. m.age <= \$upper

これらは配列などに全て記憶するのではなく、1 つの式の読み込みが終了後、すぐに式の評価を行う。

次に式の評価、すなわちクエリ内変数の置き換えを行う。

4.2.3 式の評価、クエリ内変数値の取得 : Get Parameter

Get Parameter は式を引数に取る。引数で与えられた式を評価するのである。

ここで行うのはクエリ内変数の置き換えである。クエリ内変数が存在しない場合には、得た式がそのまま戻り値となる。

クエリ内変数の有無は式を読み込みつつ、\$マークがあるか否かで判断する。クエリ内変数が存在した場合、その変数の値をフォームから取得する。取得が成功した場合には、取得した値をそのまま利用し置き換え、式を戻り値とする。取得が失敗した場合には、undefined と評価し、それを戻り値とする。

前述のクエリ例の場合、以下ようになる。

1. undefined
 1. m.name like '%name%'
 2. m.age >= 20
3. m.age >= 22
4. undefined

これらの得られた式を元のクエリに存在した AND/OR で繋げることでクエリが生成される。しかしその生成されたクエリも依然不完全なクエリといえる。

変換後のクエリ例

```
GENERATE HTML
```

```

...
FROM member m
WHERE undefined
AND (m.name like '%take%')
AND m.age >= 20)
AND m.age >= 22
AND undefined

```

なぜなら undefined という文字が含まれており、これは SQL ではサポートされていないためである。undefined を適切な値に置き換える必要がある。

その方法を次のセクションで述べる。

4.2.4 変数未入力時置き換え:undifined variable

undefined と評価された式を SQL として、正しい値に置き換える必要がある。そこで undefined を True/False に置き換えることとした。置き換え規則としての大前提は

「置き換えた結果は条件判定に全く影響しない」ということである。その結果、置き換え規則は以下のように決定した。

- 前後に AND があった場合、True
- 前後に AND が一つもない場合、False

この理由は AND の方が式の評価が早いことである。そのため AND が前後にあった場合、その前後は関係なく AND によって繋がれた 2 つの式が評価される。ではなぜ False ではなく True なのか。それは AND の真理値表(表 2)を見れば一目瞭然である。表の縦に左辺、横に右辺である。

表 2: AND 真理値表

	True	False
True	True	False
False	False	False

このようにもし片方に False を補ってしまうと、もう片方の真理値に関係なく、False となってしまう。これでは前に挙げた大前提が崩れてしまう。逆に True を補えば、もう片方の真理値が有効となる。そのため、ここでは True を補うこととした。

OR の場合の真理値表は表 3 のようになる。

このように AND の場合のように True を補ってしまうと、片方の真理値に関係なく、True となってしまうため、ふさわしくなく、False を補うのが

表 3: OR 真理値表

	True	False
True	True	True
False	True	False

妥当であると判断した。
 例:…AND ??? AND… -> …AND True AND…
 … OR ??? AND… -> … OR True AND…
 … AND ??? OR… -> … AND True OR…
 … OR ??? OR… -> … OR False OR…

しかし、ユーザによってはあえて OR に対して True を、もしくは AND に対して False を補いたい場合もあると考えられる。その場合にはクエリ変数の \$マークの前に補いたい真理値を書いてもらうこととした。

例: T\$age, F\$sex

これらの処理の結果、完全なクエリが完成する。

```

変換後のクエリ例

GENERATE HTML
...
FROM member m
WHERE True
AND (m.name like '%take%')
AND m.age >= 20)
AND m.age >= 22
AND True

```

5 検索フォーム作成方法

実際に登録されているメンバーを検索するフォームを作成する。検索画面を図 1 に、検索結果を図 2 に示す。またフォーム側 HTML ソースと利用するクエリも示す。

5.1 フォームソース

```

1   フッター部
2   <HTML> <head>
3   <link rel="stylesheet"
4   type="text/css" href="form.css">
5   </head> <body> <center>

```

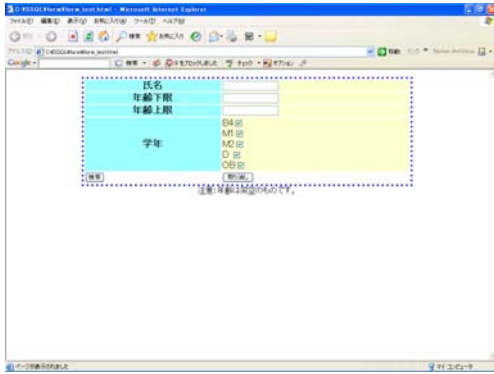


図 1: 入力画面

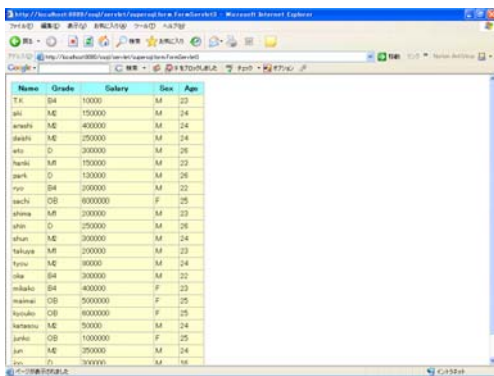


図 2: 出力結果

フッター部では HTML の基本的なタグとフォームの開始タグ (サブレットのアドレス記述) (9~12 行目)、および SuperSQL に必要な設定ファイル (13~16 行目) とクエリファイルのアドレス (17~20 行目) をサブレットに引き渡すためのタグを記述する。この際、スキーマ情報が外部に漏れてしまわないように、クエリファイル、設定ファイルの置かれているディレクトリはセキュアであることが望ましい。また設定ファイルに書かれている情報は、ホスト、データベース名、ユーザ名である。

```

1 条件入力部
2  <tr><td class="att2">氏名</td>
3  <td class="input">
4  <input type="text" name="$name">
5  </td></tr><tr>
6  <td class="att2">年齢下限</td>
7  <td class="input">
8  <input type="text" name="$lower">
9  </td></tr><tr>
10 <td class="att2">年齢上限</td>
11 <td class="input">
12 <input type="text" name="$upper">
13 </td></tr><tr>
14 <td class="att2">学年</td>
15 <td class="input">B4
16 <input type="checkbox"
17 name="$grade1" value="B4"
18 checked> <BR>
19 M1<input type="checkbox"
20 name="$grade2" value="M1"
21 checked><BR>
22 M2<input type="checkbox"
23 name="$grade3" value="M2"
24 checked><BR>
25 D <input type="checkbox"
26 name="$grade4" value="D"
27 checked><BR>
28 OB<input type="checkbox"
29 name="$grade5" value="OB"
30 checked>
31 <BR></td></tr><tr><td>

```

```

6  <table border = 0 width=70%
7  class="table">
8
9  <form method="post"
10 action="http://localhost:8080/
11 ssql/servlet/supersql.form.
12 FormServlet3">
13 <input type="hidden"
14 name="configfile"
15 value="http://www.db.ics.keio.
16 ac.jp/~tk/config2.ssql">
17 <input type="hidden"
18 name="sqlfile"
19 value="http://www.db.ics.keio.
20 ac.jp/~tk/form_test.sql">

```

条件入力部では実際の入力インターフェースを記述する。「氏名」を例にとると氏名はtextを入力させるため、<input> タグの type は text でその変数の名前はクエリに記述してある、氏名を代入すべきクエリ変数名と同じでなければならない。実際のクエリを見てみると、17行目に、\$name の記述があることが分かる。すなわち、氏名フィールドに入力された値は、この位置に代入されることになる。

また18行目ではcheckboxに対してcheckedのオプションが付けられていることが分かる。もしチェックが入っていないと「どの学年にも所属しない」という条件が送信されることになる。しかしこのような条件は通常ありえないと思われるため、予め全ての学年にチェックを入れておくほうがよりユーザの意図を反映しやすいと考えられるためである。このような条件を変数名を変えることで実装することも可能である。(4.2.4)

```

1   フッター部
2
3   <input type="submit"
4   name="submit" value="検索">
5   </td><td><input type="reset"
6   name="reset" value="取り消し"></td>
   </tr></table></form></body></HTML>

```

フッター部に関しては、HTMLの基本的なエンドタグと、検索・リセットボタンなどを記述する。

```

1   クエリ例
2
3   GENERATE HTML
4   { {
5   { "Name"@{class=name},
6     "Grade"@{class=grade},
7     "Salary"@{class=salary},
8     "Sex"@{class=sex},
9     "Age"@{class=age}    }!
10  [ m.name@{class=showname},
11    m.grade@{class=showgrade},
12    m.salary@{class=showsalary},
13    m.sex@{class=showsex},
14    m.age@{class=showage} ]!
15  } }@{cssfile=http://www.db.ics.
16    keio.ac.jp/~tk/form.css}
   FROM member m

```

```

17 WHERE m.name like '%$name%'
18 AND m.grade = '$grade'
19 AND m.age BETWEEN
20     $lower AND $upper
21 AND m.grade IN
22     ('$grade1', '$grade2', '$grade3',
23     '$grade4', '$grade5')

```

6 まとめと今後の課題

本稿では、従来提案した SuperSQL を利用することによって検索フォームを容易に作成する方法よりもさらに容易にかつ自由度の高いシステムを提案し、実装した。その中でクエリ内変数の概念を導入し、未入力変数の解釈アルゴリズムを提案、実装した。今後の課題はユーザアンケートなどを通して有用性や従来手法と比べた時の容易性の検討や実行速度の向上などが考えられる。

参考文献

- [1] M. Toyama, "SuperSQL: An Extended SQL for Database Publishing and Presentation", *Proceedings of ACM SIGMOD '98 International Conference on Management of Data*, pp. 584-586, 1998
- [2] 遠山 元道: 『ターゲットリストの拡張によるデータベース出版と概視の実現』、信学技報、Vol.93, No.152, P79-88、電子情報通信学会、1993
- [3] 小林 武彦, 遠山 元道 SuperSQLにおけるクエリの分割・合成プリミティブおよびフォーム対応の導入 In *DEWS2006*, 2006.
- [4] SuperSQL: <http://ssql.db.ics.keio.ac.jp/>
- [5] PHP:Hypertext Preprocessor . <http://www.php.net> .
- [6] Sun Microsystems. Java Enterprise Edition J2EE . <http://java.sun.com/javace/>.
- [7] W3C. HyperText Markup Language (HTML) 4.0. <http://www.w3.org/TR/1998/REC-html40-19980424/>.