Web Indexシステムにおける Webサービスとの連携の実現

佐藤　裕紀† 　遠山　元道††

† 慶應義塾大学大学院理工学研究科　〒 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: †hiroki@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし　当研究室では, 利用者主導による Web 文書の結合操作を可能とすべく, 先行研究として, Web における新たな情報資源形態である “Web Index” を提案した. 本研究では, Web Index システムにおける Web サービスとの連携を実現し, Web Index と Web サービスの相互強化や, 相互補完を目的とする. 提案手法として, Web Index と Web サービスを連携したビューとして機能するコンテナを, 上位レイヤーに配置することにより実現する. また, 本手法を含めた Web Index システムの有用性について検討する.

キーワード　Web, Web サービス, 情報統合

Implementation of Coorperation with Web Service in Web Index System

Hiroki SATOU† and Motomichi TOYAMA††

† Graduate School of Science and Technology, Keio University
Hiyoshi 3–14–1, Kohoku-ku, Yokohama, Kanagawa, 223–8522 Japan.
†† Department of Information and Computer Science, Faculty of Science and Technology, Keio University
Hiyoshi 3–14–1, Kohoku-ku, Yokohama, Kanagawa, 223–8522 Japan.
E-mail: †hiroki@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

Abstract　In a precedent study, “Web Index”, as a new resource form in the Web, was suggested in our laboratory. The Web Index enables to join the Web document by end-user’s lead. In this study, we implemented of cooperation with the Web Services in the Web Index system and we aim for mutual reinforcement and mutual complement of the Web Index and Web Service. In technical aspect, we suggest “container” to deploy in upper layer of Web Index. This container plays a role as “the view” to take a mutual cooperation between Web Indices and Web Services. In addition, We evaluate our system about utility.

Key words　Web, Web Service, Information Integration

1. はじめに

1.1 Web Index の誕生

当研究室では, 利用者主導による Web 文書の結合操作を可能とすべく, 先行研究として, Web における新たな情報資源形態である Web Index を提案した. Web Index は, Web 上に点在する他のコンテンツへの「索引」として機能し, 様々な人や団体が独自に作成することが可能である. また, 任意の HTML 文書と, 任意の Web Index とをユーザーの意図で自由に統合することができ, これは, ブラウザで現在閲覧している HTML 文書の所々の文字列を, Web Index 内の Web コンテンツの URL へのハイパーリンクに変換することで実現する. これにより, 従来離散して点在していた Web 上の情報を「結合」することが可能であり, Web コンテンツをより豊かにすることができる.

1.2 Web サービス

近年 Web サービスが注目を集めている. Web サービスとは, XML 形式のプロトコルを用いて, メッセージの送受信を行う技術である. 組織や団体が保有しているデータ等へのアクセス API を外部に提供し, これをクライアント側の Web アプリケーション等のプログラムから利用することで, データが取得できる. 例として, Amazon(商品検索)(注1), Yahoo!(検索サービスなど)(注2)が代表的である. また, Web サービスでは, パラメータを与えると, それに対応するオブジェクトが XML 形式で返却される.

Web サービスの提供プロトコルとして, “REST” 方式と

(注1)：http://www.amazon.co.jp/
(注2)：http://developer.yahoo.co.jp/

“SOAP” 方式の 2 種類があるが, 本論文においては, HTTP の GET メソッドを使ってある URL にアクセスすると XML が返却されるものを REST, HTTP の POST メソッドを使って SOAP メッセージを交換するものを SOAP と便宜上定義する(注3).

1.3 Web Index と Web サービスを相互連携するメリット

まず, Web サービス利用者は, プログラミング技術や Apache(注4)などのサーバソフトウェアの知識が必要であるが, これをラップしたシステムを作成することで, プログラムやサーバを意識せずに Web サービスが利用できる.

また, Web Index は静的ファイルであるため, 最新のデータにキャッチアップしていくことが困難であるが, Web サービスが提供するデータを利用することで, 取得するデータの鮮度が向上し, 特に株価や天気情報といった, 変動が大きいデータに対して有効であると考える.

また, Web Index のキーワードは固定であり, 同位語の集合となっている場合が多く, そのため, Web サービスの有限のパラメータ集合に対して, Web Index のキーワード集合を Web サービスへの入力に用いることで, 効果的に結果のデータ集合を取得することが可能である.

そして, Web サービスと連携した Web Index をアタッチすることによって, Web Index を介して, Web 文書と Web サービス間のマッシュアップが, ブラウザで動的に実行可能になる.

1.4 研 究 目 的

本研究では, Web Index システムにおける Web サービスとの連携を実現し, Web Index と Web サービスの相互強化や相互補完を目的とする.

技術的には, Web Index の上位レイヤーに配置する「コンテナ」を提案する. これが, Web Index と Web サービスとの相互連携をとる「ビュー」の役割を果たす. また, コンテナには 2 種類の機能を用意した.

まず, 連携結果の仮想的な Web Index を生成するビューとして機能する「ビューコンテナ」である. ビューコンテナにアクセスすることは, 内部の Web Index と Web サービスの連携結果である仮想的な Web Index をアクセスすることと等価である. また, 仮想 Web Index 生成時間の簡単な最適化についても検討する.

続いて, 内部にある Web Index をデータ源とした, Web サービスを提供するラッパとして機能する「Web サービス提供コンテナ」である. これにより, 本システムは Web サービスフィードとして機能する.

また, これらの機構を実現するために, コンテナの解析モジュールを開発する.

また, 関連研究や既存の技術と比較することで, 本手法を含めた Web Index システム有用性について検討する.

1.5 論文の構成

本論文の構成として, 2 節では先行研究である Web Index について説明する. 3 節ではビューコンテナについて説明する. 4 節ではビューコンテナの簡略化した記述方式を提案する. 5 節では Web サービス提供コンテナを説明する. 6 節ではビューコンテナの計算の最適化を検討する. 7 節では実装に関して説明する. 8 節では関連研究を紹介する. 9 節では評価検討を行う. 10 節では実験による性能測定を行う. 11 節では結論を述べる.

(注3)：http://itpro.nikkeibp.co.jp/article/Watcher/20060315/232492/?ST=ittrend より抜粋, 2009 年 1 月 18 日アクセス.
(注4)：http://www.apache.jp/

2. Web Index

2.1 Web Index の書式

Web Index とは, Web における結合可能な情報資源である [4]. XML 形式で記述された辞書型の文書であり, 記述例は図 1 のようになる. エントリの集合体となっており, 各エントリ内には, キーワードである見出し語を KW 値に格納し, 項目に対応する参照先文書の URL, または, 文書の HTML 文それ自体を DOC 値に格納する.

```
<WIX>
  <entry>
    <kw>イチロー</kw>
    <doc>http://mlb.com/ichiro</doc>
  </entry>
  <entry>
    <kw>松井</kw>
    <doc>
      <!--<html><head><title>松井</title></head>
～～～～～～～～～～～～～～～～ 中略 ～～～～～～～～～～～～～～～～
      </body></html>-->
    </doc>
  </entry>
～～～～～～～～～～～～～～～～ 中略 ～～～～～～～～～～～～～～～～
</WIX>
```

図 1　メジャーリーグ野球選手情報 Web Index の記述例

2.2 ア タ ッ チ

実際にブラウザ上で任意の HTML 文書と, Web Index を結合する操作をアタッチ (Attach) と呼ぶ. この操作は, 「アタッチャ」と呼ばれる専用のシステムを用いて行われ, アタッチャの例として, 当研究室で開発された専用ブラウザ [3] が挙げられる. 使用する Web Index は, 予めシステム内にブックマークとして登録しておかなければならない. 結合方法は, 「作者指定アタッチ」, 「ユーザー指定アタッチ」, 「フルアタッチ」の 3 種類あるが, 「フルアタッチ」アルゴリズムを紹介する. まず, Web Index 内のエントリの見出し語である KW 値を, 頭から 1 つ取得し, 現在ブラウザで閲覧している HTML 文書内で検索する. 発見したら, HTML 文書内のその文字列を, 同一エントリ内に存在する DOC 値の URL 先, または, HTML 文書へのハイパーリンクに変換し, この操作を全ての KW 値に対して行う. なお, DOC 値が HTML 文書となっている場合は, システム内で参照先の HTML 文書データを文字列型で保持し, ブラウザからリンク展開要求があった際にそのデータをポップアップで立ち上がるウインドウに渡して表示させる.

2.3 過去の研究

著者らはアタッチャを試作した [3]. 高橋らは Web Index の提案, および, 関係データベースのデータを利用した自動 Web Index 生成システムを開発した [4]. 市東らは任意の HTML 文書内のハイパーリンクデータを抽出・利用した, Web Index の

作成支援システムを開発した [5]. 朱らは Web Index システムを利用したハイパーリンク支援方法について考察を行った [6].

3. ビューコンテナ

本節では, コンテナが参照する Web Index と Web サービスを相互連携させ, 仮想的な Web Index を生成するビューの役割を果たす機能について説明する. この場合, アタッチの際にビューコンテナにアクセスすることは, 計算結果の仮想 Web Index をアクセスすることと等価である. なお, ビューコンテナの記述方法としては, XML で記述する場合と, SQL ライクな方法で記述する場合と 2 通り用意した.

3.1 仮想 Web Index 生成過程

ビューの計算方法として, Web Index 中のある値 (KW, DOC など) を Web サービスへの入力とし, その結果を生成結果の Web Index への出力に用いる. 例として, 企業名が KW 値として格納され, そのサイトの URL が DOC 値に格納されている Web Index と, 企業名を与えるとその株価を XML 形式で返却する「株価情報 Web サービス(注5)」が URL“http://stock.com/” 上にあると仮定する(注6). これらを相互連携し, 証券コードを KW 値, 株価を DOC 値となる Web Index を生成するコンテナを作成したいと考える. この場合の流れ図のイメージは, 図 2 のようになる. ここで, コンテナには, 利用する Web サービスや値を取り出すための XPath 情報, Web Index のパス, 出力のテンプレートが記述されており, この情報を読み込んで計算を開始する.

まず, 仮想 Web Index の KW 値には, 元の Web Index の KW 値を一つ取得してはめ込むようになっている. また, 仮想 Web Index の DOC 値には, Web サービスから得られた結果をはめ込むようになっている. そこで, 元の Web Index の KW 値を株価情報 Web サービスに入力し, XML の結果を得る. それを XPath 情報を用いて株価を取得し, その値を出力に利用する. そして, 新たな Web Index のエントリに整形し, 企業名と株価の対となるエントリが得られる. 同様の処理を, 元の Web Index の KW 値全てに対して行い, 最終的なビュー結果を生成する(注7).

3.2 コンテナの記述と記述例

3.1 で例示した仮想 Web Index を生成する XML 型ビューコンテナの記述例は図 3 のようになる. 以下, 例を交えて説明する.

まず, 2 行目, 11-16 行目, 19 行目にある PRINT タグでは, 中の要素を結果として出力している.

次に, 3-4 行目, 6-9 行目の XPATH タグでは, XPath 式により, Web Index や Web サービスの結果である XML データから値の抽出を行う. 例えば, 3 行目では, “Companies.wix(図 4)” という Web Index ファイルから, XPath 式により KW 値の集合を取得している.

(注5) : 本来, 証券コード等, 情報を一意に特定できる値を入力値とすることが一般的だが, 説明の都合上, 企業名を入力値としている.
(注6) : 実在のものとは無関係であるとする.
(注7) : この場合, 元の Web Index の DOC 値は全く使用されない.

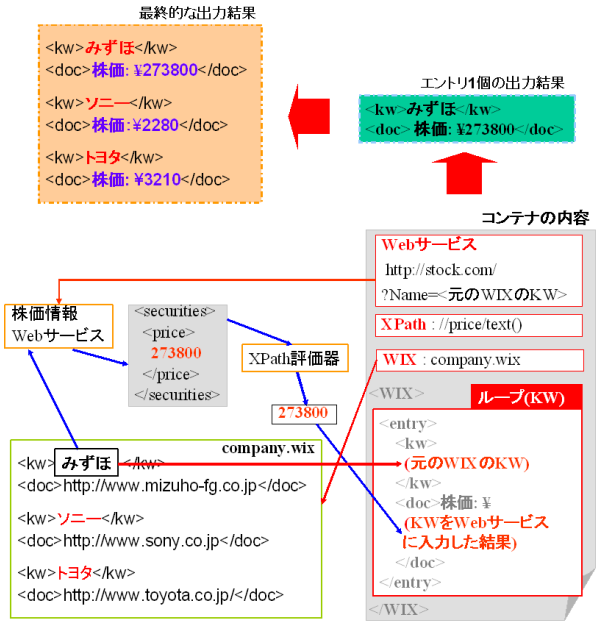

図 2 仮想 Web Index 生成の流れ図

そして, 5-18 行目, 10-17 行目の FOREACH タグで, 処理のループを記述する. なお, ID 属性は XPATH タグの ID 属性に対応している. 5 行目の例で言うと, 3-4 行目の XPath で取得した Web Index の KW 値 1 つ 1 つに対して処理を行うことを意味している.

また, 8 行目, 13 行目, 14 行目にある VAL タグで上記の XPath から抽出した値の参照を行う. なお, これは FOREACH タグ内でしか使用することができない. また, ID 属性は, XPATH タグで指定した ID 属性に対応している. 8 行目の例で言うと, 3-4 行目で取得した KW 値のうち, 5 行目の FOREACH タグのループで現在参照中の KW 値を取得し, Web サービスに与えるパラメータとして機能している.

生成結果は図 5 のようになる.

```
1  <Container>
2    <print>&lt;WIX></print>
3    <xpath id="WIX_KW" href=“ Companies.wix"
4           query="//kw/text()" />
5    <foreach id="WIX_KW">
6      <xpath id="price" query="//price/text()">
7        <href>http://stock.com/
8              ?Name=<val id="WIX_KW" /></href>
9      </xpath>
10     <foreach id="price">
11       <print>
12           &lt;entry>
13             &lt;kw><val id="WIX_KW" />&lt;kw>
14             &lt;doc><val id="price" />&lt;doc>
15           &lt;/entry>
16       </print>
17     </foreach>
18   </foreach>
19   <print>&lt;/WIX></print>
20 </Container>
```

図 3 仮想 Web Index 生成コンテナの記述例

4. ビューコンテナの簡略化

XML 型ビューコンテナを, エンドユーザーが容易に記述可能にする目的で, SQL ライクな形式での記述を提案する. こ

```
<wix>
  <head />
  <body>
    <entry>
      <kw>みずほ</kw>
      <doc>http://www.mizuho-fg.co.jp</doc>
    </entry>
    <entry>
      <kw>ソニー</kw>
      <doc>http://www.sony.co.jp</doc>
    </entry>
    <entry>
      <kw>トヨタ</kw>
      <doc>http://www.toyota.co.jp/</doc>
    </entry>
  </body>
</wix>
```

図 4　XML 型ビューコンテナの利用 Web Index 例

```
<wix>
  <head />
  <body>
    <entry>
      <kw>みずほ</kw>
      <doc>273800</doc>
    </entry>
    <entry>
      <kw>ソニー</kw>
      <doc>2280</doc>
    </entry>
    <entry>
      <kw>トヨタ</kw>
      <doc>3210</doc>
    </entry>
  </body>
</wix>
```

図 5　XML 型ビューコンテナの生成結果例

れは, SELECT 句, FROM 句, WHERE 句に分かれている. SELECT 句では, 生成する仮想 Web Index のエントリの構築方法の指定と, Web Index, Web サービスからの利用する値の抽出を行っており, FROM 句では, 利用する Web Index, Web サービスのデータ源を指定している. WHERE 句では, 生成される Web Index の条件指定を行っている.

記述例は図 6 のようになる. 以下, 例を交えて説明する.

まず, FROM 句を読むと, Web Index に “Companies.wix(図 4 を想定)” を利用する. また, URL“http://stock.com/” 上にある Web サービスを利用している. まず, Web Index からエントリを 1 つ取得する. Web サービスでは, “Name” 変数に Web Index の KW 値を渡す設定になっているため, このエントリから KW 値を抽出し, Web サービスの URL へアクセスし, 結果の XML を取得する.

また, SELECT 句を読むと, 生成する仮想 Web Index の KW 値には元の Web Index の KW 値を, DOC 値には Web サービスから XPath 式で取得した (この時, WHERE 句で設定された名前空間を使用) 株価をそれぞれ代入する設定になっており, 1 つのエントリを構築する. この操作を Web Index 中の全エントリに対して行い, エントリリストを構成する.

次に, WHERE 句の条件によって, 条件付けとエントリリストのフィルタリングを行う. 第 1 トークンでは名前空間の設定を行い, 第 2 トークンでは Web Index のタイプ設定を行っている. 第 3, 第 4, 第 5 トークンでは, エントリリストの中から, KW 値が「ソニー」か「みずほ」か「トヨタ」を持つエントリのみを抽出している. 抽出されたエントリリストから最終的な仮想 Web Index を作成する.

生成結果は図 5 と同様である.

```
SELECT KW,
       WS.//NS:price/text()
FROM Companies.wix WIX,
     http://stock.com/?Name=<KW> WS
WHERE NS='http://stock.com/StockPrices/'
       and TYPE='static-standard'
       and KW=' ソニー'
       or KW=' みずほ'
       or KW=' トヨタ'
```

図 6　SQL 型ビューコンテナの記述例

5. Web サービス提供コンテナ

5.1 概　　要

コンテナにアクセスすると, Web サービスとして機能する. 具体的には, 内部のデータ源として Web Index を利用しており, パラメータを与えると, Web Index からデータを取得して, XML として出力するという形態をとり, コンテナには Web Index からの値の抽出方法を記述する. なお, 予めシステムに Web サービス提供コンテナを登録する必要がある.

図 1 のようなメジャーリーグ選手名鑑の Web Index を Web サービス化する場合のイメージ図は図 7 のようになる. このコンテナに対して, 選手名をパラメータとして与えると, マッチした KW 値に対応する DOC 値を XML 形式で返却する.

また, Web サービスは REST 方式で提供しているため, URL でアクセスを行う. 上記の例で,
“http://www.db.ics.keio.ac.jp” という URL のサーバでシステムが稼動している場合の URL 例は,

http://www.db.ics.keio.ac.jp/WSContainer/
execute?input=ichiro&fileID=mlb

といったものになる. この例では, システム内で ID が “mlb” であるコンテナファイルにアクセスし, かつ, “input” という GET 変数に “ichiro” というパラメータを渡していることになる. また, “execute” は後述するが, 使用するメソッド名である.

また, 本機能の強みとして, 二項関係である Web Index の見出し語と項目の対をデータとして提供できること, Web サービスを提供するためのコーディング作業を省けることが挙げられる.

5.2 コンテナの記述と記述例

5.1 で例示した Web サービス提供コンテナの記述例は図 8 のようになり, 以下, 例を交えて説明する.

まず, 6 行目の INPUT タグでは, URL で入力されたパラメータへの参照を行う. ただし, INDEX 属性は, URL から入力されたパラメータ配列の添え字の番号を表しており, 0 から

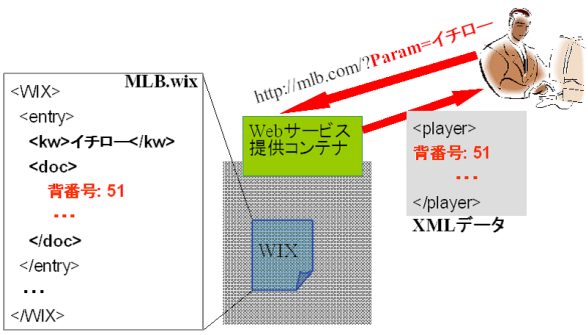

図 7　Web サービス提供コンテナのイメージ

開始する. 5.1 の URL の例で言うと, “ichiro” という文字列が参照される. この結果, 2-8 行目の XPATH ノードでは, URL から受け取ったパラメータを含む, “MLB.wix” から, エントリの DOC 値の集合を取得している.

また, OUTPUT タグの要素を Web サービスの結果として出力する. この例では, 9-13 行目の FOREACH ノードによって, XPath タグで受け取った DOC 値のリストを出力する.

```
1  <Container>
2    <xpath>
3      <id>player_data</id>
4      <wix>MLB.wix</wix>
5      <query>
6       //entry[kw='<input index="0" />']/doc/text()
7      </query>
8    </xpath>
9    <foreach id="player_data">
10     <output>選手のデータ
11             <val id="player_data" />
12     </output>
13   </foreach>
14 </Container>
```

図 8　Web サービス提供コンテナの記述例

5.3　使用可能メソッド一覧

以下に, 提供する Web サービスにおいて, 使用可能なメソッド一覧を列挙する. なお, “*” が付いている変数は, パラメータを複数受け付けることが可能であり, そうでない変数は, パラメータを 1 つ受け付ける. また, “fileID” パラメータは, システムに登録されている Web サービス提供コンテナファイルの ID である.

5.3.1　コンテナ実行メソッド

5.1, 5.2 で説明したように, Web サービスを提供する際にコンテナを読み込ませるメソッドである.

execute(input*, fileID)
input: 入力パラメータ
内容: fileID が示すコンテナファイルを読み込み, input をパラメータとしてコンテナの記述による処理を行い結果を返却する. なお, コンテナがパラメータを必要としない内容であっても “input” パラメータは最低 1 つは必要である.

5.4　プリミティブメソッド

“execute” と違い, システムに登録されている Web Index に対してメソッドを実行することで, コンテナの記述・登録を必要とすることなく Web サービスを提供することが可能である.

getDoc(KW, fileID)
KW: 入力パラメータ
内容: fileID が示す Web Index を読み込み, KW パラメータ値を KW 値に含むエントリ内の DOC 値を返却する.

getKwInSameEntry(KW, fileID)
KW: 入力パラメータ
内容: fileID が示す Web Index を読み込み, KW パラメータ値を KW 値に含むエントリ内の KW 値を返却する (入力値自体の値も含む).

getKwList(fileID)
内容: fileID が示す Web Index を読み込み, その全ての KW 値を返却する.

getWIXList()
内容: システム内で登録されている使用可能な Web サービス提供用 Web Index の ID 一覧を返却する.

getContainerList()
内容: システム内で登録されている使用可能な Web サービス提供コンテナの ID 一覧を返却する.

5.5　使　用　法

下記のような URL にアクセスすることで実行され, XML が返却される.

```
{ サーバ URL}/axis2/services/WSContainer/
{ メソッド名 }/?{ 変数名 }={ 入力値 }[&{ 変数名 }={ 入力値 }]*
```

なお, サーバ URL とは, サーバのトップ URL のことであり, URL よりホストに要求するパスを除いたものであると本論文では定義する. 例として,
“http://www.db.ics.keio.ac.jp/” といった URL を想定する. また, サービスの URL の記述例は 5.1 で説明した通りである.

また, アクセス結果は図 9 のようになる.

6.　ビューコンテナの計算最適化

本節では, ビューコンテナの計算過程の最適化に基づく, 計算時間の短縮を検討する.

6.1　ナイーブアルゴリズムにおける問題点

3 節, 4 節で説明した, 仮想 Web Index 生成の計算過程をナイーブアルゴリズムとする. アタッチャからコンテナをアクセスした場合の問題点として, 仮想 Web Index を生成後にアタッチを行うため, アタッチ結果に無関係なエントリも計算してしまい, Web サービスへのアクセスとエントリの構築過程に無駄が生じる.

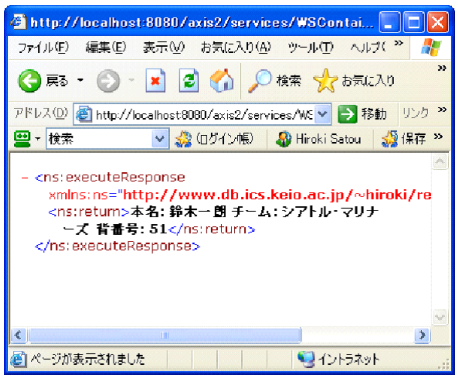

図 9　提供 Web サービスアクセス結果

6.2　提 案 手 法

予め計算過程の一部に「準アタッチ操作」を挿入し, アタッチ結果に関与するエントリのみを取得する. この取得したエントリのみに対して仮想 Web Index 生成計算を行うことで, Web サービスへのアクセスとエントリの構築の負荷が軽減される. Web サービスは, ネットワークを介して外部サーバに接続するため, サーバへの負荷や, 計算時間のボトルネックになる場合が多く, また, 一日の利用回数を制限している Web サービスも多いため, 本手法は有用であると考えられる. ただし, 現在では, 本最適化は SQL 型ビューコンテナに対してのみサポートしており, 以下, SQL 型ビューコンテナについてのみ説明する. なお, 最適化が可能な状況や, その判定方法については, 紙面の都合上省略する.

6.3　準アタッチ操作

準アタッチを下記のように定義する.

準アタッチの定義

Web Index より, アタッチに関与した KW 値集合を抽出すること.

言い換えると, フルアタッチにおいては, Web Index 内の KW 値のうち, アタッチ対象 HTML 文書内に含まれている文字列集合を抽出することであり, ユーザー指定アタッチにおいては, アタッチ対象 HTML 文書内からユーザーが選択した文字列と等しい KW 値が Web Index に含まれていた場合, その値を返却することである.

6.4　最適化の例

図 10 の HTML 文書(注8)を考える. この文書に対して, 4 節で説明した図 6 のビューコンテナをフルアタッチする場合は, 図 11 のような仮想 Web Index が作成される. 図 5 と比べ, 「トヨタ」を KW 値に持つエントリが生成されないことに注目されたい. フルアタッチャはこの Web Index に対してアタッチを行う.

(注8)：http://sankei.jp.msn.com/economy/business/090103/biz0901031302002-n1.htm, http://markets.nikkei.co.jp/kokunai/summary.aspx?site=MARKET&genre=m1&id=AS3L0801N%2008012009, http://www.asahi.com/international/jinmin/TKY200812240217.html, それぞれより抜粋, 2009 年 1 月 8 日アクセス.

また, 図 10 の文書から, 「ソニー」という文字列を選択してユーザー指定アタッチを行うと, 図 12 のような仮想 Web Index が作成され, 「ソニー」を KW 値に持つエントリのみが生成される.

```
<html>
  <head>
    <title>ニュース</title>
  </head>
  <body>
    ・ソニー、計１万６０００人を削減する大リストラ計画<br>
    ・みずほＦＧ 273 万株など売り越し<br>
    ・三洋電機買収<br>
  </body>
</html>
```

図 10　アタッチ対象 HTML 文書の例

```
<wix>
  <head />
  <body>
    <entry>
      <kw>みずほ</kw>
      <doc>273800</doc>
    </entry>
    <entry>
      <kw>ソニー</kw>
      <doc>2280</doc>
    </entry>
  </body>
</wix>
```

図 11　フルアタッチにおける最適化を施行した場合の仮想 Web Index の生成例

```
<wix>
  <head />
  <body>
    <entry>
      <kw>ソニー</kw>
      <doc>2280</doc>
    </entry>
  </body>
</wix>
```

図 12　ユーザー指定アタッチにおける最適化を施行した場合の仮想 Web Index の生成例

7.　実　　装

本システムは, Java で実装した. コンテナファイルの解析は, DOM API で行い, Web Index, 及び, Web サービス結果の XML の解析は XPath 式を用いて行った.

また, アタッチを行う際のシステム構成は, 図 13 のようになり, アタッチする際にはその都度, Web Index ビューコンテナ計算器を呼び出し, 動的に仮想 Web Index の生成計算を行い, 結果を一時ファイルに保存し, この一時ファイルをアタッチャが利用する.

また, Web サービスの実行は, Apache Axis2(注9)を用いて行った.

また, 技術的制約事項として, REST 方式 (URL を用いてアクセスする方式) 以外の SOAP 方式や JSON 形式で提供している Web サービスには対応できない.

(注9)：http://ws.apache.org/axis2/

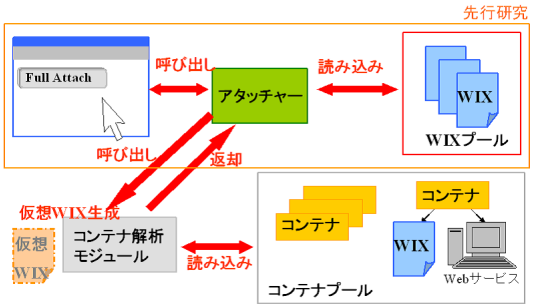

図 13　システム構成

8. 関 連 研 究

Mash Maker [1] は, 現在ブラウザで閲覧中の Web ページに, 他の Web サイトの情報を補足することが可能であり, 「ウィジェット (widget)」と呼ばれる簡易のソフトウェアを用いることで, 参照中のページに対してクライアントサイドでのマッシュアップが可能であり, 結果をインラインフレームで表示する. システム内では, データを木構造によって表現しており, 参照中の HTML を木構造に変換するための「エクストラクタ (Extractor)」を予め定義しておくことで, そのページに対してマッシュアップが可能となる. また, 各ウィジェットは, 公開 API を用いて作成が可能であり, JavaScript 等で記述する. 使用例として, craigslist(注10)で賃貸物件一覧を閲覧中に, Google Maps(注11)表示ウィジェットを利用することで, 各物件の位置を Google Maps にプロットすることが可能である.

また, マッシュアップ作成支援を行っている研究として, 予め登録されている「ブロック (Web サービスのラップしたもの, フィルタリング, ソート, ジオコードなどを行うブラックボックス化されたモジュール)」を繋ぎ合わせることでマッシュアップが作成できる研究や技術も, いくつか報告されている [2], [7], [8].

9. 評 価 検 討

仮想 Web Index 生成コンテナを含めた Web Index システムを, クライアントサイドでマッシュアップが可能な Mash Maker [1] と, マッシュアップの核である Web Index とウィジェットの比較, また, クライアントサイドでのマッシュアップの行われ方を比較し, 有用性を検証した. 結果は表 1 の通りである. 補足すると, コーディングの不必要性項目では, Web Index の DOC 値に URL または, 平易な HTML を記述する場合は, コーディングの必要性がないが, HTML に JavaScript などのプログラムを含めるとコーディング必要があるということを意味する.

Mash Maker では, 閲覧 HTML に Extrator を用意する必要があり, また, ウィジェット内のプログラムでは, JavaScript を利用しているため, 表現能力が高い (高度なアクションも可能である). 一方, Web Index システムでは, マッシュアップをテキストベースで行っており, 表現能力はやや落ちるものの, 任意のページに対してマッシュアップ (アタッチ) を行うことが可能であり, また, Web サービスと連携したい場合は, コンテナを記述することで, JavaScript で記述する場合と比べ, ノンプログラマーにも記述がしやすい. よって, 本研究は, Mash Maker と比較して, 特にノンプログラマーに対してより親和性が高いという点で優位性があるのではないかと考える.

		Web Index	Mash Maker
マッシュアップの核	実体	Web Index, コンテナ	ウィジェット
	作成法	自ら記述, ツール ([4], [5]) を利用	自ら記述・開発
	データ形式	XML(Web Index, コンテナ)	JavaSript, XML
	コーディングの不必要性	○ (Web Index ＋ 平易な HTML), △ (コンテナ利用), × (Web Index ＋ 高度な HTML)	×
	再利用性	○	○
マッシュアップ	アルゴリズム	テキストマッチ	値をウィジェットに渡す
	任意の HTML 文書に対して行えるか	○	△ (閲覧 HTML を木構造に変換する Extrator を用意する必要あり)
	表現能力	○	◎

表 1　クライアントサイドでのマッシュアップが可能なシステムの比較

10. 実　　験

HTML 文書に対してフルアタッチをするという想定の元で, SQL 型ビューコンテナの計算時間を計測し, 最適化施行の有無の計算時間の比較した.

10.1 環　　境

いずれの実験もローカルマシンで行っている. 環境は以下の通りである.

- オペレーティングシステム: Microsoft Windows XP Professional Version 2002 Service Pack 2
- CPU: Genuine Intel(R) 1.06GHz
- メインメモリ: 1024Mbyte

10.2 準　　備

アタッチ対象の HTML 文書は, 図 14 のような 1000 種類の英単語がアルファベット順に記述されたものを想定し, HTML 文書サイズが 200 キロバイトのものを用意した(注12).

Web Index は, エントリ数が 200 の図 15 のような Web Index を用意した. エントリ数の 0%から 100%まで 10%刻のアタッチ率の想定の元で作成した. なお, アタッチ率を「Web Index 内の総 KW 値のうち, 各種アタッチにおいてアタッチに関与した KW 値の割合」と定義する. 例えば, フルアタッチにおいて, エントリ数 200 個, アタッチ率 10%であれば, Web Index 内の 20 個のエントリ内の KW 値に英単語を記述し, その他には “-

(注10) : http://www.craigslist.org/
(注11) : http://maps.google.co.jp/

(注12) : ファイルのメタ情報なども含まれるため, 厳密なファイルサイズではない.

- - - -” を記入した．これは，アタッチ対象の HTML 文書の英単語の種類は固定としたため，アタッチ率の変化を Web Index 側で表現したためである．

```
<html>
  <head>
    <title>***</title>
  </head>
  <body>
********************abandon**********************
********************ability**********************
～～～～～～～～～～～～～～ 中略 ～～～～～～～～～～～～～～～～
********************restore**********************
********************restrain*********************
  </body>
</html>
```

図 14　アタッチ対象の HTML 文書の例 (一部のみ表示)

```
<?xml version="1.0" encoding="Shift_JIS" standalone="yes" ?>
<adoc type="static-standard">
  <header size="100" />
  <body>
    <entry>
      <kw>abandon</kw>
      <doc>-----</doc>
    </entry>
～～～～～～～～～～ 中略 ～～～～～～～～～～
    <entry>
      <kw>-----</kw>
      <doc>-----</doc>
    </entry>
  </body>
</adoc>
```

図 15　ビューコンテナ計算時間測定実験用の Web Index の例 (一部のみ表示)

10.3　実 験 方 法

計測区間は，SQL 型ビューコンテナ解析モジュールを呼び出してから結果が返却されるまでとした．Web サービスは Web サービス提供コンテナを用いたローカルのものを利用し，これとローカルの Web Index とを連携する図 16 のような SQL 型ビューコンテナを作成した(注13)．

```
SELECT KW, WS.//NS:return/text()
FROM http://localhost:8080/RelationalWeb/experiment/testwix/100E90A.wix WIX,
http://localhost:8080/axis2/services/WSContainer/getDoc?kw=<KW>&fileID=250E WS
WHERE NS='*' and TYPE='static-standard'
```

図 16　実験用の SQL 型ビューコンテナの例

10.4　結果と考察

アタッチ率を 0%から 100%まで 10%刻みで変化させた時，および，最適化を行わない時の実行時間を各 10 回測定し，平均値を算出した．結果は図 17 である．

アタッチ対象 HTML 文書が 200 キロバイトと，それほど大きくない状況では，図 17 が示す通り，100%以外は，どのアタッチ率においても最適化を施行した場合の方が実行時間が短い．これは，総計算時間のうちで Web サービス接続時間が大半を占めており，コンテナの解析時間や最適化オーバーヘッドコストは無視できる値であるからである．よって，利用する Web サービスにより接続時間は異なるものの，SQL 型ビューコンテナの最適化手法は有用であると言えるのではないだろうか．

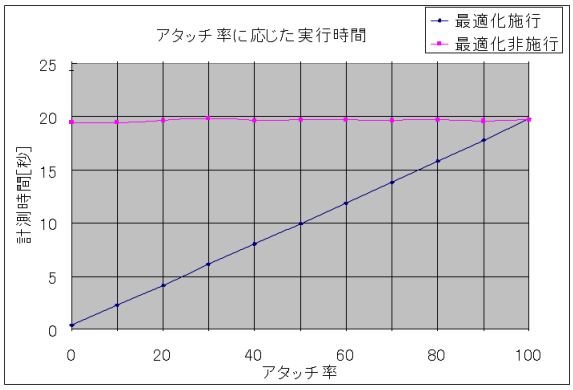

図 17　アタッチ率に応じた SQL 型ビューコンテナの最適化施行の有無における実行時間の比較

11.　ま　と　め

本研究では，Web Index と Web サービスの連携を実現した．これは，連携用のコンテナにより実現し，機能としては，仮想 Web Index を生成する「ビューコンテナ」，Web サービス提供するためのラッパである「Web サービス提供コンテナ」の 2 通りを提案した．ビューコンテナにおいては，記述方式を XML 型，SQL 型の 2 種類用意し，後者については，計算の最適化について検討した．また，有用性に関して，Mash Maker と比較した評価を行い，ノンプログラマーに対して親和性が高いという考えに至った．

また，今後の課題として，SOAP 方式や JSON 形式といった，あらゆる形態の Web サービスに対応できることや，仮想 Web Index の生成のあらゆるニーズに対応できるコンテナの書式整備を行いたい．

(注13)：コンテナは，Web Index の KW 値と，それをローカルの Web サービスにパラメータとして与えた結果を元にエントリを構築するものである．また，11 行目の wix ノードの値は使用する Web Index ファイルに応じて書き換えた．

文　　献

[1] Rob Ennals, David Gay, “User-Friendly Functional Programming for Web Mashups”, Proceedings of the 12th ACM SIGPLAN international conference on Functional programming, pp.223-234, 2007.

[2] Jeffrey Wong, Jason Hong, “Marmite: end-user programming for the web”, CHI’06 extended abstracts on Human factors in computing systems, pp.1541-1546, 2006.

[3] 佐藤裕紀, 遠山元道, A-doc ファイルのアタッチの機能を持つ専用ブラウザの試作, DEWS 2007.

[4] 高橋健太郎, 遠山元道, SuperSQL による A-doc ファイルの生成, DEWS 2007.

[5] 市東隼, 遠山元道, A-doc に基づく WEB リンク集再利用化ツールの試作, DEWS 2008.

[6] 朱成敏, 遠山元道, A-doc に基づく閲覧者主導のハイパーリンク支援, DEWS 2008.

[7] Yahoo! Pipes: http://pipes.yahoo.com/pipes/.

[8] Microsoft Popfly: http://www.microsoft.com/japan/msdn/vstudio/popfly/default.aspx.