SuperSQLを用いた組版におけるレイアウト自動修正

亀岡慎平[†] 遠山元道[††]

SuperSQLを用いたデータベース出版システムは，レイアウト指定演算子の組合せによって出力結果を構造化し，多様なレイアウト表現を簡単に実現可能であるというSuperSQLの特徴を利用したシステムであるが，クエリによってレイアウトを指定するため，出力結果が紙面内に収まりきっているかどうかは事前には分からない．そこで本研究では，指定されたレイアウトを極力崩さずに紙面内に収まるようにレイアウトを自動変換する手法を提案する．実験の結果，手作業でレイアウトを修正して紙面に情報を収めるのに比べて平均で10倍以上時間を短縮できること，そして手作業で修正したクエリとシステムが修正したクエリとを比較し，その相違箇所が平均して1人あたり1カ所程度であることを示した．さらに実験で使用した例に関して実際にシステムを実行してもらった結果，被験者の約8割が本システムを利用したいと答えたアンケートによって，本システムの有用性を示した．

Automatic Layout Modification on Publishing System Using SuperSQL

Shinpei Kameoka[†] and Motomichi Toyama[††]

Our database publishing system using SuperSQL structures the output by combining the layout specifying operators which as a result allows various layout representations. However, it is difficult to predict whether or not the output generated fits in the printed paper. In this paper, we propose a technique which efficiently rearranges the layout so that the information fits nicely in printed paper with no drastic changes in their layout. The experimental result indicates that the execution cost has been shortened more than 10 times a person average, and the gap between query modifying manually and automatically is one place a person average. Result of executing system by the subjects, about 80 percent subjects answered that the system giving satisfactory results compared to when it is done manually.

1. はじめに

近年，インターネットや携帯電話，デジタルカメラに代表されるように世界規模の急激なIT化・デジタル化が進み，個人がデジタルデータを扱う機会が急増した．そうしたことからデジタルデータを管理するデータベースが個人にとって身近な存在になってきたといえる．個人がデータベースを利用する際，検索情報を端末の画面上だけでなく紙媒体に高品位で効率的に出力したいという需要が存在する．

しかしながらここで問題となるのが，データベースから検索される情報がどの程度の量になるかということは一般的に予測できないのに対し，出力対象である紙媒体は出力領域が用紙の規格によって制限されているということである．そのためこれまでデータベース内の情報を紙媒体に高品位に出力するためには，多くの場合情報をDTPなどを用いてレイアウトする必要があった．表示されている情報を印刷したいというような場合，これでは効率が悪い．さらにデータベースが更新されるとレイアウト指定の細かな修正が必要であった．

この問題を解決するために，亀岡らはSuperSQLを用いたデータベース出版システムを開発・実装した[15]．SuperSQLには個人ユーザがレイアウトを見ながらクエリを意識することなく，直感的に実行できるGUIの研究もされており，DTPなどを使うことに比べ簡単に扱うことができる．このSuperSQLを用いたデータベース出版システムは，レイアウト指定演算子の組合せによって出力結果を構造化し，多様なレイアウト表現を簡単に実現可能であるというSuperSQLの特徴を利用したシステムであるが，クエリによってレイアウトを指定するため，出力結果が紙面内に収まりきっているかどうかは事前には分からないという問題点があった．

本研究ではSuperSQLの，クエリ中のレイアウト指

† 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University
†† 慶應義塾大学理工学部
Faculty of Science and Technology, Keio University

定演算子を少し変えるだけで，レイアウトが大きく変化するという特徴を利用し，SuperSQL を用いたデータベース出版システムにおいて出力結果をユーザの指定したレイアウトになるべく近い形で紙面内に収まるようにレイアウト演算子，また文字の大きさやセルの大きさを指定する装飾子を自動的に修正するシステムを提案し，実装した．

2. 関連研究

本研究はデータベースからの検索結果を紙面という制限された領域に収まるように，オリジナルのレイアウト指定に近い形式で自動変換するシステムの提案を目的としており，関連研究としては制限された表示領域における適切な情報提示があげられる．

現在のところ，携帯電話で満足に閲覧することができるのは，携帯電話用につくられたサイズの小さなページがほとんどである．また PDA であっても閲覧には頻繁にスクロール操作が必要となり，一度にページの一部分しか見ることができない．代表的な対応策は，携帯電話のようにページを別に用意しておくか，スタイルシートを何通りか定義することで 1 つの HTML ソースをいくつかのデバイスに対応させることなどがあげられる．しかし，これらの対応策はいずれもあらかじめコンテンツ提供者側で定義が必要である．そうしたなか，アクセス時にソースを解析し各デバイスに適した形式で情報を提示する手法を Automatic re-authoring という．この Automatic re-authoring に関する研究をいくつか紹介する．

Bickmore らは Web コンテンツを省略・小型化してユーザに提示する Digestor システムを提案した[1),2)]．Digestor システムではソースを解析し，セクションヘッダをリンクアンカとして，その内容をリンク先に移動する Outlining，フォントの縮小，そして画像の縮小または削除といった処理を行う．画像を削除する場合は `<alt>` タグ内のテキストや画像があったことを示すアイコンで置換し，縮小・削除どちらの場合も元画像を取得できる．Outlining において `<H1>` タグなどによって指定されたセクションヘッダがない場合には初めのセンテンスを抽出しリンクアンカとすることで対応する．そして発見的手法によって求めた優先順位に基づいてこれらの処理を適用しページを省略・小型化することで，オリジナルのイメージを壊すことなくユーザに提供する．

Buyukkokten らは文章の要約を行いテキストのみでユーザの要求した情報を提示する Power Browser システムを提案した[4)]．このシステムでは情報をよりコンパクトに提供するために，ユーザの必要としない情報はなるべく表示せず，ユーザの必要に応じて簡単な操作でより多くの情報を表示することができる Accordion Summarization という手法を提案している[3)]．文章の要約には単語の重要度を用いる．ここで単語の重要度とは，どれだけその単語がそのページ独特のものであるかを示し，その単語がそのページ内でどれだけ出現頻度が高いか，また Web 全体でどれだけ出現頻度が低いかによって決まる．テキストのみに要約することでページのデザインは基本的に失われてしまうがその分サイズもコンパクトになり，移動端末の通信速度でも快適に閲覧することができる．

Chen らは Web ページ内の情報を省略・要約することなく，細かい領域に色分けされたページ全体のサムネイルと各領域に対応する切り分けられた sub-page の 2 レベルでユーザに提示する Two Level Summarization という手法を提案した[5)]．ユーザは表示されるサムネイルの一部をクリックすることでそのブロックの対応する sub-page を閲覧することができる．この手法は大きく分けてページの構造解析とページの分割という 2 つのプロセスからなっている．ページの構造解析では，まず HTML ソースのタグを解析し DOM Tree を生成する．生成した DOM Tree を走査して位置情報と height/width の比からページを大きく分割し，さらにそのブロックを `<HR><DIV><TABLE><TR><TD>` などのタグに基づき分割する．その後，各ブロックを垂直・水平軸に投影し，一番 gap の大きい箇所でさらに分割する．最後にユーザに提示するページ全体のサムネイルを分割したブロックごとに色分けする．構造解析によって切り分けられた各ブロックに対応するページを実際に生成する際には様々な問題が生じる．たとえば装飾やページ内リンクの情報が失われてしまう問題がある．これを解決するために，ソースのより上位で定義された装飾情報はたどって取得し，スタイルシートで定義された情報は `<HEADER>` 内の情報をすべての sub-page にコピーすることで対応する．ページ内リンクもアンカのパスを書き換える．こうして作成した各 sub-page は，オリジナルの一部をほぼ完璧に再現することができる．

増田らは HTML の表形式データに着目し，携帯端末向けに表を項目名と項目データの組として変換・表示するシステムを提案している[14)]．このシステムは言語的性質を用いて表の各セルデータに対するベクトルを用意し，セル間の類似度をベクトル空間法によって算出する．算出したセルの類似度が低くなる部分には内容的な切れ目があると判断することで，表の項目名

と項目データを判別し，表構造を項目名に対して項目データを列挙する形式に変換しユーザに提示する．

しかし，ここで紹介した研究は縮小や要約，リンクを用いるなど様々な手法で制限された領域に情報を収めているものの，レイアウト構造自体はまったく変化させていない．増田らは表構造を分解しているが，それに特化した変換である．これに対し本研究では縮小を行いつつ，それでも収まりきらない場合はレイアウト構造自体を大きく変化させることが可能であるため，今までの手法では収めることができなかったような場合にも対応できる．2001 年には先行研究として，Maeda らが SuperSQ☆を用いて，クエリ中のレイアウト指定演算子を変更することでテーブル構造を変換し，ユーザに提示する actiview システムを提案している[7]．このシステムはテーブルの横幅が端末の画面に収まるように，レイアウトを自動変換してユーザに提供する．本研究ではより制限された紙媒体という領域が対象であるため，反復要素を折り畳む指定を可能にした．そして actiview ではクエリはシステムから発行され，それをウィンドウサイズに合わせ変更するが，本システムではユーザがクエリを発行する．本論文ではユーザが意図する様々なレイアウトに近い状態で変換をするために，レイアウト構造の内側を変換を目指すこと，変換に対する柔軟性の概念などを導入したレイアウトの自動変換手法を提案する．

3. データベース出版

出版物においてレイアウトを定型的にできる分野として，情報誌や多ページカタログ類があるが，こういった印刷物は，コンピュータのデータベースで管理されているデータをレイアウトして原稿にすることが多い．そこでデータベースからの検索結果を，そのまま出版物，プレゼンテーションのような最終生成物として出力する方法が研究・開発されており，これをデータベース出版という[9]．

またデータベースによるデータの一元管理により，印刷・デザインなどの制作の現場だけでなく，企画や販売部門でも，これらのビジュアルデータを有効活用するといったことも可能となる．画像をふんだんに使った企画書やデジタルプレゼンテーションの作成，マーケティング情報を制作物の構成に生かすといった，多目的利用の可能性もある．

3.1 データベース出版アプリケーション

現在広く使われているデータベース出版アプリケーションには以下のようなものがある．

- DBPublisher（リンクス社）
- WAVE2002（シンプルプロダクツ）
- DBPallet（Rightvision 社）

図 1 出力出版物の例（DBPublisher）
Fig. 1 Example of database publishing (DBPublisher).

主流のデータベース出版アプリケーションは，まずデザインや印刷に必要とされる大量の文字や画像のデータをデータベースで一括管理し，その中から制作に必要なものを検索によって絞り込み，自動的に出力，組版を行うといったものである．**図 1** にこのようなアプリケーションから生成された出版物の例を示す．

データベース出版の導入で，モニタの中での手作業に代わり，バッチ処理による自動組版を行うことにより作業時間の短縮，コストの削減を実現する．さらに出力結果からデータベースを更新することができるといったデータベースと連動した機能を持ったものも多い．ただしこれらアプリケーションの多くは QuarkXPress などのレイアウトソフトと連携させるため，複雑なレイアウトには結局手動組版が必要となる．

3.2 レポートライタ（Microsoft Access）

レポートライタとはデータベース出版アプリケーションの 1 つの種類で，データベースからの検索結果を簡単に見やすくまとめて紙媒体に印刷するものである．一般にこのようなアプリケーションから印刷されるものは，検索結果を人が見て分かりやすい状態で紙面上に簡易的にレイアウトするもので，広く流通する情報誌などの出版物ほどの質を目標としていない．逆に，その分操作に特別なスキルはさほど必要ないという利点があるといえる．本研究のベースとなるシステムはデータベースに SuperSQL クエリを与えるだけで，データベースからの検索結果を紙面上にレイアウトするものであるため，基本的にこのアプリケーションに分類される．

一般に広く使用されている Microsoft 社のデータベース管理ソフトウェア Access にもレポートライタの機能がある．**図 2** にその出力例を示す．

☆ 詳しくは 4.1 節参照．

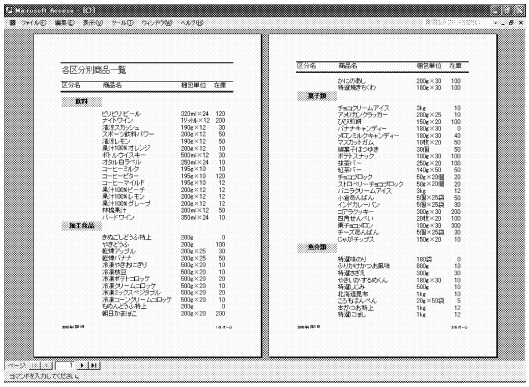


図 2 Microsoft Access のレポート出力
Fig. 2 Report writer of Microsoft Access.

3.3 本研究の位置付け

このように既存のデータベース出版アプリケーションは操作性と品質のどちらかを犠牲としていた．本研究は，SuperSQL の属性とレイアウト指定演算子を組み合わせることで “検索された情報の意味を階層的な構造で表現する” というデータベース出版において有効な特徴を活用し，出力レイアウトが紙面に収まりきらない場合にも，既存のデータベース出版アプリケーションのようにユーザによるレイアウトの細かな修正なしに紙面に収まるようなレイアウトに自動変換し，かつ一般的なレポートライタよりも多様なレイアウトと装飾指定が可能で，実用に耐えうる品質の出版物を生成することを目標とした．

4. SuperSQL とデータベース出版

4.1 SuperSQL とは

SuperSQL は SQL を拡張したワンソースマルチユースを実現するクエリ言語である[11),12)]．その質問文は SQL の SELECT 句を GENERATE <*medium*><*TFE*> の構文を持つ GENERATE 句で置き換えたものである．ここで <*medium*> は出力媒体を示し，HTML，XML，PDF，LaTeX などの指定ができる．また <*TFE*> は SQL における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式である[10)]．r_i を関係，P は SQL における条件とすると，SuperSQL クエリは以下のように表せる．

GENERATE <*medium*><*TFE*>
FROM $r_1, r_2, \ldots, r_n$
WHERE P

TFE（Target Form Expression）は，SQL では SELECT 句に記述する属性名を，レイアウト演算子である結合子，反復子と組み合わせることで出力結果の構造を指定する式である．さらに@で表す装飾子によって表のセルの幅や背景色などの詳細なカスタマイズが可能である．

表 1 結合子の種類と意味
Table 1 Connectors.

結合子	意味	クエリ表記	略記法
,	横結合	A , B	C1
!	縦結合	A ! B	C2
%	深度結合	A % B	C3

表 2 反復子の種類と意味
Table 2 Groupers.

反復子	意味	クエリ表記	略記法
[ ],	横反復	[A],	G1
[ ]!	縦反復	[A]!	G2
[ ]%	深度反復	[A]%	G3

4.1.1 結 合 子

結合子（Connector）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 2 項演算子である．横方向を 1 次元，縦方向を 2 次元，深度方向を 3 次元とし，内部表現においてはそれぞれ Connector の “C” と合わせて C1，C2 のように略記する．**表 1** にその種類と意味，略記法を示す．なお，表中の A と B は属性に限らずどんな TFE であってもかまわない．ここで深度結合とは，たとえば HTML 出力ではハイパーリンクによる結合を表す．

4.1.2 反 復 子

反復子（Grouper）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単項演算子である．また反復子に関しても，Grouper の “G” と各次元方向を合わせて同様に略記する．**表 2** に反復子の種類と意味，そして略記法を示す．結合子と同様，表中の A はどんな TFE であってもかまわない．たとえば，

[学籍番号 , 評点]!

と記述することで，横方向に連結された学籍番号とその生徒の評点をインスタンス数だけ縦方向に反復して出力する．

また反復子は単に構造を指定するだけでなく，反復したデータとそれをグルーピングするデータのネストの関係によって属性間の関連を指定することができる．たとえば

[科目名]! , [学籍番号]! , [評点]!

とすると，単に各々の一覧が表示されるだけで互いの関連は失われるが，

[科目名 ! [学籍番号 , 評点]!]!

と反復子を入れ子状にすることで，その科目における

学生の評点一覧が表示される．

4.1.3 装 飾 子

SuperSQL では関係データベースにより抽出された情報に，文字サイズ，文字列の出力領域（セル）の横幅，セル内での文字列の位置などの情報を付加できる．これらは装飾演算子@{ 装飾指定式 } によって指定することができる．装飾指定式は (項目名 = 値) として指定する．複数指定するときは各々を “,” で区切る．たとえば，科目名の背景を赤にして，セル内の中央に文字列を配置したい場合は以下のように記述する．

[科目名@{bgcolor=red, align=center}! [学籍番号 , 評点]!]!

装飾子はこのように属性に指定するだけでなく，反復子に対して指定するものもある．

4.1.4 imagefile 関数

SuperSQL における関数は，データベース検索結果の文字列に対し，特定の処理を行うための機能である．imagefile 関数とは，画像のファイル名が格納された属性とそのファイルまでのパスを指定し画像を出力するもので，以下のように記述する．

imagefile(属性， path=“パス”)

ここでパスは絶対パスでも相対パスでもかまわない．

4.2 SuperSQL 処理系のシステム概要

SuperSQL 処理系は，構文解析部（Parser），リスト構造生成部（List Constructor），メディア生成部（Code Generators）からなる（**図 3**）．SuperSQL クエリが発行されると，まず構文解析部で通常の SQL 文とレイアウト式に分け，SQL 文を DBMS に渡して検索結果を受け取る．リスト構造生成部では受け取ったフラットな結果に対し，レイアウト式に従って階層的な構造を持たせる．最後にメディア生成部がクエリで指定したメディアのソースを結果として出力する．

4.3 PDF の出力

システムではリスト構造生成部において取得した構造化された検索結果の文字列をもとに，直接 PDF のソースコードを書き込むのではなく，ドイツ PDFlib GmbH の提供するクラスライブラリ “PDFlib” を用いて間接的にコードを書き込む[6),8),13)]．このクラスライブラリ内の様々なメソッドを使うことで，PDF ファイルを比較的容易に作成することが可能である．

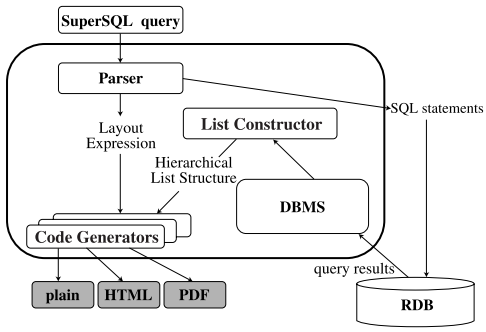


図 3 SuperSQL 処理系のシステム概要
Fig. 3 Overview of SuperSQL system.

5. 表構造の実現

5.1 反復される出力を折り返す装飾子 “fold”

PDF のように出力領域が限られているメディアの場合，出力結果が紙面からはみ出してしまいすべての結果を見ることができないことがある．そのため，構造的な意味をできる限り保ったまま，紙面に収まるように構造を組み換える工夫が必要となる．そこで本システムでは装飾子 “fold” によって，反復子に対してある回数連続して出力した後に別の方向（縦に対しては横，横に対しては縦）に 1 つずらして出力する．たとえば，

[学籍番号 , 評点]!@{ fold=3 }

とすると，学籍番号と評点の組が縦に 3 回繰り返して出力された後，横に 1 つずらしてまた上から 3 回繰り返して出力していくことになる．この装飾子 fold はこのような回数指定のほかに長さでも指定可能である．以下に示すように装飾子の付く反復子の要素は単なる属性だけでなくどのような TFE であっても，また折り畳み指定をした反復子がレイアウト演算子の項となっていてもかまわない．

[科目名 [学籍番号 , 評点]!]!@{ fold=3 }
[科目名 [学籍番号 , 評点]!@{ fold=3 }]!

5.2 ラベル付けアルゴリズム

本システムでは長方形を隙間なく並べることで表構造を実現する．このときサイズを揃えるべき長方形を判別するために各要素，そしてレイアウト指定演算子に対してラベルを付与する．各セルには，幅に関するサイズ調節を行うものと，高さに関する調節を行うものの 2 種類のラベルを付与し，同じラベルが付与された要素間で幅（高さ）を揃えることで表構造を生成する．

ラベルはクエリの TFE を木構造にしたときに，深さ優先探索を基本として各要素に順に付ける．以下の TFE に対する，幅を調節するラベル付けの例を**図 4** に示す．

[A ! { B , [C]!@{fold=xxx} , D }
! [E],@{fold=yyy}]!

ここで木構造中の TFE の表記法は表 1・表 2 の略記法に従う．本研究のラベル付けアルゴリズムにおいて，

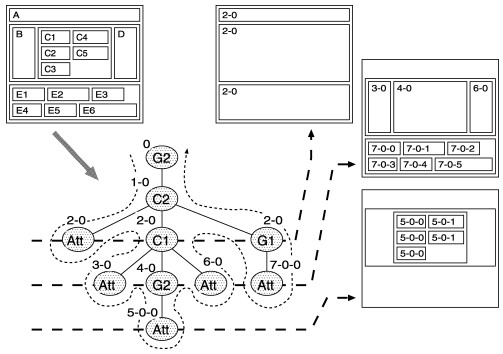


図 4 ラベル付け例（幅）
Fig. 4 Example of labeling (width).

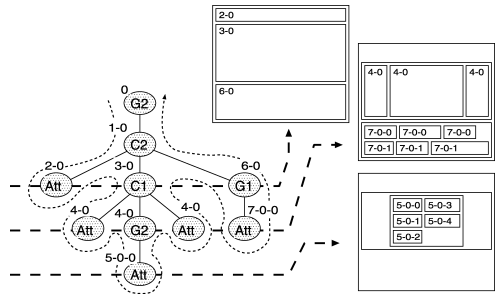


図 5 ラベル付け例（高さ）
Fig. 5 Example of labeling (height).

単純な深さ優先探索と異なる点は以下の 4 点である．

- 縦結合子（C2）のすべての子要素には同じラベルを付与する．これは縦に並べる要素は幅を揃える必要があるからである．
- 反復子の子孫ノードすべてに Suffix を 1 つ追加する．つまり，各ノードには木構造を root までさかのぼったときに通る反復子の数だけ Suffix が付与される．
- 縦反復子（G2）のインスタンスには同じラベルを付与し，折り畳むたびに Suffix をカウントアップする．これは列ごとに幅を揃える必要があるからである．
- 横反復子（G1）のインスタンスに関しては，たとえ折り畳んだとしてもインスタンスごとに Suffix をカウントアップし，幅は揃えない．

もし図 4 で root ノードの縦反復子が折り畳まれた場合，2 列目のインスタンスでは root 以下の全ノードの 1 つ目の Suffix が 1 となり，以降 3 列目 4 列目となるごとにカウントアップされていく．

高さを調節するラベルに関しても，縦と横の概念を入れ替えてこれと同様に各要素に付与する（**図 5**）．

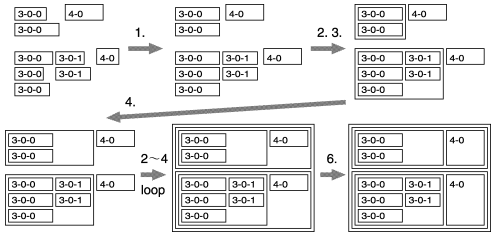


図 6 セルのサイズ調節アルゴリズム（ラベルは幅調節）
Fig. 6 Cell size adjustment algorithm (width-label).

5.3 サイズ調節アルゴリズム

本節では 5.2 節のように付与したラベルを利用して，どのようにセルの大きさを調節していくかを述べる．属性，つまり TFE の木構造における leaf ノードは同じラベルを付与されたセル間で最大の幅（高さ）を持つものに合わせてサイズを調節する．そして，レイアウト演算子に対して出力するセルに関しては，ラベルと子ノードのサイズによってサイズを調節する．各レイアウト演算子は TFE の木構造において，必ず 1 つ以上の TFE を子ノードとして持ち，自身のノードのサイズは各インスタンスの子ノードすべてを囲う最小外接矩形（MBR）となる．そしてさらに視認性を高めるために MBR レベルでも同ラベル間でサイズ調節を行う．以下に本手法のサイズ調節アルゴリズムを示す（**図 6**）．

1. leaf ノード（属性）の各インスタンスのサイズを同ラベル間で最大のものに合わせて調節．
2. 子ノードがすべてサイズ調整済みのノードを探索．
3. 探索したノードと同じラベルの全インスタンスに関して，子ノードのサイズから MBR の幅（高さ）を計算．
4. 全インスタンスの MBR 中で最大の幅（高さ）に合わせてサイズを調節．
5. root ノードにたどり着くまで 2〜4 を繰り返し処理．
6. 各 MBR において余白を埋めるように子ノードのサイズを再調節．

付与したラベルに従い以上のようにセルのサイズを調節することで，表構造は適切なレイアウトで，コンパクトに情報を提示することができる（**図 7**）．

6. SuperSQL を用いた自動組版修正

本研究では，ユーザの指定したオリジナルのレイアウト指定に近い形で紙面内に情報を収める変換をレイアウトの最適化と定義する．

ユーザの指定に近づけるために，まずユーザが要素の幅を指定した場合はその値は変更しない．フォント

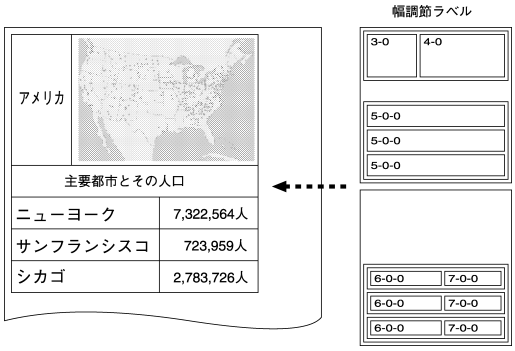


図 7 ラベル付けの結果
Fig. 7 Result of labeling.

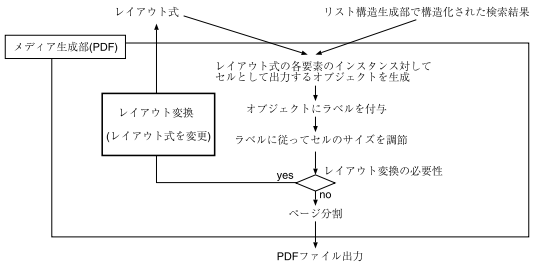


図 8 メディア生成部内の処理概要
Fig. 8 Overview of code generator.

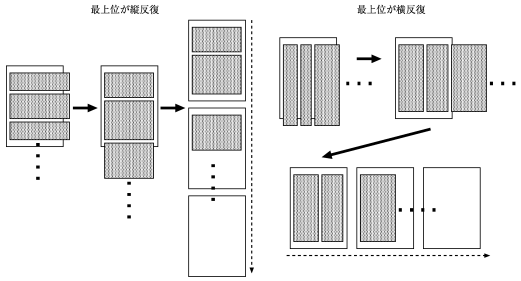


図 9 レイアウト最適化とページ分割
Fig. 9 Layout optimization and page division.

サイズを指定した場合も指定した要素に関しては，そのサイズを基点にフォントの縮小を行う．フォントの縮小のみによって，レイアウト構造を変えずに紙面内に情報が収まれば，それが最も近いレイアウト指定による最適化となる．しかしフォントの縮小だけでは紙面から大きくはみ出しているような場合に変換できない．ここで本システムにおいて，どのようなレイアウト変換を近いと考えるかを説明する．たとえば，一般的に新聞のテレビの番組表においてチャネル順に並んだ列の並び順の変更の方が各番組内での出演者の並び順の変更よりもレイアウトの変更が大きいと感じ，ポータルサイト YAHOO! において様々にカテゴリ分けされたリンク集に対して個人ツールやトピックスといったものが右側ではなく下側にレイアウトされるような変更の方が，各要素内でのハイパーリンクの並び順の変更よりもレイアウトの変更が大きいと感じる．このように，レイアウト階層のなるべく内側を変換することはオリジナルのレイアウトを保つ 1 つの指針となる．同様に，テレビの番組表をより小さなスペースに表現するような場合，各チャネル欄の幅の比を保持した方がレイアウトの変化は少なく感じる．本システムでは，レイアウトの変換を行わないフォントの縮小が最優先であること，なるべく内側を変換すること，そして変換時に横結合要素の幅の比を保持すること，これら 3 つの指針に従ってレイアウトの変換を行い最適化を目指す．

しかしながらユーザにとって，必ずしも内側を変化させることが近いレイアウト変換と感じられるかどうかは分からない．そこで本研究では変換に対する柔軟性の概念を提案し，システム実行の際に起きる各レイアウト変換に関して，内側の変換にこだわった結果縮小した方向に対してもう一方へ大きく伸びてしまうような場合，その変換は行わずに外側での変換を許容する．システム実行に際しては，この閾値と最小フォントサイズは必ず指定されなければならず，ユーザが明示的に指定しない場合デフォルト値が適用されることとなる．

メディア生成部における本システムの処理概要を**図 8** に示す．本システムでは生成した表構造が 1 ページに収まらない場合，レイアウトの最適化またはページの分割を行う．ページの分割は TFE の最上位の反復子の反復方向に対してのみ行い，反復方向ではない方の表構造の長さ（最上位が縦反復ならば幅）がページに収まらない場合はレイアウトの最適化を行い，その方向のサイズをページ内に収め，その後ページの分割を行う（**図 9**）．

6.1 レイアウト自動修正アルゴリズム

レイアウト変換アルゴリズムには最上位の反復子が縦反復で表構造の幅を縮小するものと，最上位の反復子が横反復で表構造の高さを縮小するものがあるが，本節では前者を例にとって説明する．後者のアルゴリズムはこれから説明するものの幅と高さの概念を入れ替えたものとなる．

最上位の反復子が縦反復で，生成された表がページの横幅に収まっていない場合，本システムでは TFE の木構造でなるべく深いレベルにある要素に変更を加えることで表構造の横幅をページに収まるように縮める．各ノードで減らさなければならない幅（長さ）を Δ_{excess} とすると，まず木構造の root ノードに紙面か

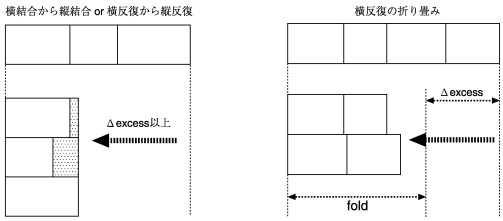


図 10 横結合・横反復の変換例
Fig. 10 Change example: C1 and G1.

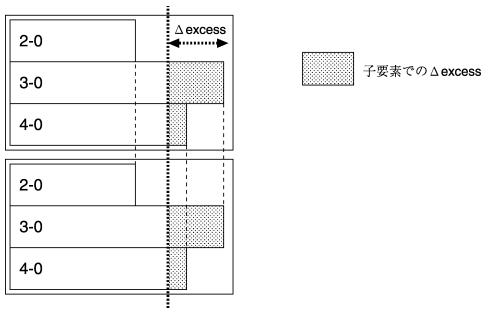


図 11 縦結合の変換に使用する幅調節ラベルの例
Fig. 11 Width adjustment label used to change of C2.

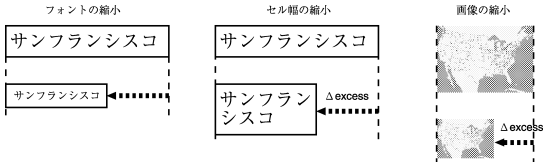


図 12 属性・画像の変換例
Fig. 12 Change example: attribute and image.

らはみ出した長さが Δ_{excess} として与えられ，子ノードにおいて Δ_{excess} 以上幅を縮めることができるかどうかを再帰的に leaf ノードまで調べる．子ノードにおいて Δ_{excess} 以上縮小することができない場合，また後述する変換に対する柔軟性の概念によってこれ以上内側の変換を行わないと判断した場合は自身のレイアウトを変換する．各ノードにおける変換アルゴリズムを順に述べる．

6.1.1 横結合・横反復での変換アルゴリズム

横方向にセルを連結するこれらの TFE では子要素での変換を試みるほかに，連結方向を縦に変更することなどでレイアウト幅を減らすことができる．以下にその優先順位を述べる．

- 横結合子（C1）の場合
  1. 各子要素の幅に比例して Δ_{excess} を分配し，子要素での変換を試みる．
  2. 自身を縦結合に変換する（**図 10**）．
- 横反復子（G1）の場合
  1. Δ_{excess} 減らした長さで折り畳み指定（図 10）．
  2. 自身を縦反復に変換する（図 10）．

以上の変換で Δ_{excess} 以上幅を縮小できない場合は親ノードへ変換を委託する．なお，横反復子での変換において子要素での変換がないのは，本変換アルゴリズムに従うと反復する各インスタンスがリストだった場合，それぞれインスタンスごとにレイアウト構造が違ってしまい視認性が著しく低下する可能性があるからである．

6.1.2 縦結合・縦反復での変換アルゴリズム

縦方向にセルを連結するこれらの TFE では子要素での変換を試みる以外にない．ただし縦結合では幅を揃えるラベルによってサイズが調節されてしまっているが，このサイズに基づいてすべての子要素に Δ_{excess} をそのまま渡す必要はない．なぜならば Δ_{excess} 減らさなければならないのは最大幅を持っている要素だけでその他の子要素は Δ_{excess} 以下の縮小，もしくはまったく変換する必要がない可能性もあるからである．そこで本システムでは，幅と高さを調節する 2 つのラベル以外にもう 1 つのラベルを用意する．このラベルは反復子で関係する Suffix 以外は深さ優先探索と同様に付与し，このラベルを利用してページ幅をオーバした場合の縦結合，そしてページの高さをオーバした場合の横結合における変換アルゴリズムで参照するサイズの調節をし，保持しておく（**図 11**）．

6.1.3 属性での変換アルゴリズム

属性，つまり leaf ノードではフォントサイズの縮小，またセル幅を縮小して文字列を折り返すことでレイアウト幅を縮小することができる．属性と同様 leaf ノードとなるイメージ関数によって出力された画像とともにその変換優先順位を以下に示す．

- 属性（Att）の場合
  1. フォントサイズの縮小（**図 12**）
     装飾子によってフォントサイズが指定されているときはそのサイズを基点とする．
  2. セル幅の縮小による文字列を折り返し（図 12）
     装飾子によってセル幅が指定されているときはこの変換は行わない．
- 画像の場合
  1. セル幅を縮めて画像を縮小（図 12）
     装飾子によってセル幅が指定されているときはこの変換は行わない．

以上の変換で Δ_{excess} 以上幅を縮小できない場合は親ノードへ変換を委託する．ここで属性の場合，フォントサイズの縮小とセル幅の縮小という 2 通りの変換があるが，1 つの要素に対してこれらを同時に行うことはできない．これは現在の仕様となっている．なお

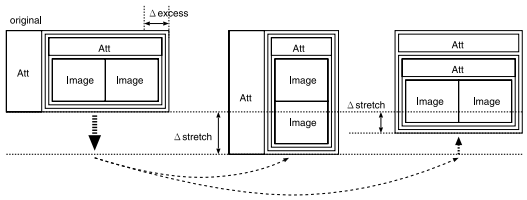


図 13 より内側の変換がより大きな縦の伸びにつながる例
Fig. 13 Internal change causes expansion of another direction.

前述のとおり，フォントサイズはあらかじめ許容する最小サイズを指定することが可能である．

6.2 柔軟性の概念

前述したように本システムのレイアウト最適化の大きな 1 つの指針は，レイアウト階層のなるべく内側を変換することであるが，内側のレイアウト変換を優先することが必ずしも見た目のレイアウト変更の最小化につながるとは限らない．幅を減少させるレイアウト変換において，より内側を変換することによって生じる縦方向への伸び $\Delta_{stretch}$ が，より外側を変換して生じる伸びよりも大きくなる場合には，外側優先の変換の方が見た目のレイアウト変更が小さいと考えられる（**図 13**）．

そこで本システムでは，各要素の変換に対する柔軟性という概念を導入する．各要素で減らさなければならない幅（高さ）Δ_{excess} と，ある変換を行ったときの Δ_{excess} に対する高さ（幅）の伸び $\Delta_{stretch}$ との比が閾値 θ_{flex} 以上の場合，その要素はその変換における柔軟性が低いと判断し，その変換を行わない．

$$if\ \theta_{flex} < \frac{\Delta_{stretch}}{\Delta_{excess}}\ then\ \text{その変換は行わない}$$

この閾値を用いることで，減らさなければならない Δ_{excess} に対し，他方向へ大きく変化する変換を抑制することができる．たとえば，図 13 において減らさなければならない $\Delta_{excess} = 20\,\mathrm{cm}$，内側の画像の結合方向を横から縦に変更した場合の $\Delta_{stretch1} = 40\,\mathrm{cm}$，外側の属性とその他すべての結合方向を横から縦に変更した場合の $\Delta_{stretch2} = 15\,\mathrm{cm}$ とすると，

$$\frac{\Delta_{stretch1}}{\Delta_{excess}} = \frac{40}{20} = 2$$
$$\frac{\Delta_{stretch2}}{\Delta_{excess}} = \frac{15}{20} = 0.75$$

となり，閾値 θ_{flex} が

$$2 \leq \theta_{flex}$$

の場合は内側の変換が採用されるが，

$$0.75 \leq \theta_{flex} < 2$$

の場合は外側の変換が採用され，

$$\theta_{flex} < 0.75$$

の場合，この例では解が得られないこととなる．つまりこの閾値が大きければ大きいほど内側の変換を試みる．閾値が小さければ小さいほど幅を縮めたことによって縦方向へ大きく変化するような変換は行わないため，結果として外側のレイアウト変換を許容する可能性が高まるとこになる．

6.3 アルゴリズムの終了

本システムでは図 9 に示すように，レイアウト修正とページ分割によって制限された領域に情報を収めるが，最上位の反復子の 1 インスタンスが 1 ページに収まらない場合はレイアウト最適化が成功しなかったことをユーザに伝え，解を出さずに終了する．PDF がオンライン文書として利用できることを考えて，ハイパーリンクを利用し 1 インスタンスの一部を別ページにすることも可能であるが，印刷した場合その情報は失われてしまう．ハイパーリンクとともに該当する情報が記述されたページ数を出力することで対応することも考えられるが，この問題については現在考慮中である．

7. 実 行 例

本システムではクエリ中に細かな装飾指定をすることで，一般の出版物として耐えうるような品質の出力を得ることが可能である．しかし，ユーザが紙面のサイズを意識せずにレイアウト指定を行ったような場合，装飾指定が不十分であるために紙面に情報が収まらない可能性がある．本章では一般的に考えられる PC に関する情報誌・多ページカタログのようなレイアウトを例にとり，本システムが実際にどのような修正をクエリに対して行うかを示す．

ここで本章で取り扱うデータベースについて簡単に説明する．

- 関係 item：性能や付属品など，PC の基本的な情報を保持．
- 関係 shop：PC を扱う量販店に関する情報を保持．
- 多対多関連 sale：各店舗の販売する PC の在庫数などの情報を保持．

7.1 レイアウト最適化

最上位の反復子が縦反復で，生成された表構造がページ幅をオーバした場合のレイアウト最適化実行例を 2 通り示す．

7.1.1 セル幅の指定

図 14 に示すクエリによって生成されるページ内に収まらない表構造に対して，装飾子によるセル幅指定を自動的に付加してページ内に情報をレイアウトしたレイアウト最適化実行例を**図 15** に示す．

ユーザにとって，データベース中の画像のサイズや各文字列の文字数は不明であるため特に意識せずに問合せを行う場合，たとえセルの幅指定を行ったとしてもそれが効果的かどうかは分からない．そのため図 15 のように生成された表構造がページ幅に収まりきらない可能性は高い．このようなクエリに対して上記の箇所に効果的に幅指定を付与してレイアウト最適化を行う．

この例では，PC の画像が幅 8.5 cm，そして OS と CPU を横結合したものとインタフェース類とアクセサリ類の縦結合の最大幅が 45.3 cm，それらが横に結合した 53.8 cm がオリジナルクエリで生成された表構造の幅となっているため，余白を除いた 20 cm 幅の紙面に対して大きく超過してしまっている．横結合（C1）では Δ_{excess} を結合した各要素の最大幅に比例して分配するため，33.8 cm の超過分を 8.5 : 45.3 に分配して PC の画像が幅 3 cm に，縦結合のセルが 17 cm に縮小されている．さらに OS と CPU の横結合に関して結合したセルの最大幅がこの 17 cm を超過しているため，ここでも超過分を分配して，OS が 6.1 cm，CPU が 10.9 cm になるようにセル幅が指定された．

7.1.2 横反復の折り畳み

図 16 に示すクエリによって生成されるページ内に収まらない表構造に対して，セル幅の指定のほかに横反復子を折り畳む指定を自動的に付加してページ内に情報をレイアウトしたレイアウト最適化実行例を**図 17** に示す．

画像のサイズや各文字列の文字数と同様に，反復子における反復するデータインスタンスの数はあらかじめユーザには分からない．ページ幅を超えないように折り畳み指定をすることは可能であるが，その反復子が他の要素と横結合されている場合，各々にどれだけの幅を割り当てるのが効果的かという判断はできない．このような場合も，上記の箇所に効果的に幅指定と折り畳み指定を付与してレイアウト最適化を行う．

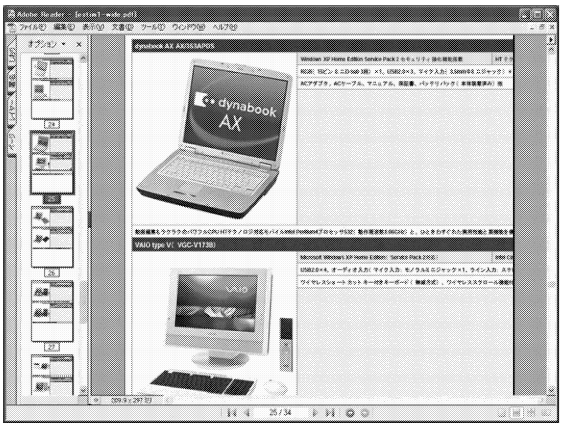
この例でも先ほどと同様に，PC の画像とその特徴，そしてその PC が販売されている店舗の写真を横反復したもの，この 3 要素の横結合が大きくページ幅を超過してしまっている．その超過分を各要素の最大幅に比例して分配した結果，PC の画像と特徴のセルに幅指定，店舗写真の横反復に折り畳み指定が付加されて

```
GENERATE PDF                                          ← 幅指定追加箇所
[
  i.name@{bgcolor=#404040, fontcolor=#FFFFFF, size=10} !
  {
    imagefile(i.photo, path="image/")@{align=center},
    {
      {
        i.os@{bgcolor=#FFDDDD} , i.cpu@{bgcolor=#DDDDFF}
      } !
    i.interface !
    i.accessory
    }
  } !
  i.feature
]!
FROM item i
;
```

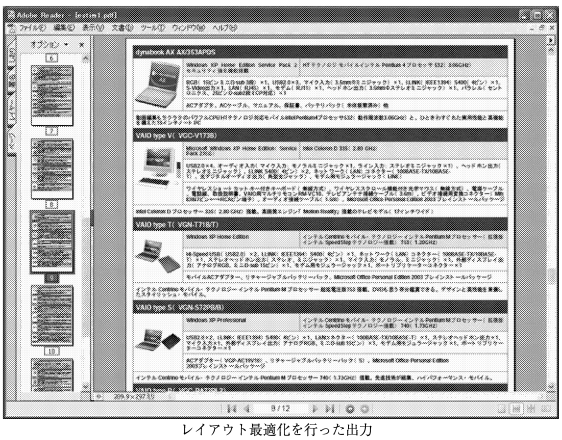
図 14 実行例 1 のクエリ
Fig. 14 Query of output result 1.

```
GENERATE PDF                                          ← 幅指定追加箇所
[
  i.name@{bgcolor=#404040, fontcolor=#FFFFFF, size=12} !
  {
    imagefile(i.photo, path="image/")@{valign=center},
    i.feature@{size=10},
    {
      "販売店"@{size=10, align=center, bgcolor=#FFDDDD} !
      [ imagefile(s.photo, path="image/") ],   ← 折り畳み指定追加
    }
  }
]!
FROM shop s, item i, sale x
WHERE s.id = x.shop_id AND i.id = x.item_id
;
```

図 16 実行例 2 のクエリ
Fig. 16 Query of output result 2.

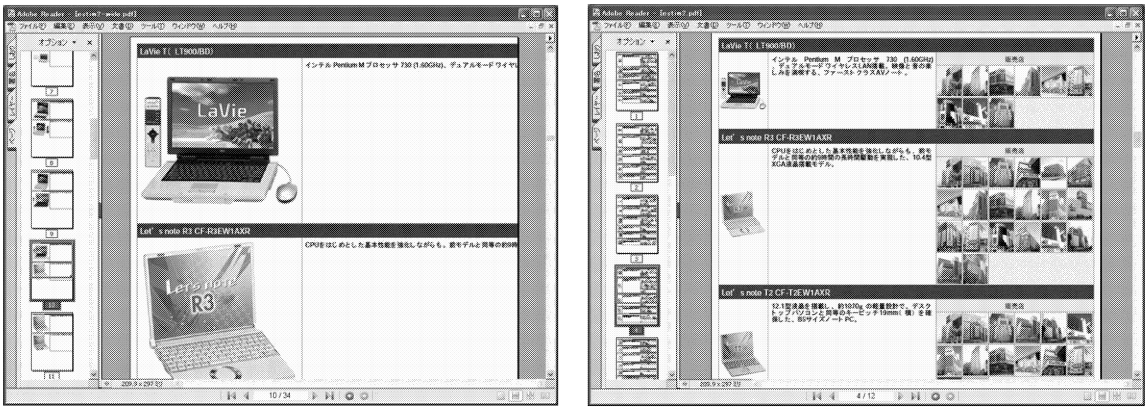


レイアウト最適化を行わない出力



レイアウト最適化を行った出力

図 15 実行例 1：レイアウト最適化
Fig. 15 Output result 1: Layout optimization.



レイアウト最適化を行わない出力　　レイアウト最適化を行った出力

図 17 実行例 2：レイアウト最適化
Fig. 17 Output result 2: Layout optimization.

```
GENERATE PDF                                          ← 幅指定追加箇所
[
 i.name@{bgcolor=#404040, fontcolor=#FFFFFF} ,①  ①  外側を変更した場合の
 {                                                    レイアウト変換箇所
  {
   imagefile(i.photo, path="image/")@{valign=center} ,
   {
    i.feature@{size=10} !
    "OS"@{bgcolor=#707070, fontcolor=#FFFFFF} ! i.os@{size=10} !
    "CPU"@{bgcolor=#707070, fontcolor=#FFFFFF} ! i.cpu@{size=10} !
    "ディスプレイ"@{bgcolor=#707070, fontcolor=#FFFFFF} ! i.display@{size=10}
   }
  } !
  "販売店"@{size=10, align=center, bgcolor=#FFDDDD} !
  [ imagefile(s.photo, path="image/")@{width=60} ],   ← 折り畳み指定追加
 }
]!
FROM shop s, item i, sale x
WHERE s.id = x.shop_id AND i.id = x.item_id
;
```

図 18 実行例 3 のクエリ
Fig. 18 Query of output result 3.

図 17 のようなレイアウトに変換されている．

7.2 柔軟性の効果

図 18 に示すクエリによって生成されるページ内に収まらない表構造に対して，とにかく内側のレイアウト変換を追求してレイアウト最適化を行ったものと，柔軟性の閾値を用いることで外側での変換を許容し，縦方向への変化量を抑えてレイアウト最適化を行った実行例を**図 19** に示す．

外側でのレイアウト変換を許容する場合，幅指定と折り畳み指定の長さを内側の変換を追求した場合ほど縮めず，さらに上記のように横結合が 1 カ所縦結合に変換される．

この例では，まずオリジナルクエリの幅超過分 Δ_{excess} が 29.4 cm，1 つの PC に関するインスタンスの高さが，

PC 画像 + 文字列のセル + 販売店画像
$= 8.5\,\mathrm{cm} + 1.0\,\mathrm{cm} + 2.6\,\mathrm{cm} = 12.1\,\mathrm{cm}$

であった．ここで柔軟性の閾値 θ_{flex} を 0.2 にした場合，図 19 の左下の変換が採用された．この変換後の

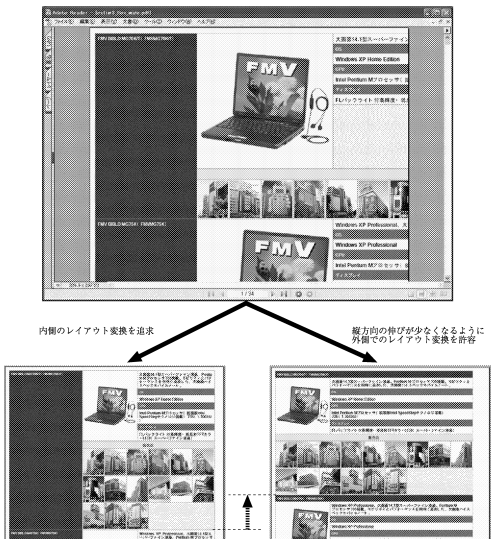


図 19 実行例 3：柔軟性の効果
Fig. 19 Output result 3: Effect of flexibility.

1 つの PC に関するインスタンスの高さは，

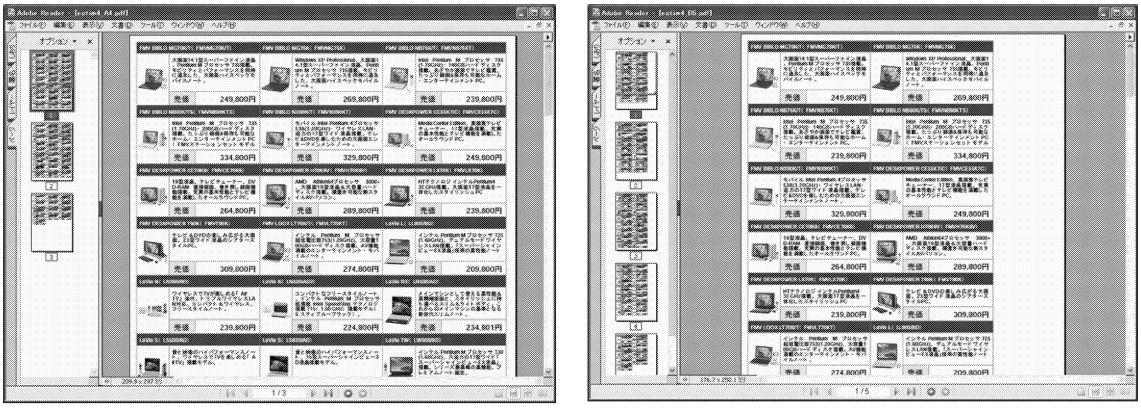
PC 画像の右にある縦結合 + 文字列のセル + 販売店画像 × 3
$= 7.7\,\mathrm{cm} + 1.0\,\mathrm{cm} + 2.6\,\mathrm{cm} \times 3 = 16.5\,\mathrm{cm}$

よって $\Delta_{stretch}$ は 16.5 − 12.1 で 4.4 となる．この変換における Δ_{excess} に対する $\Delta_{stretch}$ の比は

$$\frac{\Delta_{stretch}}{\Delta_{excess}} = \frac{4.4}{29.4} = 0.150$$

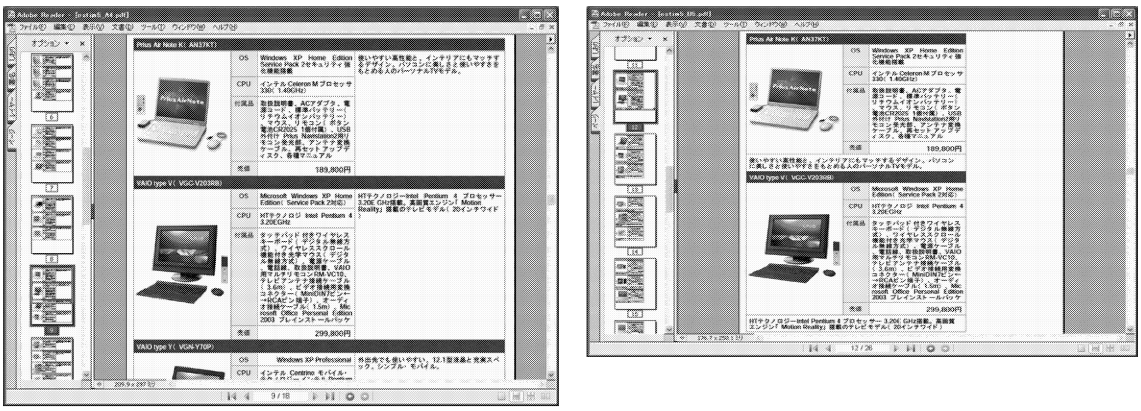
であり，閾値よりも低い値になっている．次に θ_{flex} を 0.1 にした場合先ほどの変換は採用されず，図 19 の右下の変換が採用された．この変換後の 1 つの PC に関するインスタンスの高さは，

PC 名のセル + PC 画像 + 文字列のセル + 販売店画像 × 2



A4 用紙を想定してレイアウト指定された出力　　B5 用紙にレイアウト最適化された出力

図 20 用紙サイズの縮小にともなうレイアウト最適化（実行例 4）
Fig. 20 Layout optimization according to reduction of paper size (output result 4).



A4 用紙を想定してレイアウト指定された出力　　B5 用紙にレイアウト最適化された出力

図 21 用紙サイズの縮小にともなうレイアウト最適化（実行例 5）
Fig. 21 Layout optimization according to reduction of paper size (output result 5).

$$= 1.0\,\mathrm{cm} + 5.8\,\mathrm{cm} + 1.0\,\mathrm{cm} + 2.6\,\mathrm{cm} \times 2$$
$$= 13.0\,\mathrm{cm}$$

よって $\Delta_{stretch}$ は 13.0 - 12.1 で 0.9 となる．この変換における Δ_{excess} に対する $\Delta_{stretch}$ の比は

$$\frac{\Delta_{stretch}}{\Delta_{excess}} = \frac{0.9}{29.4} = 0.031$$

であり，新しい閾値 0.1 よりも低い値である．

7.3 出力用紙サイズの変更

このレイアウト最適化アルゴリズムは出力した表構造が予想外にページに収まりきらなかった場合だけでなく，ユーザが A4 の紙面にレイアウトすることを前提に細かくレイアウト指定をした出版物を B5 の紙面にレイアウトし直すような場合，つまり出力用紙サイズの縮小時にも適用できるためクエリの再利用が可能となる．

図 20 に示す実行例はユーザが指定した細かな指定はユーザの意図として変更せず，折り畳み幅を B5 用紙に合わせて変更するのみである．**図 21** に示す実行例でも同様に細かな変更せず，B5 用紙に合わせて 1 カ所横結合を縦結合に変更している．

8. 評価・検討

8.1 手動レイアウト変換との比較

前章では情報誌や多ページカタログを意図したクエリを自動変換した出力結果を示した．ここでその有用性を評価するために，20 人の被験者に対して次のような実験を行った．7.1 節で紹介した 2 つの実行例を本システムによるレイアウト最適化を行わずに，被験者が手作業でレイアウト指定を修正することで表構造をページ内に収めるのにかかる時間を測定し，システムによる自動変換にかかる時間と比較した．そしてさらにその自動変換による出力が，どれだけ被験者の意図に近いレイアウトとなっているかを調べた．本来こ

表 3 本システムの最適化と手作業によるレイアウト変換との所要時間比較
Table 3 Comparison of excusion time between optimization of this system and layout modification by hand.

	実験 1（図 15）	実験 2（図 17）
手作業（秒）	最低：225 平均：332 最高：446	最低：194 平均：253 最高：295
本システム（秒）	22	20

表 4 出力結果のレイアウト指定の比較
Table 4 Comparison of layout specification.

	実験 1（図 15）	実験 2（図 17）
異なる属性に対する幅指定箇所	5	2
異なる属性に対するフォントサイズの指定箇所	4	1
レイアウト指定演算子の変更箇所の違い	4	7
幅指定の長さの差が被験者の指定した長さの 1 割以上である箇所	7	15

のシステムは研究開発中の SuperSQL の GUI を利用して実行することを想定しているが，GUI もまだ開発途中であり連携がとれないことから，実験においては直接 SuperSQL クエリを記述して実行した．そのため本実験では SuperSQL クエリの構文を理解している者を被験者とした．

2 つの実行例のどちらに関しても，被験者は 7.1 節で紹介した修正を加える前のクエリのレイアウト指定によって情報を出力したかったと仮定し，事前にクエリとそのクエリによって生成される表構造がページに収まっていない PDF ファイルを見せ，よく理解してもらう．またページ幅が余白を除いてどれだけの長さであるかも明示した．あくまでもクエリの指定するレイアウト構造が作成者の意図に基づくレイアウトであるが，被験者には装飾指定の追加以外にも必要に応じてレイアウト演算子の変更を行ってもらった．クエリを書き換えて実行した出力がまだページに収まっていない場合は，その出力を参考に引き続きページ内に収まるまで実験を行った．

8.1.1 所要時間の比較

表 3 に手作業によるレイアウト変換にかかった時間と，本システムによるレイアウト最適化にかかった時間とをまとめた．この時間にはシステム・手作業ともに単純にレイアウトを変換するのにかかった時間ではなく，データベースへの問合せなどを含めた実行全体にかかった時間である．

実験の結果，本システムのレイアウト最適化が手作業によるレイアウト変換よりもほとんどの場合 10 倍以上速いことが分かった．時間がかかる主な原因は，データベースに格納された画像の大きさや文字列の文字数，反復されるデータインスタンス数が事前に分からないことから，横連結されている要素各々にどれほどの割合でページ幅を割り振るべきかの判断ができないということであった．2 つの実験を通して，1 回の変換で紙面に情報を収めることができた被験者は 1 人だけであり，図 15 の変換に関しては，ほとんどの被験者がシステムの変換結果と同様にレイアウトを変えずに幅指定をしただけにもかかわらず最低 3 回，最高 5 回実行を繰り返さなければ表構造をページ内に収めることができなかった．これに対し図 17 の変換に関しては，最低 1 回，最高 3 回とより少ない実行回数で出力を得ることができた．これは折り畳み指定によるレイアウト幅縮小はさほど手間がかからないことを示すと同時に，折り畳み装飾子の使用価値の高さを示しているといえる．

8.1.2 出力結果のレイアウト指定の比較

システムがどれだけユーザの思い描くレイアウトに近いレイアウト構造に自動変換できているかを評価するために，「被験者が自らクエリを修正した出力」と「システムの自動変換による出力」に関して，以下の 4 点について各々20 人の被験者における合計数を調べることで，両出力のレイアウト指定と装飾指定の類似度を比較し，その結果を**表 4** にまとめた．

- 異なる属性に対する幅指定箇所
- 異なる属性に対するフォントサイズの指定箇所
- レイアウト指定演算子の変更箇所の違い
- 幅指定の長さの差が被験者の指定した長さの 1 割以上である箇所

なお実験 1 に関しては 7 個，実験 2 に関しては 5 個の属性がクエリ中に存在している．結果どの項目においても 1 人あたり 1 カ所以下であり，このシステムによってユーザが手作業によってレイアウトを指定した出力とさほど変わらない出力を得られることが分かった．

8.1.3 レイアウト最適化の満足度

被験者を対象に 7 章で紹介した各実行例を実際にシ

ステム上で実行してもらい，本システムのレイアウト最適化の満足度をアンケートによって調査した．図 19 以外に関して，それぞれ実際に実行してみてこのシステムを使用したいかという問いに対して，図 15 では 15 人，図 17 では 13 人，図 20 では 18 人，図 21 では 17 人が使用したいと答えた．このアンケートにおいて被験者から特に評価が高かったのは，図 20 と図 21 の出力用紙サイズの縮小に対する効果であった．クエリを見ることもなく手軽に満足のいく出力を得ることができるという点が評価されたポイントである．

図 19 に関しては，柔軟性の概念を導入したことによるもう一方へ大きく伸びてしまうレイアウト変換の抑制に有効性を感じるかという問いに対して，20 人全員が感じると答えた．しかし便利といえるが閾値をどれくらいに設定すれば出力結果に変化が表れるのかが分かりにくいという意見があり，これは今後の課題といえる．

8.2 従来の関連システムとの比較

本研究のレイアウト最適化を含めたシステムの出版能力を従来の関連システムと比較し評価した．

8.2.1 SuperSQL の HTML 出力・Actiview システムとの比較

SuperSQL の HTML 出力でも，生成された HTML ソースをブラウザから印刷することでデータベースからの検索結果を出版することができる．しかし生成された表構造を紙面幅に収めるためには，細かな指定が必要になる．セル幅はある程度自動で調節されるが，セルの幅が縮まりすぎてはテキストの折り返しが頻繁に起こってしまい視認性の低い表構造になってしまう．これに対し SuperSQL を利用した Actiview システム[7] ではブラウザの横幅に合わせてレイアウトの変換を行うため，これをうまく利用すれば HTML 出力に比べて視認性の高い表構造で紙面内にレイアウトすることができる．

しかし，たとえ HTML 出力において表構造の幅が紙面に収まるように細かくレイアウトを指定したり，Actiview を利用したりしたとしても，HTML にはページという概念がないため，場合によっては印刷時のページの切れ目が表構造を非常に見にくいものにしてしまう．

8.2.2 Microsoft Access におけるレポート出力との比較

Access のレポート出力では**図 22** のように GUI によってレポートを作成することができる．以下にその手順を述べる．

- 1 つ以上のテーブル or クエリから出力する属性を選択．
- どの属性でグルーピングするかを選択．
- より内側でのグルーピングの順序を指定．
- ソートする属性を選択（4 つまで）．
- 表のタイプを選択．
- 全体的なデザインのタイプを選択．

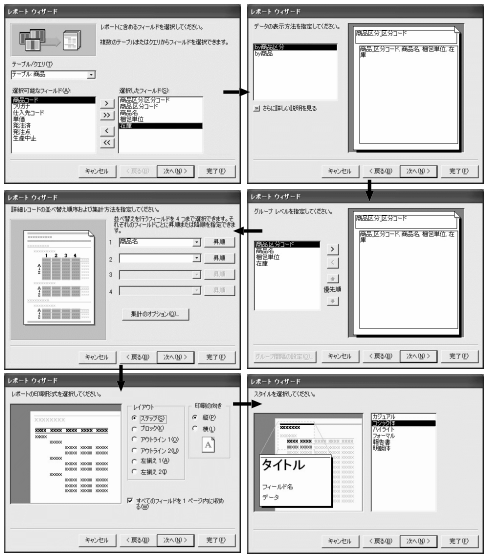


図 22 Microsoft Access におけるレポート作成手順
Fig. 22 Screenshot of how to make report in Microsoft Access.

図 22 のように Access では，コンピュータと対話しながらレポートを作成できるため，初心者でも比較的簡単に扱うことができる．

しかし，Access ではレイアウトの自動変換機能はなく，数多くの属性を紙面内に収めたい場合やはり細かなレイアウト指定が必須であり，対話的とはいえユーザにかかる負担は大きい．また Access ではレイアウトに関して多くの制約があり，情報を適切な位置関係でコンパクトにレイアウトすることが難しい．まず，本システムではデータを反復出力する場合，縦方向のほかにも横方向が指定可能だが，Access では縦方向しか指定することができない．また，グルーピングの階層も 10 レベルまでと決められている．本システムはこの点に関して特に制限はない．さらに Access では，グルーピングするものの右または下側に該当するものを並べることしかできないため，より上位階層の要素を右または下側に配置するような表構造は実現することができない．一般により上位階層の要素を右または下側に配置するような構造をした表はけっして少なくなく，こういった構造を表現できることは非常に有効であるといえる．折り返し指定ができないため，

1 ページ内に収めたい情報がページを超えてしまった場合，文字の大きさを小さくするほかない．本システムでは柔軟なレイアウト指定が可能であるために文字の大きさを小さくすることなく，情報をよりコンパクトにまとめることができる．

9. 結　　論

本研究では，個人がデータベースの内容を紙媒体に高品位で効率的に出力したいという需要に対して，出力結果が紙面に入りきらない場合自動的にレイアウト最適化を行う手法を提案し，実装した．レイアウトの最適化においては，本研究ではレイアウト構造のより内側を変換することで紙面内に情報を収めることを目指す一方で，外側のレイアウト変換を許容し表構造の大きな変化を抑制する手法を提案した．内側と外側の変更を利用者が大域的にコントロールするために，柔軟性の概念と，その閾値の指定を導入した．

また，実験によって手作業でクエリを修正し紙面内に表構造を収めた出力を得るまでにかかる時間と，本システムのレイアウト最適化アルゴリズムによる変換を利用して紙面内に表構造を収めた出力を得るまでにかかる時間を比較し，本システムを用いることで平均で 10 倍以上時間を短縮できることを示した．そして被験者が自らクエリを修正した出力と，システムの自動変換による出力のレイアウト指定を比較し，その相違箇所が平均して 1 人あたり 1 カ所程度であることを示した．さらに実験に使用した例に関して，実際にシステムを実行してもらったうえで被験者に対してアンケートを行い，自動変換によって得られた出力のレイアウト構造に対する満足度を調査した．その結果，被験者の約 8 割が本システムを利用したいと答えた．特にすでにあるサイズの紙面上に出版されたものをより小さなサイズの紙面に出力する際にこのシステムを利用した場合の満足度はきわめて高いことが分かった．最後に，本システムによって得られる出力を印刷した場合と，SuperSQL の HTML 出力そして Actiview システムによって生成された HTML ソースをブラウザから印刷した場合との比較，Microsoft Access のレポート出力と本システムの表現力の比較を行い本システムの評価を行った．

個人がデジタルデータを扱う機会が急増し，データベースが身近な存在となるなかで，一般ユーザが手軽に検索結果を紙媒体に印刷することを可能にする本手法によって SuperSQL を用いたデータベース出版システムの実用性が高まったといえる．

参 考 文 献

1) Bickmore, T., Girgensohn, A. and Sullivan, J.: Web Page Filtering and Re-Authoring for Mobile Users, *The Computer Journal*, Vol.42, No.6, pp.534–546 (1999).
2) Bickmore, T. and Schilit, B.N.: Digestor: Device-independent Access to the World Wide Web, *Proc. 6th Internationl WWW Conference*, pp.655–663 (1997).
3) Buyukkokten, O., Garcia-Molina, H. and Paepcke, A.: Accordion Summarization for End-Game Browsing on PDAs and Cellular Phones, *SIGCHI '01*, Vol.3, No.1, pp.213–220 (2001).
4) Buyukkokten, O., Kaljuvee, O., Garcia-Molina, H., Paepcke, A. and Winograd, T.: Efficient Web Browsing on Handheld Devices Using Page and Form Summarization, *ACM Trans. Inf. Syst.*, Vol.20, No.1, pp.82–115 (2002).
5) Chen, Y., Ma, W.-Y. and Zhang, H.-J.: Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices, *Proc. 12th International WWW Conference*, pp.225–233 (2003).
6) GmbH, P.: http://www.PDFlib.com.
7) Maeda, Y. and Toyama, M.: ACTIVIEW: Adaptive data presentation using SuperSQL, *Proc. VLDB '01*, pp.695–696 (2001).
8) Merz, T.: PDFlib Reference Manual Version 6.0.1 (2004).
9) Miller, R., Tsatalos, O. and Williams, J.: DataWeb: Customizable Database Publishing for the Web, *IEEE MultiMedia*, Vol.4, No.4, pp.14–21 (1997).
10) Seto, T., Nagafuji, T. and Toyama, M.: Generating HTML Sources with TFE Enhanced SQL, *ACM Symposium on Applied Computing*, pp.96–100 (1997).
11) Toyama, M.: SuperSQL: An Extended SQL for Database Publishing and Presentation, *Proc. ACM SIGMOD '98*, pp.584–586 (1998).
12) Toyama, M. and Nagafuji, T.: Dynamic and Structured Presentation of Database Contents on the Web, *Proc. EDBT '98*, pp.451–465 (1998).
13) Wesley, A.: *Portable Document Format Reference Manual* (1993).
14) 増田英孝，塚本修一，安富大輔，中川裕志：HTML の表形式データの構造認識と携帯端末表示への応用，情報処理学会論文誌：データベース，Vol.44, No. SIG12(TOD 19), pp.23–32 (2003).
15) 亀岡慎平，遠山元道：SuperSQL における PDF

出力の実装，第 14 回データ工学ワークショップ：DEWS2003 (2003).

(平成 17 年 3 月 20 日受付)
(平成 17 年 7 月 5 日採録)

（担当編集委員　田島 敬史）

亀岡　慎平（正会員）

平成 15 年慶應義塾大学理工学部情報工学科卒業．平成 17 年同大学大学院開放環境科学専攻修士課程修了．

遠山　元道（正会員）

昭和 54 年慶應義塾大学工学部管理工学科卒業．昭和 59 年同大学大学院博士課程修了．同年助手．現在情報工学科専任講師．博士（工学）．平成 8 年オレゴン大学院大学（OGI）客員研究員．平成 10〜13 年 JST さきがけ研究 21 研究者．主にデータベースの研究に従事．電子情報通信学会，ACM，IEEE 会員．