A2-6

ACTIVIEWにおける適応型表提示の最適化

前田葉子 †　遠山元道 ‡

† 慶應義塾大学院 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: † yoko@db.ics.keio.ac.jp, ‡ toyama@ics.keio.ac.jp

近年の携帯電話や PDA などの小型携帯端末の普及は、WWW へのアクセス手段を多様なものとしている。したがって、閲覧環境ごとに適応したページを提供する技術が必要とされてきている。SuperSQL を拡張して実現した ACTIVIEW では、データベースに格納したデータをダイナミックに HTML ページとして WWW を通して提供する際に閲覧画面のサイズを考慮して表のレイアウトを変換することで表示ビューの適応化を目指した。ここで行われたレイアウト変換は簡単な発見的手法を用いただけに過ぎず、十分な最適化が実現されていなかった。そこで、本稿ではレイアウト変換について最適なレイアウトの基準を定め、評価関数を導入し、最適なレイアウト変換を実現することを目指す手法を提案する。

キーワード : SuperSQL 、 WWW 、 アダプテーション 、 ビュー

The optimization of adaptive data presentation at ACTIVIEW

Yoko Maeda †　Motomichi Toyama ‡

†School of Open and Environmental System, Faculty of Science and Technology, Keio University.
‡Department of Information and Computer Science, Faculty of Science and Technology,
Keio University.
E-mail : †yoko@db.ics.keio.ac.jp ‡toyama@ics.keio.ac.jp

Recently, wireless PDA (Personal Digital Assistant) and portable phone becomes very popular and it requires to create the HTML pages which adapt to user's browsing environment. We propose an adaptive data presentation system (ACTIVIEW) , which is an extention of SuperSQL, enables to create HTML pages dynamically adapt to the size of browser. We changed layout based on few heuristic rules. Therefore we could not say the answer is optimized. In this paper, we introduce the cost estimation function and propose the optimization algorithm for layouts.

1 はじめに

近年、携帯電話や PDA などの小型携帯端末が急速に普及する一方で、家電ネットワークやウェアラブルコンピュータなどの研究が進み、種々多様な WWW へのアクセス手段が考えられる。この流れに伴い、より多くの人が満足するコンテンツを作成するよりも、それぞれの個人が満足するページを提供する事でページを利用する人全ての満足度を高めることに主眼が置かれるようになってきている。そこで、個人の閲覧環境に合わせたページを提供する適応化の技術が必要とされている。

本研究では、Web を通して多くのユーザがデータベースに格納している有用な情報を共有する状況において、小型携帯端末を含む様々な端末に適応したデータの提供方法として ACTIVIEW(適応型表示表示ビュー) を実現した。SuperSQL を拡張して実現した ACTIVIEW では、データベースに格納したデータをダイナミックに HTML ページとして WWW を通して提供する際に閲覧画面のサイズを考慮して表のレイアウトを変換することで表示ビューの適応化を目指した。ここで行われたレイアウト変換は簡単な発見的手法を用いただけに過ぎず、十分な最適化が実現されていなかった。そこで、本稿ではレイアウト変換について、変換前の平坦な表から起こりうる全てのレイアウト変換パターンを定め、新しく評価関数を導入し、評価関数の値を利用して最適なレイアウト変換を実現することを目指す最適化の手法を提案する。

2 ACTIVIEW

2.1 ACTIVIEW とは

ACTIVIEW は SuperSQL の新しいメディアとして提案した。SuperSQL とは SQL を拡張した問い合わせ言語を用いて、データベースに問い合わせた平坦な構造の出力結果を構造化し、指定されたメディアの応用データへ変換するシステムである。SuperSQL が特徴とするデータベースの出力の多彩なレイアウト機構を利用することで、従来からある HTML メディアの出力をクエリーで指定した固定のレイアウトではなく、表示画面に適応するよう自動変換することを実現した。

ACTIVIEW を生成するクエリーは、SuperSQL クエリーと基本は同じく、SQL の select 節を出力メディア、属性間の結合子と属性の装飾子と共に記述する。レイアウト変換を実行する際に重要な役割を担うのが結合子であり、結合子の種類と意味を以下に説明する。使用例の x は属性値とする。

結合子	連結の種類	使用例	レイアウト式
,	横連結	x_1 , x_2	(C1 1 2)
!	縦連結	x_1 ! x_2	(C2 1 2)
%	ハイパーリンク	x_1 % x_2	(C3 1 2)
[ ],	横連結反復	$[x_1]$,	(G1 1)
[ ]!	縦連結反復	$[x_1]$!	(G2 1)
[ ]%	深さ方向反復	$[x_1]$%	(G3 1)

＜例＞

SuperSQL クエリーの例を図 1 に示す。クエリーから生成されるレイアウト式やレイアウトの木構造表現、生成される表も同様に図 1 に示す。この例では employee 値によってネストする。

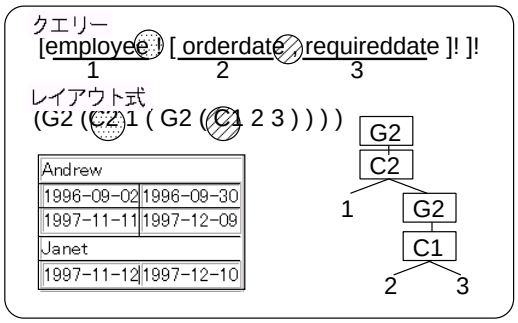

図 1: SuperSQL クエリー例

2.2 システム構成

ACTIVIEW を実現するシステム構成について簡単に説明する。図 2 において太線で囲まれる部分が SuperSQL 処理系である。SuperSQL 処理系では、まず受け取った SuperSQL クエリーを通常の SQL と構造情報を含むレイアウト式に分解する。ACTIVIEW ではここで、表示画面サイズに適応するようにレイアウト式を変換する。SQL によってデータベースから受け取ったデータと変換後のレイアウト式を元に HTML ページを生成し表示する。

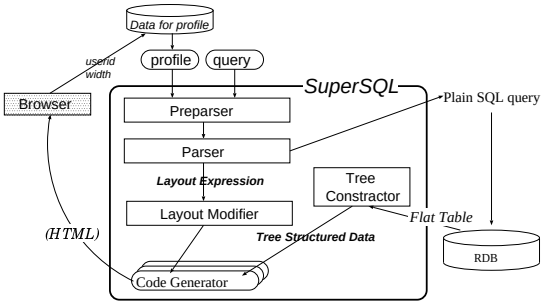

図 2: システム構成

3 最適化

ACTIVIEW では表示画面サイズに合わせて、表示する表のレイアウトを変換することで適応化を実現する。表示画面よりも大きな表が表示された場合、横にも縦にもスクロールする必要がでてくるので不便であり特に携帯電話などではスクロール自体が容易ではない。このような問題に対処するため表示画面の横幅に収まるようなレイアウトに自動変換してデータを表示するものとした。したがって、基本は横連結であったデータを縦連結に変換する。また、携帯電話に代表される非常に小さな画面の場合はハイパーリンクを利用することで、限られた画面への必要の無いデータの表示はなるべく避け、欲しい情報を早く表示する解決法として縦方向の連結よりも、深さ方向の連結 (ハイパーリンク) に変換する。ここで問題となるのが、どの横連結を縦連結または深さ方向の連結に変換するかという点である。今までの ACTIVIEW では、いくつかの発見的手法を適用し横幅に収まることだけを条件としたもので、十分な最適化が実現されていなかった。本稿では、表示画面の横幅に収まることも条件として持つが、それ以外の条件についても視野に入れて、表示画面に最適なレイアウト変換を目指し、その方法を述べる。

3.1 最適化の基準

最適なレイアウトとは、「見やすさ」、「領域の有効利用」など様々な角度から考えることができる。表を構成する属性の数から考えられる全てのレイアウトパターンの内で最適なレイアウトを求めるには、いくつかの制約を考え、それらを満たしたレイアウトの内で、どのレイアウトを最適とするかを考える必要がある。ACTIVIEW では、最適なレイアウトの指標として以下の 4 つを用いる。

1 表示幅

2 画面充填率

3 意味反映

4 見やすさ

1 の表示幅に関しては、章の始めでも述べたように表示画面の横幅に収まることを絶対条件としている。

2 の画面充填率に関しては、表示画面の横幅に収まる範囲内で最も領域を有効利用しているレイアウトを求めるためである。

3 の意味反映に関しては、システムによる自動レイアウト変換によってデータを表示する側、つまりクエリーを作成するプログラマーの意図に反したものにならないためである。

4 の見やすさに関しては、利用者依存の部分が大きいと考えられる。その他にもデータを表として表示する際特有の見やすい条件があると考える。

章の始めでも述べたが、ACTIVIEW では、携帯電話端末のような非常に小さな画面に対して縦連結ではなくハイパーリンク連結を利用する。縦に長すぎる場合にはハイパーリンクを導入するなど「ハイパーリンク基準」が別途必要と考える。

3.2 ACTIVIEW で考える最適化とは

前節 3.1 で述べた 4 つの指標を ACTIVIEW でどのように取り入れるかについて述べる。まず、意味反映にあたるプログラマーが与えるクエリーにどこまでレイアウトに関する制約を設けるかが問題となる。属性を表示する順序やグルーピングまでをもシステムが決めてしまっては、意味的にプログラマーが意図していないレイアウトの表ができあがってしまう可能性がある。したがって、まずプログラマーがクエリーを作成する段階で以下の 3 つの事項をレイアウトに盛り込む。

- 属性の順序
- ネスト
- グルーピング

ここで言うネストとは、図 1 の例に示したように [ ] を用いて、同じ属性値をもつタプルをひとまとめにすることである。またグルーピングとは、中括弧で属性を括ることでレイアウト上の結合の優先度を示すものである。例えば、{ 大学名, 研究室 }, { 名前 , 学年 } とクエリーに記述した場合、“研究室” に関して” 名前” との間の連結よりも” 大学名” との連結を優先してほしいというプログラマーの意図が反映している。これらを「プログラマー制約」とする。

次に、表の最大幅が表示画面の横幅に収まるという制約がある。これを「幅制約」とする。「プログラマー制約」を満たす全レイアウトの中から「幅制約」を満たすレイアウトを求める。「プログラマー制約」と「幅制約」を満たしたレイアウトを評価関数によってランキングし、最も良いとするレイアウトを適用する。ここでランキングを行う際の基準として横幅が表示画面に収まる範囲で最大幅になるレイアウトを最適とする方法では候補が多くなってしまうので、今回は指標の 1 つである画面充填率を基準とした評価関数 (3.4 章) によるランキング方法を用いて最適なレイアウトを求める方法を示す。つまり、表示画面の横幅に収まることは絶対条件として、図 3 のように { 表示画面の横幅 (WD) × 表の高さ (H)} の内に実質データが表示に要する面積 (U) が占める割合を充填率とし、充填率が最も高いレイアウトを最適とする。

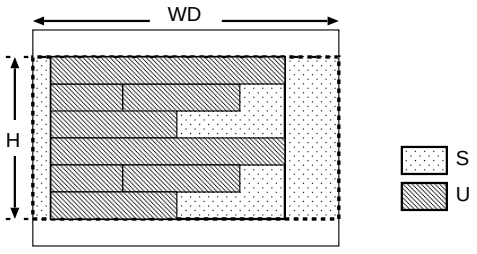

図 3: 充填率

3.3 レイアウトから表幅の計算方法

表示画面の横幅に収まるか否かを調べるために、レイアウト式から生成される表の横幅がどのぐらいになるのかを求める必要がある。ここではレイアウト式から生成される表幅の計算方法について述べる。ここで述べる方法を用いて、候補となる全レイアウトパターンの表幅を計算し 3.4 節で説明する評価関数を用いて値が高いレイアウトが適用されることになる。

まず、1 つの属性値を表示するのに必要な横幅を l_n と記す。n 個の属性値を持つ T_n の表幅 $L(T_n)$ は、横連結と縦連結の場合で以下のようになる。

$$L(C1 \ : \ T_n) \ = \ \sum_{k=1}^{n} l_n$$

$$L(C2 \ : \ T_n) \ = \ \max_{k=1 \rightarrow n} \ l_n$$

3.4 評価関数の導入

候補となる全レイアウト変換パターンの中で、表示画面サイズに最適なレイアウトを選び出すため評価関数を用いる。今回は、表示画面の横幅に必ず収まる範囲内で画面充填率を基準に持つ評価関数を定義する。ここで le はレイアウト式を表している。表示画面の横幅を WD、生成する表の高さを H、各属性を表示するのに必要な面積の全属性についての和を U とすると、評価関数 $f(le)$ は以下のようになる。

$$f(le) \ = \ \begin{cases} 0, & L(le) \ > \ WD \\ \frac{U}{WD \times H}, & L(le) \ \leq \ WD \end{cases}$$

3.6 節で説明する全レイアウト変換候補の内で評価関数 $f(le)$ の値が最高値になるものを最適なレイアウト変換とする。

3.5 見やすさ

今回は、評価関数の基準として画面充填率だけを取り入れたが、最後に見やすさをどのように取り入れるかについて述べる。見やすさは利用者依存の部分が大きい。1 つの要因として利用者によって重要だと考えるデータが異なることが挙げられる。閲

覧環境のサイズだけでは無く、利用者がプライオリティを置くデータの情報を得ることで、1 つの評価基準となる「ユーザ制約」が導入できる。この制約をレイアウトにどのように反映するかが次なる問題となるが、特に携帯電話端末などで用いる横連結から深さ方向連結への変換の際は、ハイパーリンクを貼る情報がインデックスとなるため、「ユーザ制約」が見やすさに大きな影響を与えると考える。また、図 4 のように利用者が" 名前" の情報にプライオリティーを置いた場合、画面充填率によらず (a) と比較して (b) のレイアウトを好むと考える。

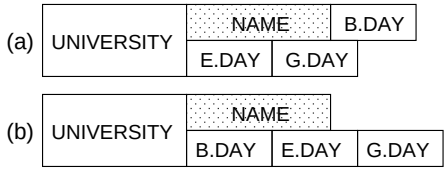

図 4: ユーザ制約の例

「ユーザ制約」とは別に見やすさに影響を与える要因として SuperSQL 特有の verb 関数がある。データベースから得たデータだけではなく、verb 関数を用いると引数の文字列を表に埋め込むことを可能とする。図 5 のように verb 関数によるデータは関係するデータとの位置関係が重要であり (a) と比較して (b) の方が見やすいと考える。

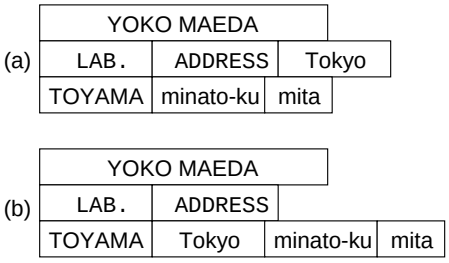

図 5: verb 関数の影響

3.6 レイアウト変換

「プログラマー制約」を取り込んだクエリーから生成する横連結だけからなるレイアウトからの縦連結を含むレイアウトへの全変換パターンを求める方法を説明する。全ての変換パターンについて 3.3 節で述べる計算を行い、評価関数の値が最高値のものを最適なレイアウト変換とし適用する。

3.6.1 基本変換

1 タプルが n 個の属性値からなる場合、属性値の集合を以下のように記す。ここで、1〜n の順序は「プログラマー制約」のため崩すことは行わない。

$$Tn \ = \ \{1, 2, ..., n\}$$

T_n の全ての要素が横連結した状態を以下のようなレイアウト式 le であらわす。この状態はレイアウト変換前の基本的な状態である。

$$le \ = \ (C1 \ 1 \ 2 \ ... \ n) \ = (C1 \ : \ T_n \ )$$

1	2	...	n

基本変換では、T_n を部分集合である $T_k(T_k \ \subset \ T_n)$ と $T_n - T_k$ に分解し 2 つの部分集合を縦連結する。

$$(C1 : T_n) \rightarrow (C2 \ (C1 : Tk) \ (C1 : Tn - Tk) \ )$$

ただし、

k=1 (T_k の要素が t のみであるとき)

$(C1 : T_n) \rightarrow (C2 \ \ t \ (C1 : T_n - T_k))$

基本変換は、$k \ \leq \ 2$ になるまで繰り返し適用することで、新しいレイアウトを生成する。図 6 に基本変換を木構造表現で記す。図 6 において、変換後の点線で囲まれた横連結について、可能であれば基本変換を繰り返す。また、T_n の全変換パターン数を $W(n)$ とする。

$$W(n) \ = \ 1 \ + \ \sum_{k=1}^{n-1} \{W(k)W(n-k) - 1\}$$

n=1,2 $W(n) = 1$

$n = 4$ の場合の例を図 7 に示す。

3.6.2 ネスト、グルーピングが存在する場合

ネストやグルーピングがプログラマー制約としてクエリーに指定してある場合、ネスト化した属性値

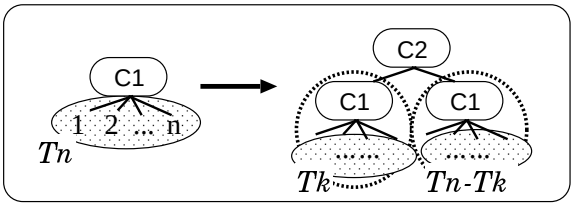

図 6: 基本変換

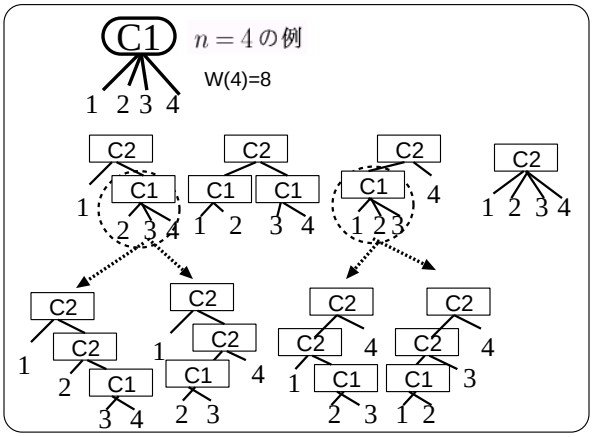

図 7: n=4 の場合の基本変換

の集合もグルーピングした属性値の集合も同様に、レイアウト変換を行った際に離れて表示されることがあってはいけない。そこで、グルーピングした属性値やネスト化した属性値は集合として 1 つの要素と見なして基本変換を適用する。つまり、T_n に含まれる属性値の内の m 個がグルーピングされているとき、T_n は m 個の要素を持つ代わりに m 個の要素からなる属性集合 $T_m(T_m \subset T_n)$ をひとつの要素として持つことになる。このような複雑な構造を持つ場合、全変換パターンはグルーピングの数とそれぞれグルーピングされた属性集合の構造の複雑さが影響してくる。したがって、T_n の要 素として k 個の属性値集合が存在し、それぞれの集合の全変換パターンを $R_j(1 \leq j \leq k)$ とする。T_n に含まれる属性値の内で k 個の属性値集合に含まれない属性値が m 個存在するとき、全変換パターンは以下のようになる。

$$\sum P((Wk+m), R_1, ..., R_k)$$

レイアウト探索はトップダウンに行うとする。図 8 に示すようにあらかじめ構造が与えられた場合、レベル 1 から横連結、縦連結の場合を求める。レベル 1 の変換後に求められたレイアウトそれぞれに関し、レベル 2 に関して同様に再帰的に変換を行ってゆく。

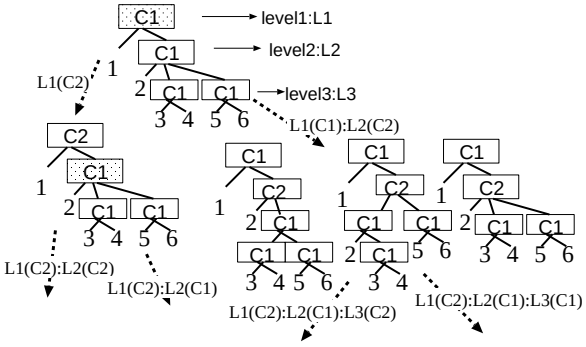

図 8: 変換パターン

3.7 探索空間の縮小

プログラマー制約を与えた上でも、全変換パターンを求めるのには膨大な計算量が必要となり多大なコストがかかる。したがって、どれだけ探索空間を縮小できるかが重要な問題となる。

1 つは「幅制約」を探索する過程に適用する方法である。前節 3.6 に述べた方法には木構造を変換する過程の内で、幅制約を絶対に満たすことの無いパターンが含まれており、そのようなパターンについて変換を繰り返すだけ無駄である。例えば、図 9 の例を用いて説明する。それぞれの属性に割り当てられた大きさと表時画面の横幅から、レベル 1 に横連結を選んだ時点で幅制約を満たさないレイアウトになってしまう。したがって、レベル 1 は縦連結であることが 決定できるので、横連結から派生する変換パターンは求めない、つまり枝刈りの手法を取り入れることができる。

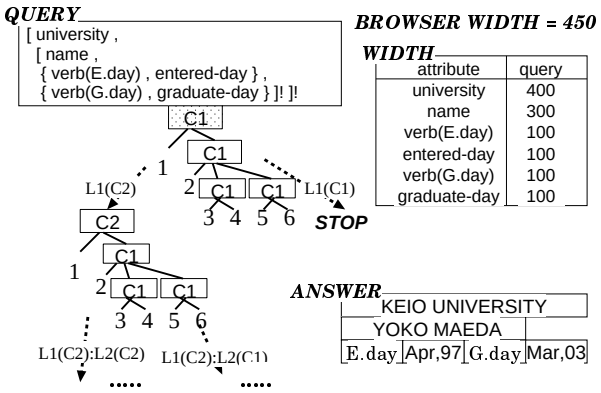

図 9: 幅制約による探索空間の縮小例

4 結論と課題

本論文では、ACTIVIEW で行われるレイアウト変換について、最適なレイアウトを実現することを目指し、その方法について述べた。具体的には、プログラマーの意図を反映したクエリーから考えられるレイアウトパターンを全数探索し、今回は、画面充填率を基準とした評価関数を定義、導入した結果を最適なレイアウトとした。

4.1 考察

大きく以下の 2 つに議論の余地が残ると考える。

- 最適なレイアウトの基準
- 変換パターンの探索方法

今回は最適なレイアウトの基準として、「表示幅」「画面充填率」「意味反映」「見やすさ」の 4 つを挙げたが、これで十分であるのか、それぞれは十分に取り入れられているのかが重要となってくる。したがって、それぞれの基準を用いた場合、用いない場合の比較を行うことで、妥当性を検証する必要があると考える。特に「見やすさ」に関しては、3.5 章で 2 つの例を挙げたが、他にも影響する要因は考えられる上にどのように取り入れるかが問題となる。他に影響を与える要因としては、ネストである。ネストは同じ属性値を持つタプルをひとくくりにするので、共通の属性値と、それぞれのタプルの間には見やすさに影響する関係があると考える。

変換パターンの探索過程に、データを表で提示する際特有の発見的ルールを用いることは有効であると考える。しかし、発見的方法では必ずしも最適解を求めてくれる保証が無くなってしまうので、どのような発見的ルールを用いれば、最適解を求める確率が高いのかを検証した上で導入する必要があると考える。

4.2 課題

今回は評価関数の基準として、画面充填率を取り入れたに過ぎない。特に、3.1 章で示した、最適なレイアウトの 4 つの指標の内、「見やすさ」は取り入れていない。どういったレイアウトを” 見やすい” とするのか、またどのように取り入れるのかが今後の課題の 1 つである。また、ハイパーリンク導入時の基準に関しても課題が残る。

次に大きな課題となるのが、計算量の削減である。今回は「幅制約」による探索空間の縮小について 3.7 章に述べた。この方法は、表示幅が極端に大きな属性が存在するときに非常に有益ではあるが、全ての属性の表示幅に大差が無い場合、ネストやグルーピングがされていない場合、属性が多い場合には効果を持たない。したがって、他の方法が必要となる。今後の課題を以下にまとめる。

- 「見やすさ」に関する基準を評価関数に導入
- ハイパーリンク基準の導入
- 変換パターン探索の際のコストダウン

参考文献

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

[2] Motomichi Toyama, “SupreSQL: An Extended SQL for Database Publishing and Presentation,” *ACM SIGMOD '98* , pp. 584-586, 1998

[3] Yoko Maeda, Motomich Toyama, “ACTIVIEW: Adaptive data presentation using SuperSQL,” *27th VLDB*, 2001

[4] 前田 葉子, 遠山 元道, “ACTIVIEW:SuperSQL を利用した適応型表示ビューの実現” 電子情報通信学会データ工学ワークショップ,2001