結合ビューや複数ビューを提供するWebデータ閲覧・編集環境におけるデータ整合性管理

熊谷　良夫† 　仙波　雅也† 　鎌田十三郎†

† 神戸大学大学院システム情報学研究科情報科学専攻
〒 657-8501 兵庫県神戸市灘区六甲台町 1-1
E-mail: †{kumagai,senba,kamada}@cs26.scitec.kobe-u.ac.jp

あらまし　現在，様々な Web サービスが提供されており，個人やグループのデータを格納・編集できるサービスも各種存在する．一方で，Web アプリケーションの操作性も向上しており，複数のウィンドウで Web データの閲覧・編集を行えるような GUI フレームワークも提供されている．我々のグループでは，複数の Web サービスに対する結合ビューを提供し，その結合ビュー上でのデータ編集を可能とする計算環境を提供している．結合ビュー上で編集操作を行った場合，システムは結合ビュー上の影響範囲を再計算する．本研究では，同一データ要素が複数のビュー（結合ビューを含む）に出現している際のデータ整合性およびシステムの挙動をモデル化し，加えて，編集操作によって編集箇所が消失するようなケースの判定方法を示し，その対処法の検討も行う．

キーワード　データ整合性，情報統合，Web アプリケーション

1. はじめに

現在，様々な情報が Web サービスより提供されている．サービスの中には，個人やグループのデータを格納・編集できるものも存在する．例えば，Google Calendar API [3] を使うことで，ユーザは自身の予定を検索することや，新しく作成もしくは修正した予定を登録することができる．

一方で，操作性の高い各種コンポーネントを備えた Web アプリケーション（以下，Web アプリ）を作成するための GUI フレームワークも提供されてきている．アプリ開発者は，Web サービスと通信を行う仕組みや，取得したデータを表示するための GUI コンポーネントを利用でき，Web サービスを利用した様々なデータを扱う Web アプリを作成可能である．

我々の研究グループでは，異なる Web サービスから取得したデータを結合し，1 つのビューとしてデータの閲覧や編集ができる JoinedViewEditor と呼ばれる機能を提案・実装している [7]．本機能は，Web アプリケーション開発用の JavaScript フレームワークである Sencha Ext JS [6] の上に構築されたもので，GUI コンポーネントも含めて提供している．開発者は，結合したい複数の Web サービスとその結合条件を指定するだけで，簡単に結合ビューを得ることができる．Web サービスへのデータアクセス要求は，システムによって自動的に行われる．システムは，Web サービスからテーブル形式のデータを取得し，指定された条件に従って左結合演算を行う．エンドユーザは，GUI 上のエディタを通して，結合ビュー上でデータ編集が可能である．システムは，ユーザの編集内容に応じて結合条件の再評価を行い，編集を反映した結合ビューを提供する．また，データ編集は即座に Web 上に反映されるわけでなく，ユーザは，編集による影響を確認してから，その編集を保存するか取り消すかを決定することができる．

本研究の目的は，同一データが複数箇所に出現する場合におけるデータ整合性保持と，結合ビューの自動更新をモデル化し，その実現を図ることにある．データ要素は，複数のビューに同時に出現することや，単一結合ビューの複数箇所に出現することがある．論文 [7] では，1 つの結合ビュー内における整合性保持には対応していたが，複数ビュー間でのデータ整合性については対応できていなかった．

本研究では，ビュー間のデータ要素共有と結合ビューの再評価の仕組みをモデル化し，その上で，データ編集によるビューへの影響範囲に関する議論を行う．加えて，データ編集により編集対象自体が結合ビューから消失するようなケースや，結合ビューに対する編集箇所が別の編集操作によって消失するようなケースなどの，好ましくない結果を引き起こしかねない編集操作の分類を行い，その対処案を検討する．

論文の構成は，以下の通りである．まず，2. 節にて関連研究の紹介を行い，3. 節にて我々が取り扱いたい結合ビューの閲覧・編集状況の紹介ならびに関連研究との差を紹介する．4. 節にて，データ要素と結合ビューおよびその自動更新のモデル化を行い，5. 節にて，拒否すべき編集操作や警告すべき編集操作に関する議論を行う．

2. 関連研究

2.1 マッシュアップツール

容易にマッシュアップを実現するためのツールが，各種提供されている．ただし，Web サービスに対する CRUD 操作を想定したものは少ない．

Yahoo! Pipes [10] や Popfly [9] は，Web サービスの接続関係を GUI 操作などにより指定できるようにすることで，開発者が，より簡単にマッシュアップを行うことを目指したツールである．例えば Yahoo! Pipes では，各種 Web サービスに対応する Source や，ソート操作やフィルタ操作を表す Operator などが用意されており，開発者はこれらをつなぐことでマッシュアップを作成する．Yahoo! Pipes は，主にマッシュアップ結果を RSS feed として出力することを想定しており，Popfly はマッシュアップ結果を表示するための UI blocks を備えるが，CRUD 操作は想定していない．

論文 [5] で提案されたマッシュアップ環境はインタラクティブなビューアを提供し，システムはユーザの操作に応じて画面表示に必要な要素についてのみ要求駆動でデータ合成を行う．ただし，Web サービスからデータを取得するのみであり，やはり CRUD 操作は提供されていない．

Mash Maker [1], [2] ではマッシュアップ機能を持つウィジェットを公開しており，ユーザは，ウィジェットを使って Web ページに異なるサイトから取得した情報を統合できる．このウィジェットは，エキスパートユーザによって開発されている．Mash Maker は木構造のデータモデルと関数型言語 (Mash Maker Language) を提供しており，ウィジェット開発者は，Web サービスに問い合わせ，その結果を木構造データに付け加えるようなプログラムを記述できる．データを Web サービスに送って保存するような機能を付加したい場合は，対応する Web サービスリクエストを発行するようなプログラムを記述することになる．

2.2 結合データのためのエディタ

結合演算は，データベース管理システムにおいてよく利用される演算である．そして，いくつかのデータベースシステムは，テーブルの結合および結合ビューの閲覧が可能な GUI 環境を提供し，中には，結合ビューを通してソースデータの編集が可能なものも存在する．Microsoft Access が提供する環境では，データシートビューを用いることで，結合ビューを通してのデータ編集が可能である．ただし，状況によって，一部または全てのカラムが編集できないことがある [8]．いくつかの特徴的なケースについては，5.1 節で改めて説明する．編集はローカルにあるデータベースに即座に反映され，加えてクエリを再評価することでビューを常に結合条件が満たされた状態に保つ．

Google Fusion Tables [4] は，Google Docs に保管しているテーブル形式データを扱い，ユーザ間でのデータ共有も可能である．ユーザは，見ているテーブルに対して属性を選び，それを結合条件に用いて別のテーブルを左結合することができる．また，各種データ形式に応じて，地図や棒グラフなどの表示形式も選ぶことができる．Google Fusion Tables でも，ビュー上で属性値の編集が可能であるが，結合データをテーブル形式で表示した際には，結合条件に指定した属性の値は編集できない．

2.3 Sencha Ext JS

Sencha Ext JS [6] は，インタラクティブな Web アプリケーションを構築するための JavaScript フレームワークである．主な特徴として，開発者は様々なウィジェットを利用でき，MVC (Model-View-Controller) モデルに従った実装が可能である．利用可能なウィジェットとして，テーブルビューア，ウィンドウ，メニューバー，ボタン，チャートなどがある．なお，本論文は Ext JS をベースとして，提案環境の構築について研究を行っている．

開発者は，オブジェクトの構造と型を Model として定義できる．Store は，Model インスタンスの集合を格納するものである．開発者が Store の load メソッドを実行すると，クエリ条件としてあらかじめ設定したフィルタと共に，CRUD 要求が Proxy を介して Web サーバに送られ，その結果が Model インスタンスとして Store に格納される．また，検索結果を複数のページに分割する Web サービスのために，Store の 1 ページ当たりの要素数 (pageSize) を指定可能である．フィルタは Web サーバへの問い合わせ時に使われるだけでなく，Store に格納された Model インスタンスに対して，ローカルにフィルタを適用する使い方も可能である．このフィルタでは，開発者は，独自の条件を記述したユーザ定義関数を適用することができる．Proxy は，サーバの URL や返り値のフォーマットを定めたものであり，Web サービスにつき，1 つ指定する．Web サーバに要求を送る方式として，Ajax や JSONP などの種類が Proxy には用意されており，開発者は，種類を指定するプロパティを変更するだけで，各々の方式の Proxy を利用できる．

ExtJS の提供するテーブルビューアなどのデータビューアは，Store のデータ更新に応じた描画処理を行っており，Ext JS は，データを表示するビューアと，データを格納・管理する Store に分離した MVC モデルを実現している．具体的には，開発者が Store に Model インスタンスを追加／削除したり，既にある Model インスタンスのデータを修正したりすると，データ変更のイベントがその Store より発行される．Store のデータを表示しているデータビューアは，そのデータ変更のイベントに応じて，自身の再描画処理を行うレンダラを実行する．

Ext JS は，データモデルに関していくつかの高度な機能を提供しており，その 1 つとして，複数の Model 間の一対多の関係を扱える仕組みを Model クラスより提供する．例えば，とある評価コメントがあるレストランを参照している時，開発者は HasManyAssociation を定義でき，そのレストランに関連した評価コメントの一覧を検索し，その結果を Store として返すメソッドを利用可能である．

3. 事例検討

本節では，本研究の動機となるアプリケーション事例を紹介し，提案システムの動作について説明する．また，関連システムとの動作の違いについても紹介する．

図 1 は，飲み会情報を管理するアプリ例である．アプリ開発者は，例えば飲み会開催情報とレストラン評価コメント用のサービスは自分で準備し，レストラン情報については外部の Web サービスを利用するといったことが可能である（図上）．

開発者は，各飲み会開催情報に対して，開催場所のレストラン情報とレストランへの自分の評価コメント（たかだか一つ）を左結合する形で結合ビューを作成し，図下方の Grid 画面上

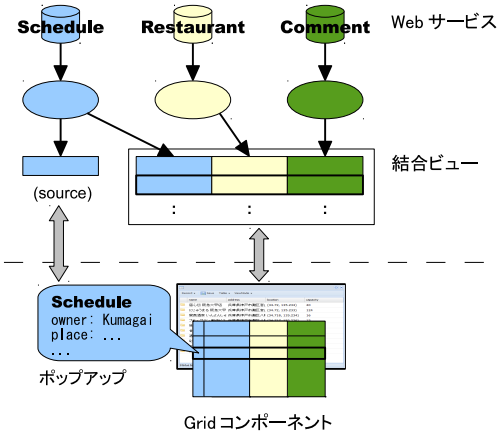

図 1　飲み会アプリの例

で表形式表示している．同様に，同一結合ビュー上のレストラン位置属性を，Map 上のマーカとして表示している．ユーザは，Grid 画面上の結合ビューを通してデータ編集することもできるが，開催情報／レストラン情報／コメント一覧それぞれに対応したアイコンをクリックすることで，当該データをポップアップ画面に表示し，その画面を通してデータ編集することも可能である．図は，開催情報をポップアップ表示した場合を示している．

アプリ利用者は，結合ビューを通して関連情報を比較一覧することが可能であり，そのビュー上で，既にある飲み会情報を編集することや，新たに飲み会情報を作成することができる．評価コメント欄については，例えば，飲み会参加後に，その評価を書き込むといった使い方が想定される．

以下，利用者の利用事例を通して，システムの動作について簡単に紹介する．利用者が新たな飲み会開催情報を作成したい場合，Grid 画面左上にある追加ボタンをクリックする．追加された開催情報の開催日時などの各項目は空であり，利用者は各項目に書き込むことや，ドラッグ&ドロップ操作により過去の飲み会開催情報の各項目の内容をコピーすることができる．今，過去に利用して評判の良かったレストランを再度訪れたいと考え，そのレストランを利用した開催情報の開催場所の項目からレストラン名をドラッグし，現在作成中の開催情報の開催場所の項目にドロップする．すると，それまで空であったレストラン情報とその評価コメントの場所に，入力した開催場所のレストランに該当する情報が表示される．このとき，システム内部では，ドラッグ&ドロップにより入力されたレストラン名に合致するレストラン情報および評価コメントを Web サービスから取得し，その情報を載せている．

Grid 画面上での編集が面倒な場合は，例えば開催情報に関するアイコンをクリックすることで，ポップアップ画面に選択した飲み会の開催情報をフォーム形式で表示し，編集することが可能である．当該要素は，複数画面上に表示されることになるが，例えば利用者がポップアップ画面上で飲み会開催情報の開催日時を編集した場合も，Grid 画面上の対応箇所に編集結果が即時に反映される．

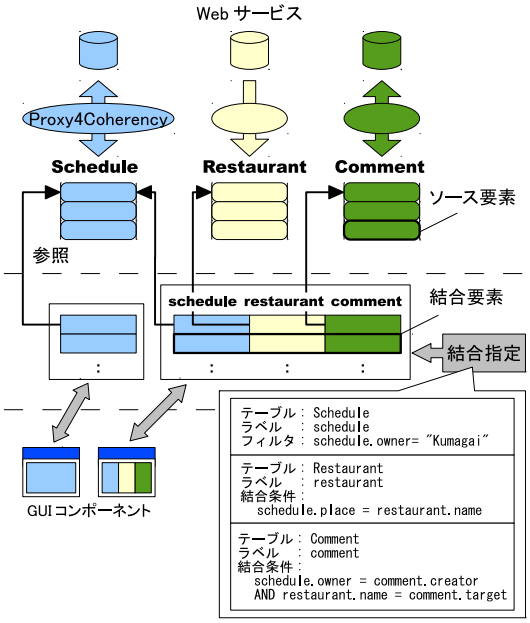

図 2　データ要素とビューとの関係

4. モ デ ル

本節で示すモデルは，ビュー上に複数出現するデータ要素と結合ビューとの関係を表現したものであり（図 2），システムの動作仕様についても，本モデルを用いて記述する．なお，本モデルで用いたデータ表現は，現実の実装を表したものではない．

本モデルでは，Web サービスを**テーブル**として扱い，テーブル中の各データ要素を**ソース要素**と呼ぶ．Web サービスから取得した要素ならびに，Web サービスに登録するために新規生成した要素が該当する．ソース要素は，列に相当する複数の属性とその属性値を持つ．Web サービスへの問い合わせは，我々が Proxy4Coherency と呼ぶ Proxy を介して行う．Proxy4Coherency は，ロードしたソース要素の管理を行い，同一要素を表すソース要素が複数存在しないように取りまとめ，編集の一本化を図る．Web サービスへの問い合わせ結果はソース要素のリストとして表現され，通常 Store クラスに格納される．

システムは，複数テーブルの結合を許している．テーブル結合には左結合演算を使用しており，加えて，一つの左要素に対して，結合条件を満たす右要素をたかだか一つ見つけてきて結合を行う．開発者は，システムに対して**結合指定**を与える．結合指定は，結合対象となるテーブルごとに，そのテーブル名，識別のための**ラベル**（同一テーブルの結合を考慮），および**結合条件式**を持つ．また，結合順序は，結合指定に記述された順となる．以後結合対象のテーブルを，そのラベル（記号 L などを利用）で識別する．結合条件式には，対象テーブルの属性（**適合属性**と呼ぶ）に対する条件を，そのテーブルよりも左に位置するテーブルの属性（**選択属性**と呼ぶ）の値もしくは定数を用いて表現した条件式か，もしくはその AND 結合式が許される．テーブル L の属性 a は $L.a$ として表記する．また，テーブル L の結合条件式で用いられた選択属性の集合を $Selectors(L)$

として表す．**最左**テーブルについては，結合条件式の代わりに，その属性値に関するフィルタ条件のみが与えられる．

システムは，結合指定を受けて結合ビューの作成を行う．結合ビューは**結合要素**のリストであり，上記フィルタ条件を満たす最左テーブルの各要素に対して，結合要素が一つ作られる．結合要素は，各右テーブルに対してその結合条件式を満たすソース要素を検索し，その参照を保持する．結合要素 j のテーブル L に対応するソース要素を $j.src(L)$ として表記する．結合要素は，参照元のソース要素上での値変化を監視し，$L_0.a_0 \in Selectors(L_1)$ の値が変化した場合は，結合条件を満たすテーブル L_1 の要素を検索し，j が持つ参照を置き換える．結果，L_1 の各属性の値が変化し，さらなる右テーブル要素への参照を再評価が行われうる．属性 $L_0.a_0$ の値変更により，再計算を行うテーブル群を $L_0.a_0$ の**影響範囲**と呼び，$Dominated(L_0.a_0)$ で表す．$Dominated(L_0.a_0)$ は以下の 2 式で定められる．

$$L_0.a_0 \in Selectors(L') \Rightarrow L' \in Dominated(L_0.a_0) \quad (1)$$

$$L'.a' \in Selectors(L'') \wedge L' \in Dominated(L_0.a_0) \Rightarrow L'' \in Dominated(L_0.a_0) \quad (2)$$

図 2 の結合指定を例にとると，`schedule.place` が $Selectors$(`restaurant`) に含まれるので，(1) より

$$\texttt{restaurant} \in Dominated(\texttt{schedule.place}) \quad (3)$$

である．加えて，`restaurant.name` $\in Selectors$(`comment`) と (3) から，(2) より

$$\texttt{comment} \in Dominated(\texttt{schedule.place})$$

も成り立つ．これは，ある結合要素 j において，$j.src$(`schedule`) の `place` 属性の値が変更されると，j の `restaurant` および `comment` テーブル部が再評価対象となり，結合条件を満たす要素への参照に置換されることを意味する．また，j を持つ結合ビュー上での編集だけでなく，例えば `Schedule` テーブルを見ている別の単一ビュー上での編集であっても，`Schedule` のソースが変更された場合，結合要素 j にて上述の再評価が発生する．

結合要素は，適合属性の値の変化も監視しているが，その取り扱いについては，次節で述べる．

5. 編集箇所の消失とその判定法

提案システムでは，編集に応じて結合条件の再評価を行い，編集を反映した結合ビューを提供する．ただし，場合によっては，編集によって編集対象自体が結合ビューから消失するようなケースや，結合ビューに対する編集箇所が別の編集操作によって消失するようなケースなどが発生しうる．これら状況を，事例を通して説明したのち，データモデルを用いて整理し，対処案を検討する．

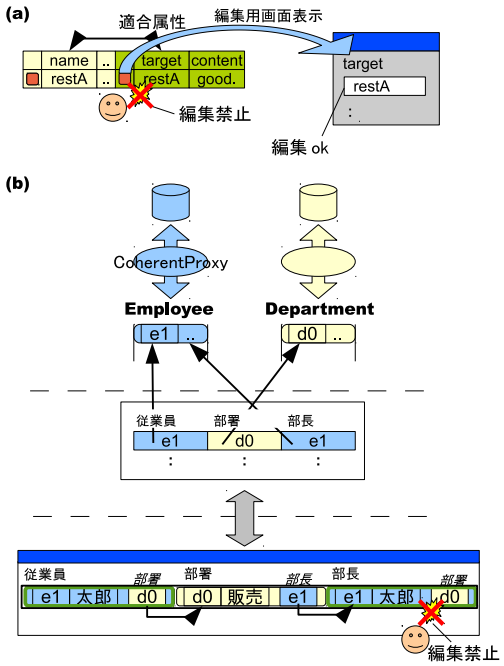

図 3 編集対象自体が消失するケース

5.1 編集対象自体が消失するケース

本システムでは，結合ビューを通しての適合属性の編集を禁止している．これは，編集操作によって結合条件が満たされなくなる恐れがあるためである．システムが結合条件の再評価を行い，別のソース要素に切り替えた場合，編集直後に編集対象ソース要素が，ビュー上から消失することになってしまう．例えば，図 3 (a) の状態において，結合ビュー上での過去の評価コメントが別のレストランを対象にしたものであると利用者が気付いたとする．その際，結合ビュー上で `comment.target` 属性を修正できてしてしまうと，書き込みと同時に，記述対象がビュー上から消失することになってしまう．ただし，上記はあくまで結合ビューを通しての編集を禁止したものであり，当該コメント要素を編集したければ，編集用ポップアップ画面を表示し，その上で編集することは可能である．結合要素は，ソース要素上での適合属性値の変化を監視しており，修正に応じて他のソース要素に置き換えるべきか判断するため，結合条件の再評価を行う．なお，結合条件に等価判定を用いている場合，適合属性と選択属性は同じ値であるため，結合ビューを表示する際は適合属性側の表示を off にすることが多いことを付記しておく．

編集対象自体の消失は，別の状況においても起こりうる．テーブル $L_1.a_1$ への属性値編集が，$L_0.a_0$ への属性値変化を引き起こし，$L_1 \in Dominated(L_0.a_0)$ であれば L_1 のソース要素がビュー上から消失する可能性がある．我々のシステムでは，あるソース要素への書き込みが別のソース要素への編集を引き起こすことはないが，$L_0.a_0$ と $L_1.a_1$ が同一ソース要素の同一属性である場合を考慮しなくてはいけない．図 3 (b) の例では，従業員である太郎と部長である太郎が同一人物であり，「部長. 部署」の変更が「従業員. 部署」の変更を意味するため，結果「部長太郎」が消失することになる．このような状況を避ける

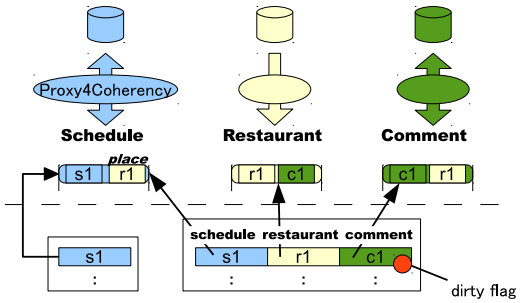

図 4　以前書いた編集が消えるケース

ために，我々のシステムでは，結合要素 j の $L_1.a$ への書き込みは，$j.src(L_0) == j.src(L_1) \wedge L_1 \in Dominated(L_0.a)$ の場合，編集を禁止している．

ここで比較として，Microsoft Access における結合ビュー上での編集について説明する．Access では，多対一関係を持つ結合要素について，結合条件に使われている属性（結合フィールドと呼ばれる）のうち，“多” 側の結合フィールドの編集を許可している．ただし，フィールドの編集についてはいくつかの制限を設けており，公式サイト [8] では，Access 2007 における下記の制限について説明されている．

(1)　一対多リレーションシップで結合したビュー上で，“多” 側の結合フィールドが “一” 側の同じ行にある場合，“多” 側の結合フィールドを編集できるがビューの更新は即座に行われず，一度コミットする必要がある．

(2)　3 つ以上のテーブルを結合していて，多対一対多リレーションシップが存在する場合，全てのカラムが編集不可能である．

また，Access 2010 にて挙動を確認したところ，一対多結合において “多” 側の結合フィールドは基本的に編集でき，結合ビューの更新も随時行われる．また，“多” 側の結合フィールドの編集により，編集対象の行が消失してしまう場合，警告が出て編集をコミットできないことがわかった．ただし，図 3 (b) の例で「部長. 部署」の編集を試みたところ，同時にその編集が「従業員. 部署」に反映されるが,「従業員. 部署」の変更に応じた影響範囲の再計算は,「従業員＝部長」の場合に限って行われなかった．すなわち，循環参照がある場合は，結合ビューの反映が限定的に行われることを確認した．

5.2　以前書いた編集が消えるケース

本節では，編集によって以前の結合ビュー上の編集が消失する場合について検討する．

本システムでは，データ編集は即座に Web 上に反映されるわけではなく，利用者は一連のデータ編集の影響を確認した上で，データを Web 上に保存するかどうか判断することができる．このため，ビュー上には未保存の編集要素が存在する．例えば，図 4 の場合，結合ビューを通して結合要素 j の `comment.text` が編集されていたとする．仮に，この状態で `Schedule s1` の `place` 属性が編集されたとすると，$j.src$`(restaurant) (= r1)` や $j.src$`(comment) (= c1)` が結合ビュー上から消失する可能性があり，ユーザは自分の編集箇所がどこであったのか見失う恐れがある．また，複数の編集間に衝突を起こしている可能性も考えられる．

このため，結合ビューを通してソース要素を修正した場合，結合要素は当該ソース要素への編集があったことを記録し（図中 dirty flag），そのソース要素の編集が保存されるまで保持する．加えて，ソース要素における選択属性相当における編集操作を監視しておき，ソース要素 $j.src(L_0)$ において属性 a の編集が行われようとした際，テーブル L_1 の要素が dirty であり，かつ $L_1 \in Dominated(L_0.a)$ が成立する場合に警告が行われるものとする．

6.　ま　と　め

本研究では，同一データ要素が複数のビュー（結合ビューを含む）に出現している際のデータ整合性およびシステムの挙動をモデル化し，加えて，編集操作によって編集箇所が消失するようなケースの判定方法を示し，その対処法の検討を行った．モデルでは，Proxy4Coherency を介し，同一要素を表すソース要素が複数存在しないように取りまとめ，編集の一本化を図った．

謝　　辞

本研究の一部は，科学研究費補助金基盤研究 (C) 22500091 の補助を受けたものである．

文　　献

[1] Rob Ennals, Eric Brewer, Minos Garofalakis, Michael Shadle, and Prashant Gandhi, “Intel Mash Maker: join the web,” *SIGMOD Record*, Vol. Vol. 36, No. No. 4, pp. 27–33, (2007).

[2] Rob Ennals and David Gay, “User-friendly functional programming for web mashups,” *in the 2007 ACM SIGPLAN international conference on Functional programming (ICFP ’07)*, pp. 223–234, ACM, (2007).

[3] Google, “Google calendar api,” `https://developers.google.com/google-apps/calendar/v3/reference/`.

[4] Google, “Google fusion tables,” `http://www.google.com/fusiontables/Home/`.

[5] Sohei Ikeda, Takakazu Nagamine, and Tomio Kamada, “Application framework with demand-driven mashup for selective browsing,” *Journal of Universal Computer Science*, Vol. 15, No. 10, pp. 2109–2137, (2009).

[6] Sencha Inc., “Sencha docs ext js 4.0,” `http://docs.sencha.com/ext-js/4-0/`.

[7] Yoshio Kumagai, Masaya Senba, Takakazu Nagamine, and Tomio Kamada, “Joined view editor for mashups of web data stores,” *SNPD2012*, (2012).

[8] Microsoft, “Edit data in a query (applies to: Microsoft office access 2007),” `http://office.microsoft.com/en-us/access-help/edit-data-in-a-query-HA010097876.aspx`.

[9] Microsoft, “Microsoft popfly,” `http://www.popfly.com/`.

[10] Yahoo!, “Yahoo! pipes,” `http://pipes.yahoo.com/pipes/`.