社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－DBS－143 (49)
2007／7／3

SuperSQLを用いたRIA作成支援の提案

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 〒223-8522 神奈川県横浜市港北区日吉3-14-1

†† 慶應義塾大学 理工学部 情報工学科 〒223-8522 神奈川県横浜市港北区日吉3-14-1

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

あらまし 本稿では，SuperSQLを用いることで容易にRich Internet Applicationを作成する方法を提案する．近年Rich Internet Application，通称RIAと呼ばれるアプリケーションが多く登場し始めている．有名な例はAjaxを利用して作成されたGoogle MapsやG-Mailなどであろう．RIAの技術を用いて作成されたアプリケーションは優れたユーザインターフェースを持ち，操作性や視認性，オフラインでも扱えるなど多くのメリットを持つ．しかしながらこれらのアプリケーションを作成するためには膨大な時間が必要となってしまう．またSuperSQLではクエリを発行することでHTMLを出力することができる．しかしながらSuperSQLもまた出力したHTML上での操作性やレスポンスタイムとデータの新鮮度のトレードオフなどの問題点を抱えている．問題点のいくつかはクラシックなHTMLであるため起こる問題点である．そこで本稿ではRIA技術の一つであるAjaxに着目し，データベース内のデータをHTML化できるSuperSQLをAjaxに対応できるように拡張することでRIA作成支援を行い，同時にSuperSQLの問題点の改善を図る．

キーワード RIA, SuperSQL, Webアプリケーション, Ajax

A Proposal of Making RIA by using SuperSQL

Takehiko KOBAYASHI† and Motomichi TOYAMA††

† School of Science for OPEN and Environmental Systems, Faculty of Science and Technology, Keio University. Hiyoshi3-14-1, Kouhoku-ku, Yokohama-shi, Kanagawa, 223-8522 Japan

†† Department of Information and Computer Science, Faculty of Science and Technology, Keio University Hiyoshi3-14-1, Kouhoku-ku, Yokohama-shi, Kanagawa, 223-8522 Japan

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

Abstract This paper proposes the way to make Rich Internet Application by using SuperSQL. Now a days, there are many web applications on the web, especially Rich Internet Application(RIA). Well-known examples are 'Google Maps' and 'G-mail' and more. Applications that made with RIA technologies have rich user interfaces, capability with offline use, and more benefit. However, to make RIA contents, Web developers have to take long time to develop RIAs. On the other hand, SuperSQL can make HTML file by SuperSQL query. Nevertheless, SuperSQL have problems with usability on HTML published by SuperSQL and trade off between freshness of data and response time. Some of problems are because of using traditional HTML. On this paper, we focus one of the RIA technologies, Ajax, and we extend SuperSQL to give Ajax capability. So that Web developers can make RIA contents much faster. At the same time, we also solve SuperSQL's problems.

Key words RIA, SuperSQL, WebApplication, Ajax

1. はじめに

近年, インターネット上に優れたユーザインターフェースを持つアプリケーション, 通称 Rich Internet Application (RIA) [3] が増え始めている. RIA は従来の HTML やサーバサイドプログラミングによるアプリケーションが抱えていた操作性や実行時間の問題など様々な問題を解決する方法のひとつとして注目されている. RIA を作成するための代表的な技術は FLASH と Ajax [2] である. 代表的な Ajax を利用した RIA は Google Maps [9] や G-Mail [10] であろう. しかしながら Ajax を利用したアプリケーションにも問題点は存在する. その代表が作成コストの問題である. 多くの Web アプリケーション同様, アプリケーションの構築に多くの知識や時間が必要となってしまうことが問題点である.

またここで SuperSQL [1] に目を向けてみると SuperSQL はデータベースの内容を HTML 化することができるが, 問題点も存在する. SuperSQL によって生成できる HTML は従来のクラシックな HTML であり, 操作性などに問題を抱えている. またデータの新鮮度やレスポンスタイムにもトレードオフが存在しており, 問題点のひとつとなっている.

本論文では RIA の問題点と SuperSQL の問題点, 両方を解決するために SuperSQL を用いて RIA 作成支援を提案する.

次に本稿の構成を述べる. 2. 章で従来技術, 特に Ajax について述べる. 3. 章で, 本稿で拡張する SuperSQL の概要を述べる. そして 4. 章で SuperSQL における問題点を述べる. また 5. 章で提案手法について述べる. 6. 章で SuperSQL の従来手法と提案手法の比較検討を行い, 7. 章でまとめと今後の予定を述べる.

2. 既存技術と問題点

Web アプリケーションにおける操作性やレスポンスタイムなどの問題点を解決するために登場した技術が FLASH や Ajax に代表される RIA と呼ばれる技術である. ここでは本稿で扱う Ajax に焦点を当てる.

2.1 Ajax とは

Ajax とは"Asynchronous JavaScript + XML"の略で JavaScript, DHTML, CSS, XML(HTML) を組み合わせたユーザーインターフェース構築技術で, 近年注目されている. JavaScript の XMLHttpRequest を利用して非同期通信を行い, 通信結果に応じてダイナミック HTML を利用してページの一部分を書き換える. レンダリングレイヤーとロジックレイヤーを明確に分離し, それらを非同期にすることで通信中もユーザインタラクションをすることができる. また JavaScript によってクライアント側での計算・操作を可能にし, サーバとの通信を極力少なくするようにしている. そのため従来のアプリケーションと異なり, オフライン状態でもサーバとの通信を行うインタラクション以外であれば利用することが可能である.

2.2 Ajax の問題点

一見非常に優れた技術に見える Ajax だが, 問題点がないわけではない. それは従来のアプリケーションでも同じことだが, アプリケーション作成に膨大なコストがかかってしまうことである. また Web アプリケーションを作成するためには多くの知識を必要とする. Ajax の場合, JavaScript, HTML, CSS, XML に加えてサーバ側のロジックを実装するために Java や PHP などのプログラミング言語の知識も必要である.

2.3 研究目的

そこで本研究では Java や PHP, JavaScript といったプログラミング言語の知識がない人でも RIA を容易に作成できる方法として SuperSQL の拡張を提案する. 本研究ではデータベースの内容を利用するアプリケーションを考え, HTML, CSS, および SuperSQL クエリの知識のみで Ajax を利用した RIA を作成できることを目的とする.

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 [1] [7]. 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

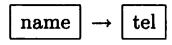
結合子はデータベースから得られたデータをどの方向 (次元) に結合するかを指定する演算子であり, 以下の 3 種類がある. 括弧内はクエリ中の演算子を示している.

- 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 

3.2 反 復 子

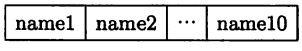
反復子は指定する方向に, データベースの値があるだけ繰り返して表示する. また反復子はただ構造を指定するだけでなく, そのネストの関係によって属性間の関連を指定できる. 例えば

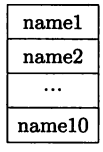
[科目名]! , [学籍番号]! , [評点]!

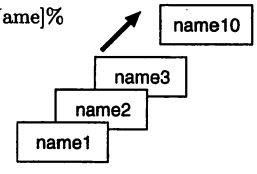
とした場合には各属性間に関連はなく, 単に各々の一覧が表示されるだけである. 一方, ネストを利用して

[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], 

- 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]! 

- 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 4 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.

sinvoke(cou.name, file="./menu.sql", att=co.country)

3.4.3 invoke 関数

invoke 関数はリンクを生成するための関数である. sinvoke 関数の場合, SuperSQL を手動で実行することでリンク先を生成しておくが, invoke 関数の場合, ユーザのリクエストに応じて動的にリンク先を生成する.

invoke(cou.name, file="./menu.sql",
condition="ca.country="+co.country)

3.4.4 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. SuperSQL における改善目標

SuperSQL における改善目標は 3 つある.

- 生成した HTML 上での操作性・ユーザビリティ
- レスポンスタイム
- データの新鮮度

一つ目は出力される HTML がクラシックな HTML である故に生じる問題点である. 多くのデータを得るためにはアンカーを多数クリックしなければならないが, その都度ページ全体がリフレッシュされてしまうため, ブラウザの戻るボタンなどの操作が必要で. 操作性が悪い. またページ遷移を理解するのが難しく, 自分がどういった経緯でデータを得たか, 以前のデータをどうすれば得られるかなど, ページ構成がわかりづらくなることがある.

二つ目のレスポンスタイムは三つ目のデータの新鮮度とのトレードオフである. 後述する二つの関数, invoke 関数と sinvoke 関数はそれぞれ静的, 動的にリンク先を生成する方法である. 静的生成の場合, 予めリンク先を SuperSQL を利用して生成しておくためレスポンスタイムは早い. しかしデータベース更新

に弱く, データの新鮮度が低いという問題点を持つ. 一方動的生成の場合, ユーザのリクエストに応じてSuperSQLによってページを生成するためデータベース更新に強く, データの新鮮度が高い. しかし生成に時間がかかってしまうため, レスポンスは相対的に悪い. 以下に二つの関数を説明する.

4.1 invoke関数

invoke関数ではリンク先を生成するためのSuperSQLクエリファイル, DB名やDBサーバなどが記述されているSuperSQLの設定ファイルなどをGET方式でサーバ側に送信する. サーバ側ではサーブレットによってSuperSQLシステムを起動し, 取得したクエリや設定ファイルを利用して必要なソースを生成し, ユーザに返答する. ソース生成直前のデータをキャッシュしておく研究[4]などによりレスポンスタイム向上を図っている.

4.2 sinvoke関数

sinvoke関数とは, リンク先を予め静的に生成しておく手法である. アンカータグの生成にsinvoke関数を使い, リンク先を生成するクエリにはFOR EACH句を利用する. ユーザはリンク元の生成とリンク先の生成のため, 最低2回SuperSQLを手動で実行しなければならない. 従来はリンク元とリンク先を生成クエリを1つのクエリとしていたが, 石川らの研究[5]によってデータベース更新時のコストの観点から複数のクエリに分割している.

5. 提案手法

2.2章や4.章の問題点を解決する方法としてSuperSQLによってAjax対応ページを作成可能にする. Ajax対応ページを作成可能にすることにより以下のようなメリットがある.

- RIAを作成する時間が短縮できる
- 動的生成を用いることで, データの新鮮度を高めることができる.
- 1ページ全てではなく一部を作成するため, 動的生成時の実行時間が短縮できる.

Ajax対応ページを作成するに当たり, 必要な作業は以下の5つである.

- Ajaxのための JavaScript 関数の作成
- JavaScriptを使うためのフロントエンドページの作成を可能にする
- Ajaxに対応するためにアンカータグ生成の拡張
- 装飾子の追加
- ページを作成できるサーブレットの作成

以下それぞれに関して詳細を述べる.

5.1 JavaScript関数の作成

サーバと通信し, その結果を所定の領域(div)に表示するための関数をprototype.js[8]を利用して作成した. prototype.jsはAjaxを作成するためのフレームワークの1つである. 作成した関数はリンク先生成・表示用の関数loadFileと検索結果生成・表示用の関数searchExeである. loadFileの引数は表示領域(div)のid, クエリ, 条件とし, searchExeの引数はformタグのidと表示divのidとした.

また「閉める」動作を可能にするため, 該当領域に既に表示がある場合, アンカーのクリックによって領域の内容を削除する. 図1はSuperSQLを利用して作成された検索フォームの検索結果[6]だが, アンカーをクリックすることによってその直下に選択した車のデータが出現する. その後車のデータを「閉めたい」場合, 再度アンカーをクリックする. しかしこの動作では, 1つの領域を複数のアンカータグによって書き換える場合に問題が生じる. 図3上の3の領域にあるアンカー(例:NSX)をクリックすると, どのアンカーをクリックした場合にも4の領域に車(NSX)のデータが表示される. しかし別の車(Civic)のデータを表示しようとしても4の領域に表示があるため, 4の領域を「閉じて」しまう. 実際に動作させたい環境は図1のようにアンカーと領域が1対1対応している場合である. そこで正しく動作するためにdiv-idに主キーの値を含む設定をした場合のみ, 「閉める」動作をすることとした. 主キーがdiv-idに含まれる以上, その領域に関しては1つのアンカー以上から更新されることはなく, 対象アンカーをクリックすることで「閉める」動作をしても問題はないと考えられる. 複数のアンカーによって共有されている領域を閉じる方法に関しては, 現在検討中である.

5.2 フロントエンドページの作成

フロントエンドページですべきことは二つある. ひとつはjsファイルの設定, もう1つがdivタグの設定である.

SuperSQLではGENERATE <media>とすることで出力メディアを指定する(例:HTML出力の場合, GENERATE HTML). Ajaxは HTMLであるため, 新メディアではなく作成時のオプションとした.

jsファイルの設定はAjaxオプションが設定されていた場合, HEADタグ内の情報を書く際に <script src="ajax.js"></script> と書き込むことで実現している.

次にdivタグの設定について述べる. 本研究ではembed関数を利用された箇所はdivタグを設定する. embed関数は1ページを複数のクエリの集合として作成できるようにする関数である. すなわちembed関数は複数の「コンテンツ」を生成するクエリをまとめて, ひとつの「ページ」に仕上げることになる. 通常リンク先は「コンテンツ」の内容を書き換えたもので, アンカーは「コンテンツ」を更新することになる.

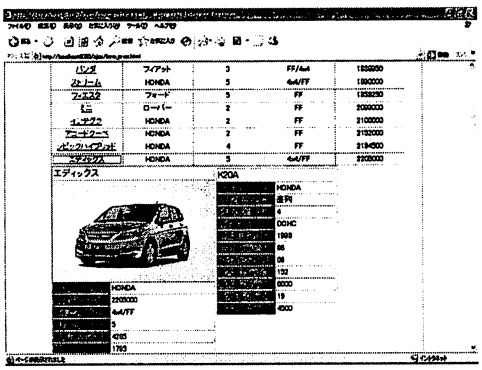


図 1　検索結果表示

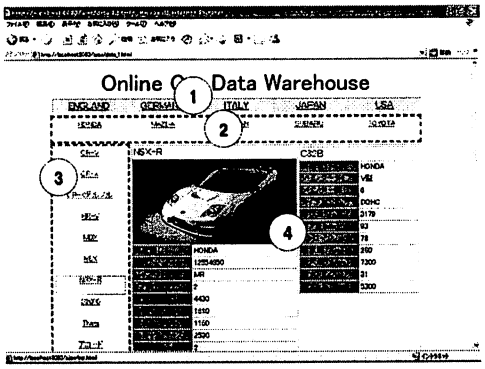


図 3　画面例 (FrontEnd.html)

```
GENERATE HTML
{
  {embed(file="menu1.sql")}!
  {"車種名", "製造者", ··· }!
  [sinvoke(ca.name,
    file="car.sql",
  att=ca.id)@{dispdiv=car+att} ,
    co.name, ··· ] !
   {embed(file="car.sql",
              where="ca.id=",
             att=ca.id)
  @{status=hidden,divid=car+att} }] !
}
FROM ···
```

図 2　検索結果表示クエリ (search.sql)

FrontEnd.html 生成クエリ (FrontEnd.sql)

```
GENERATE HTML
{  {
   { embed(file="menu1.sql") } !
   { embed(file="colist2.sql") } !
   {
     { embed(file="list1.sql") } ,
     { embed(file="car.sql")  }
   }
}@{tablealign=center, tableborder=0}
}@{cssfile=demo.css}
FROM car ca;
```

図 4　FrontEnd 作成クエリ

そのため"embed 関数=div タグ"という図式は理にかなっていると言える. その画面例を図 3, 生成クエリを図 4 に示す. menu1.sql はメニューバーを生成するクエリであり, フロントエンドクエリで menu1.sql が embed されている. menu1.sql の生成結果は図 3 中の 1 の箇所にあたる. 同様に colist2.sql は会社のリストを生成するクエリであり, colist2.sql が embed されている. colist2.sql の生成結果は同様に 2 の箇所にあたる. 同様に list1.sql は車のリストを, car.sql は車の詳細データを表示するクエリである. これらのサブクエリは「コンテンツ」を作成するクエリと言え, 図 4 で示したフロントエンドクエリは「ページ」を生成するクエリということになる.

5.3　アンカータグの拡張

通常のアンカータグでは href によってリンク先を指定するが, Ajax の場合リンク先を表示するために JavaScript を利用するため, onClick 属性に必要な関数呼び出しを書く必要がある. そのため 5.1 章で述べた loadFile 関数を呼び出すように, アンカータグの作り方を変更した. リンク先の表示領域はユーザが指定することができるが, ユーザの指定がない場合, 自動的にクエリ名と同じになる. (例：car.sql へのアンカータグの場合, 領域 ID は car)

図 3 の場合, list1.sql によって生成された車の詳細データ (car.sql) を要求するようなアンカーがある. このアンカーをクリックしたときに書き換えられるべき領域は, フロントエンドページで car.sql が embed された領域であることは明白である. それゆえ「クエリ名=領域名」としている.

5.4　装飾子の追加

embed 関数によって生成される領域の div タグの id は, システムが自動的に設定している (5.3 章). ただしユーザが明示的に div-id を指定したい場合には embed 関数の装飾子に新たに追加した divid とリンクを生成する関数 (現状では sinvoke 関数を転用) の装飾子に追加した dispdiv を対にして設定する. 例えば図 1 の場合, 車の詳細データページへのアンカーを生成する sinvoke 関数の装飾子に dispidv=car+att, 車の詳細データを表示する領域を作成する embed 関数の装飾子に divid=car+att を記述する. また att と

	Invoke	Sinvoke	提案手法
DB 更新コスト	○	×	○
データ新鮮度	◎	×	◎
レスポンスタイム	×	◎	◎
セキュリティ	×	◎	○

表 1　リンク先生成方法比較

は各関数に指定している属性のことである. (図 2 の sinvoke 関数の場合, ca.id)

さらに embed 関数に初期段階ではその内容を表示せず, ユーザインタラクションによって表示する status=hidden をいう装飾子を追加した. 図 1 のように, 車の詳細データをユーザインタラクションによって表示する場合などに利用できる. そのクエリ例を図 2 に示す.

5.5　embed サーブレット

提案手法のサーブレットは前述した invoke 関数のサーブレットとほぼ同じであるが, Ajax 対応ページを生成するために, 上記拡張を行った. また Ajax に対応するため, UTF-8 を利用する.

6.　従来手法との比較・検討

提案手法のプロトタイプを実装し, SuperSQL における 2 つの関数と提案手法を比較する. 表 1 に 3 つの関数の特徴をまとめた. 項目ごとに 3 種類を比較すると, DB 更新のコストやデータ新鮮度は動的生成である提案手法と invoke 関数が優れている. しかし GET 方式で様々な値を受け渡している invoke 関数はセキュリティ面で問題がある. 提案手法も GET 方式で受け渡しているが, ページ遷移をしないため URL がユーザに見えることはなくセキュリティ面での向上も果たしている. しかしながら js ファイルを閲覧できる状態にしておくと, 様々な情報が漏れる可能性があり, 今後更なる改善が必要と思われるため, ◎ではなく○とした.

次にレスポンスタイムに関しては静的生成である sinvoke 関数が優れているといえる. しかし提案手法も sinvoke 関数と同等のレスポンスタイムを記録している. その実験結果を表 5 に示す. この実験環境はノート PC 上にサーバを用意し, インターネット回線を通さずに実験を行っている. 3 種類のページ遷移に対して 30 回の平均レスポンスタイムを計算した. 提案手法は動的生成であるにもかかわらず, 静的生成である sinvoke 関数と同等のレスポンスタイムを記録している. すなわち提案手法はページの部分更新を可能にすることで動的生成と同等のデータの新鮮度を得つつも静的生成と同等のレスポンスタイムを記録できるようになった, と言える.

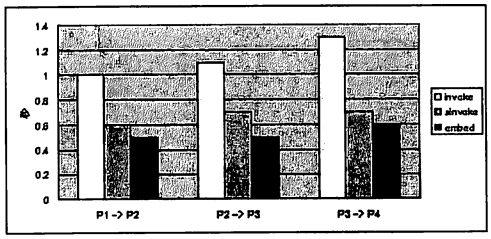


図 5　レスポンスタイム比較

7.　まとめと今後の課題

本研究では SuperSQL を Ajax 対応ページを作成可能にできるよう拡張し, 一部評価を行った. 本研究で Ajax に対応した RIA を比較的容易に作成できるようになったと思われる. 今後の課題は大きく二つある. 1 つはクライアントサイドのロジックである. RIA の特徴のひとつはクライアント側でもデータの操作などができることであるが, 本研究では現在, クライアント側のロジックに関する議論がなされていない. 今後生成できるページをよりリッチにするためには, クライアント側のロジックを実装可能にすることが必要である. もう 1 つは評価である. 今回は SuperSQL における 3 種類の関数のレスポンスタイムによる比較を行った. しかし環境が自己完結状況であるため, きちんとインターネット回線を介した場合の評価をしなければならない. また「RIA を容易に作れる」という点に関しての評価も取る必要がある. 現状では工数による比較を考えている. すなわち RIA を本研究を利用して作成した場合の工数, Ruby On Rails などを利用して作成した場合の工数, およびスクラッチから作成した場合の工数を計算し, 比較する. こうすることでより客観的な評価ができると考えられる.

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] L.D. Paulson. Building Rich Web Applications with Ajax. *Computer*, 38(10) pp.14-17, 2005

[3] J.Allaire.Macromedia Flash MX-A next-generatoin rich client. Technical report, Macromedia, March 2002

[4] 有澤達也, 遠山元道 SuperSQL 処理系における中間データキャッシュの実装と効率化 DEWS2006

[5] 石川 恭子, 有澤 達也, 遠山 元道静的生成されたデータ集約型 Web サイトの効率的な更新手法 DBWS2004

[6] 小林 武彦, 遠山 元道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 DBWS2006

[7] SuperSQL: http://ssql.db.ics.keio.ac.jp/

[8] prototype.js: http://www.prototypejs.org/

[9] google maps: http://maps.google.co.jp

[10] gmail: http://mail.google.com