

学位論文 博士（工学）

SuperSQLを用いた適応型Webビュー
の生成と最適化

2009年度

慶應義塾大学大学院理工学研究科

慎 祥 揆

目次

1	はじめに	1
1.1	背景と研究目標	2
1.2	論文の構成	5
2	ACTIVIEW の実現	7
2.1	SuperSQL の概要	8
2.1.1	TFE (Target Form Expression) とは	9
2.1.2	結合子	11
2.1.3	反復子	12
2.1.4	装飾子	15
2.1.5	関数	16
2.1.6	文字列定数	17
2.2	ACTIVIEW	18
2.3	システム構成	20
3	ACTIVIEW における適応化	27
3.1	幅制約	28
3.2	開発者制約	32
3.2.1	開発者制約 - 順序	33
3.2.2	開発者制約 - 入れ子	33
3.2.3	開発者制約 - グループिंग	35
3.2.4	中括弧の導入	36
3.3	幅占有率目標	38
3.4	充填率目標	39
3.5	長さ目標	40
4	ACTIVIEW によるレイアウト変換	44
4.1	レイアウト変換の基本概念	45

4.2	横縦変換による幅変化	45
4.3	リンク変換による長さ変化	47
4.4	予備実験	48
4.4.1	従来のレイアウト変換	48
4.4.2	予備実験による比較	50
4.4.3	$SEQ(Same)$ 適用手法	52
4.4.4	$SEQ(Group)$ 適用手法	54
4.4.5	$SEQ(AllStringFirst)$ 適用手法	54
4.4.6	$P_B(Same)$ 適用手法	57
4.4.7	$P_B(Group)$ 適用手法	58
4.4.8	$P_B(AllString)$ 適用手法	61
4.4.9	Web 開発者の意味反映	63
5	レイアウト変換適用の戦略	65
5.1	ACTIVIEW によるレイアウト変換	66
5.2	ACTIVIEW のリンク変換	68
5.2.1	リンク変換戦略の基本概念	68
5.2.2	反復子との連結子にリンク	69
5.2.3	中括弧内でのリンク変換の禁止	71
5.2.4	多くの子要素をもつネストの優先	72
5.2.5	長いネストの優先	73
5.2.6	ネストの長さ予測アルゴリズム	75
5.2.7	奥にあるネストの優先	80
5.2.8	リンク変換適用アルゴリズム	80
5.3	統一変換可能な連結	82
5.4	一貫性のあるレイアウトの生成	84
5.4.1	同一領域内 (同一ネスト内・同一中括弧) での変換の優先	85
5.4.2	反復子 (ネスト指定) , グルーピングとの結合の優先	87
5.4.3	上位ノードの優先	89
5.4.4	少ない変換適用の優先	91
5.4.5	中括弧内の変換の統一変換の優先	92
5.4.6	リンク変換の優先順位の変換	94
5.4.7	メインアルゴリズム	94
5.4.8	レイアウト変換戦略アルゴリズム	96
5.4.9	結果レイアウトの選択	98

6	実験および評価	99
6.1	実験方法	100
6.2	実験結果・評価	101
6.3	実際の Web ページへの適用	119
6.3.1	カテゴリ化可能なページへの適用結果と比較	122
6.3.2	データ集約的な Web ページへの適用結果の比較	132
6.3.3	統計データあるいは複雑なテーブル型ページへの適用結果の比較	138
7	関連研究	146
7.1	Web コンテンツの抽出, 省略, 要約	147
7.1.1	Digestor システム	147
7.1.2	Power Browser システム	149
7.2	特殊 Web ブラウザ	150
7.2.1	WEST	151
7.2.2	RSVP	152
7.3	Markup Language	154
7.3.1	XHTML	154
7.3.2	WML (Wireless Markup Language)	155
7.4	従来の研究との比較	156
8	結論	158

表 目 次

2.1	結合子の種類と意味	12
2.2	反復子の種類と意味	13
3.1	結合子ごとの計算	31
5.1	各変換手法の比較	66
5.2	各ネストにある属性 t の IC_t 計算結果	75
6.1	結果レイアウトの選択	115
6.2	日本のアクセス数上位 40 位の Web サイト	120
6.3	Web ページの構造化結果の比較	144

目次

2.1	Trinity データモデル	8
2.2	各結合子の問合せ文の例と結果テーブル	12
2.3	単に各々の一覧が表示されるレイアウトの例	14
2.4	ネストされた構造のレイアウトの例	14
2.5	各反復子の問合せ文の例と結果テーブル	15
2.6	文字列定数を用いた例	17
2.7	従来の Web 開発	19
2.8	ACTIVIEW によるレイアウト生成の概念	19
2.9	システムの概要図	21
2.10	通常の SQL 文とその結果	23
2.11	SuperSQL よって構造化された Web ビューの例	23
2.12	ACTIVIEW の問合せ文	24
2.13	ACTIVIEW の問合せ文 (図 2.12) から生成される Web ビュー	25
2.14	レイアウト情報の木構造表現	26
3.1	レイアウトーサイズ式の木構造表現	30
3.2	開発者による属性の順序定義	33
3.3	チームによるネスト	34
3.4	ポジションによるネスト	34
3.5	中括弧との横連結による結果レイアウト	35
3.6	中括弧が使用されていない場合の問合せ文と結果レイアウト	36
3.7	中括弧が使用されている場合の問合せ文と結果レイアウト	36
3.8	coach の情報のみが中括弧でくくられている場合	37
3.9	中括弧を自動的に導入した場合	37
3.10	ユーザ表示画面に対する結果レイアウト	38
3.11	充填率の比較	39
3.12	長さ目標を満たす前のレイアウト	41
3.13	長さ目標を満たしたレイアウト	42

4.1	横縦変換	46
4.2	リンク変換	47
4.3	単純順序探索による変換手法	49
4.4	単純順序探索と全数探索の実行時間	50
4.5	予備実験に使われた問合せ文	51
4.6	結果レイアウトの幅占有率と充填率の割合 ($SEQ(Same)$)	52
4.7	$SEQ(Same)$ 手法によって生成される結果レイアウトの例	53
4.8	候補レイアウトの幅占有率と充填率の割合 ($SEQ(Group)$)	54
4.9	$SEQ(Same)$ と $SEQ(Group)$ で生成される候補レイアウトの比較	55
4.10	候補レイアウトの幅占有率と充填率の割合 ($SEQ(AllStringFirst)$)	55
4.11	$SEQ(Same)$ 手法によって生成される 840pixel の幅をもつ候補レイアウト	56
4.12	$SEQ(AllStringFirst)$ 手法によって生成される 780pixel の幅をもつ候補 レイアウト	56
4.13	候補レイアウトの幅の種類と充填率 ($P_B(Same)$)	57
4.14	$P_B(Same)$ によって生成されるレイアウト結果の例	58
4.15	候補レイアウトの幅の種類と充填率 ($P_B(Group)$)	59
4.16	$P_B(Group)$ によって生成されるレイアウト結果の例 (760pixel の幅)	59
4.17	$P_B(Group)$ によって生成されるレイアウト結果の例 (670pixel の幅)	60
4.18	$P_B(Group)$ によって生成されるレイアウト結果の例 (690pixel の幅)	60
4.19	候補レイアウトの幅の種類と充填率 ($P_B(AllString)$)	61
4.20	$P_B(AllString)$ によって生成されるレイアウト結果の例 (710pixel の幅)	62
4.21	$P_B(AllString)$ によって生成されるレイアウト結果の例 (710pixel の幅)	62
4.22	予備実験に使われた選手に関する問合せ文	63
4.23	レイアウト変換を行う際の意味を反映	64
4.24	リンク変換の適用によって作成されたマルチページ	64
5.1	レイアウト変換の流れ	67
5.2	変換対象の探索	68
5.3	ネスト構造でのリンク優先	70
5.4	中括弧内でのリンク変換	71
5.5	多くの子要素をもつネスト優先変換戦略	73
5.6	データ代表要素個数	74
5.7	長いネスト優先リンク変換戦略	80
5.8	奥のネスト優先変換戦略	81
5.9	統一変換可能な連結	83

5.10	複数のネスト内と同一ネスト内での変換の木構造例	85
5.11	複数のネスト内で変換が行われた結果レイアウトの例	86
5.12	同一ネスト内で変換が行われた結果レイアウトの例	86
5.13	反復子との連結子に変換を行う木構造例	87
5.14	ネスト内で変換が行われた場合の結果レイアウト	88
5.15	ネストとの連結子に変換が行われた場合の結果レイアウト	88
5.16	ネスト内の上位ノードを優先する木構造例	89
5.17	優先順位がない場合の結果レイアウト	90
5.18	ネスト内の上位ノードを優先した場合の結果レイアウト	90
5.19	少ない変換適用を優先する例	91
5.20	同一中括弧内の文字列定数を同時に変換する例	92
5.21	同時に変換が行われた場合の結果レイアウト	93
5.22	同時に変換が行われなかった場合の結果レイアウト	93
6.1	データベースのテーブルスキーマ	100
6.2	問合せ文でのネスト指定と結果レイアウトの例	102
6.3	リンク戦略に従って生成されたレイアウトの例	103
6.4	幅条件だけを基準にして分割したレイアウトの例	104
6.5	リンク変換を行う連結子	104
6.6	リンク先のレイアウトになる範囲の例	105
6.7	中括弧の適用による範囲指定	106
6.8	複数の中括弧内の連結子変換による結果レイアウトの例	106
6.9	同一ネスト内の連結子変換による結果レイアウトの例	107
6.10	文字列定数に同時に変換を行った結果レイアウトの例	108
6.11	システムが指定した中括弧間の連結子に変換を行った結果レイアウトの例	108
6.12	上位中括弧間の連結子に変換を行った結果レイアウトの例	110
6.13	上位ノードに変換を行った結果レイアウトの例	110
6.14	反復子, グルーピングとの結合を優先したレイアウトの例	111
6.15	親ノードの方の変換を優先した結果レイアウトの例	111
6.16	幅制約を満たしたリンク先の結果レイアウト	112
6.17	変換されたレイアウトの充填率	113
6.18	単純順序探索の結果レイアウト	114
6.19	提案手法による結果レイアウト	114
6.20	変換された結合子の数	116
6.21	実行時間	117

6.22	実際の端末機からのアクセス結果	118
6.23	同一端末機であっても異なる表示画面に対するレイアウト結果	118
6.24	データの縮約による構造の再設計が必要なページの例	121
6.25	閉ざされた Web ページの例	121
6.26	Google ページへの応用例	121
6.27	Yahoo Japan サイトのメインページ	122
6.28	カテゴリのみを取り出して生成した Web ページの例	123
6.29	実際の Web サイトのメインページの例	124
6.30	Yahoo Japan サイトのカテゴリ例	125
6.31	Yahoo ページを再構成するためのスキーマ	125
6.32	ACTIVIEW を用いて再構成した Yahoo ページ	126
6.33	再構成される Yahoo ページを生成する ACTIVIEW の問合せ文	127
6.34	ACTIVIEW の問合せ文 (図 6.33) によって生成される候補レイアウト の幅の種類と充填率	127
6.35	ACTIVIEW によるリンク変換	128
6.36	低い充填率の結果レイアウト	129
6.37	カテゴリ化されている Web ページの例	130
6.38	カテゴリ化されている Web ページを生成する ACTIVIEW の問合せ文	130
6.39	図 6.38 の ACTIVIEW 問合せ文から生成される初期レイアウト	131
6.40	ACTIVIEW の問合せ文 (図 6.38) によって生成される候補レイアウト の幅の種類と充填率	131
6.41	データ集約的な Web ページ	132
6.42	代表的なテーブル型の Web ページの例	133
6.43	テーブル型の Web ページを生成する ACTIVIEW の問合せ文	133
6.44	関連性を付けた ACTIVIEW の問合せ文	134
6.45	Yahoo Japan サイトのうち、地図情報と天気情報の Web ページ	134
6.46	天気情報データベースのテーブルスキーマ	135
6.47	天気情報 Web ビューを生成する ACTIVIEW の問合せ文	135
6.48	天気情報ページの初期レイアウト	136
6.49	Yahoo の天気情報ページ	136
6.50	レイアウト変換戦略による再構成	137
6.51	異なる長さ目標値に従った結果レイアウト	138
6.52	統計データの例	139
6.53	複雑なテーブル型の Web ページ	139

6.54	Kakaku データベースのテーブルスキーマ	140
6.55	図 6.53 のようなテーブルを作成するための問合せ文	140
6.56	ACTIVIEW の問合せ文 (図 6.55) によるレイアウト生成	141
6.57	テーブルの分析	141
6.58	中括弧が指定されていない場合の結果レイアウトの例	142
6.59	中括弧の指定による要素間の関連付けを行った問合せ文	143
6.60	中括弧の指定における結果レイアウトの例	144
6.61	ACTIVIEW の問合せ文 (図 6.59) によって生成される候補レイアウト の幅の種類と充填率	145
7.1	Powser Browser システム	150
7.2	WEST ブラウザ	152
7.3	RSVP ブラウザ	153

第 1 章

はじめに

1.1 背景と研究目標

近年，WWW の普及とともに，WWW を利用したビジネスアプリケーションや，利用者への情報提供，企業内での情報開示がさかんに行われ，ユーザの Web ページに対する要求も多様化している．Web ページ提供側も，それにともない，一人一人のユーザの環境に合わせたビューの提供や，個人向けに差別化された情報を提供することが望まれている [1, 2, 3]．そのため，ユーザごとに異なるコンテンツのページを提供することや，近年増加している小型携帯端末からの WWW の利用を考慮した専用のページを別に作ることが求められている [4, 5, 6]．

本論文では SuperSQL システム [7, 8, 9] を利用することにより，関係データベースからの検索結果を，ユーザごとに異なる Web アクセス環境に適応するアダプテーションの技術を提案する．本論文では，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をアダプテーションと呼ぶ．アダプテーションはパソコンに限らず，小型携帯端末や携帯電話の爆発的増加にともない注目されてきている．

SuperSQL とは，通常の SQL に TFE (Target Form Expression) という構造化演算子を導入した問合せ言語である．演算子の導入によって拡張された問合せ文を SuperSQL の問合せ文と呼ぶ．関係データベースに問い合わせて得た平坦な構造の出力結果を SuperSQL の問合せ文の構造化定義に従って構造化し，SuperSQL の問合せ文で指定したメディアである XML 文書，HTML 文書，PDF [10] 文書，Excel 文書，SWF (Shockwave Flash) ファイルなどに変換するシステムが SuperSQL システムである．

本論文では，WWW からのデータベースの利用という観点から，アダプテーションの問題に取り組んでいる．関係データベースからの検索結果をユーザの Web アクセス環境に提供した結果を Web ビューと本論文では定義する．データベースは，各種情報システムの中核として，多くの情報を格納している．データベースと WWW を統合することで，あらゆる場所から多くの情報を多くの人々が利用できるようになる．Web を通して多くのユーザがデータベースに格納されている情報を共有する状況であるが，大部分はデスクトップコンピュータ環境か，特定のユーザアクセス端末機環境である．現在の手法はこれらの環境に固定された Web ページを表示するものであり，変化するユーザの端末環境への動的な対応ができない．

最近，携帯端末においては，端末機によって表示画面の仕様が異なるだけでなく，表示画面を縦や横に変えながら表示することも可能になっている．このように，変化する端末機の画面サイズや，発表される多様な端末機に 1 つずつ対応した (ユーザ環境に適応した) Web ページを提供することは現在の技術ではきわめて困難である．この問題を解決するために様々な研究や技術の開発が進んでいる．しかし，これらは Web に含まれている内容の削除や縮約などの Web コンテンツに対する解決手段を提案して

いるだけで、Web ページの構造を対象にした研究は行われていない。そこで、前田ら [11, 12, 13, 14, 15] は様々な表示仕様の端末からのデータのアクセスに対応し、ユーザ端末環境に適応した Web ビューを動的に提供することに着目し、適応型表示ビュー ACTIVIEW を提案している。

本論文で紹介する ACTIVIEW は、前田らによって提案された先行研究を基にしている。前田らは SuperSQL の問合せ文の演算子を変換することで、元の問合せ文に定義された初期レイアウトからユーザ表示画面の幅に合うレイアウトに再構成するシステムを Lisp ベースで実現した。しかし、基本的に左から順に横連結を縦連結に変更する単純な手法で行っていたため、ユーザに望ましいレイアウトとならない場合もあった。初期の ACTIVIEW から得られるレイアウトには次のような問題があった [14]。

- 表示画面の横幅に収まることだけを条件としているため、表示画面の幅に対して極端に細いレイアウトになる場合がある。
- ハイパーリンク連結の機械的な導入により、リンク元とリンク先の内容的な関連性が崩れたページを生成する場合がある。
- 表示画面の横幅に収まるという条件だけでは、ユーザにとって欲しいデータが見やすいレイアウトではなくなる場合がある。

この問題点を解決するため、前田は ACTIVIEW のレイアウト変換の最適化の実現による適応化機能とともにプロファイルデータベースの利用方法の改善による個別化機能について新しい提案をした [15]。より見やすいレイアウトを提供するために、結果レイアウトがどれくらいユーザ表示画面を効率的に利用しているか評価する“ FILLTH ”関数と元の問合せ文から生成可能なすべてのレイアウトパターンを候補レイアウトの対象にして最終結果レイアウトを選択する手法を提案した。しかし、提案している“ FILLTH ”関数はレイアウトの構造のみを対象にしているため、意味的に整理されたレイアウト結果を生成しているとは限られない。そのうえ、元の問合せ文から生成可能なすべてのパターンを対象にしているため、最終結果レイアウトの生成時間が非常に長くなってしまいうパフォーマンス的な問題がある。結果レイアウトを生成するのに時間がかかってしまうとユーザアクセスに動的な対応ができなくなる。

適応化を実現するにあたって、動的な Web ビュー生成過程によって得られる結果がユーザに望ましくないものであってはレイアウトの変換を行う意味が低くなる。特に ACTIVIEW のようにユーザ表示画面の幅に応じたレイアウト変換を動的に行う場合、可能なレイアウトの中から最適なレイアウトが効率的（最終結果レイアウトの生成時間を短く）に選ばれることでシステムの有効性が向上する。

さらに、元の間合せ文を定義する開発者の意図的な意味（元の間合せ文に現れる構造定義）をレイアウト変換を行った結果レイアウトに反映する必要と、パフォーマンスの問題を解決するため、より最適化されたレイアウト推薦による効率的な結果レイアウト生成手法を提案する必要性が高まる。

他に、SuperSQL システムが Lisp から Java ベースになったために、SuperSQL システムを用いている ACTIVIEW も再作成する必要性が生じた [16] 。

すなわち、再構成されるレイアウトの最適化問題やレイアウト変換の基準、再構成されたレイアウトを評価する手法や効率的な変換手法などの必要性が高まったため、本研究では新しい最適化手法を提案した上で、全般的に ACTIVIEW システムを再設計し、

- Web 開発の際負担になる、多様な Web ページサイズに動的に対応したレイアウトの自動生成
- ユーザ環境に適応した見やすいレイアウトの提供
- 最終結果レイアウトの構造が意味的に統一された一貫性のあるレイアウトの提供
- ユーザに最終結果レイアウトの提供が速くできるパフォーマンスのよいレイアウト変換戦略

の実現を目指す。

ACTIVIEW は、関係データベースと WWW の連携システムである SuperSQL システムの基礎の上に開発したアダプテーション技術である。関係データベースからの構造化されていない平坦な検索結果を構造化する SuperSQL システムの機能を応用し、この構造を再構成することで様々なユーザ環境、すなわちユーザ表示画面に適応した Web ビューの提供を行う。Web ビューの作成は SuperSQL システムを用いているため、まず ACTIVIEW は SuperSQL の間合せ文をユーザ表示画面に合わせた表（レイアウト）を生成するように書き換える。元の間合せ文を書き換えることによって初期レイアウトとは異なるレイアウトが生成され、ユーザ表示画面に適応した表を提供することが可能になる。

SuperSQL の間合せ文は関係データベースからの結果を入れ子構造に再構成できるように通常の SQL 文を拡張したものである（2.1 節）。

ACTIVIEW では構造化定義ができる SuperSQL の間合せ文の構造定義を、

- 関係データベースからの検索結果を横連結から縦連結にすることによってレイアウトの幅を変更する。
-

- 横か縦に連結しているレイアウトにリンク定義を導入することで結果レイアウトの長さを変更する。

という 2 つのレイアウト変換の基本概念を用いて, SuperSQL の問合せ文に定義された初期レイアウトの再構成を行う。

本論文では, この基本概念に従うレイアウトの構造変換は, ユーザ環境に最適なレイアウトをもつ Web ビューを提供するべきであるという動機から, 2 つの最適化制約と 3 つの目標指標をレイアウト変換の基準として提案する。

ユーザ環境に最適化されたレイアウトを生成するために ACTIVIEW はまず,

“ 幅制約 ”, “ 開発制約 ”

の 2 つの守るべきである, 最適化制約を満たしたレイアウトを元の問合せ文から生成する。この生成された複数のレイアウト結果を最終結果レイアウトの候補レイアウトとする。この候補レイアウトからユーザに提供する最終結果レイアウトを選択するために最適化レイアウトの判断基準になるのが,

“ 幅占有率目標 ”, “ 充填率目標 ”, “ 長さ目標 ”

の 3 つの目標指標である。これらのレイアウトの評価基準を定めることで, より良いレイアウトを生成する手法を提供する。

本論文の主な貢献は次の通りである。

- ACTIVIEW システムの最適化及び新しいレイアウト生成手法の提案により, ユーザ端末環境に適応した Web ビュー (結果レイアウト) を動的に提供することを可能にする。
- 2 つの制約と 3 つの目標指標の提案によって, ユーザ環境に最適なレイアウトをもつ Web ビューの提供を可能にする。
- 上述の 2 つの制約を満たした上で, ユーザに見やすいレイアウトを提供するため, 意味的な関連性を維持したレイアウトの生成手法を提案する。

1.2 論文の構成

本論文の構成は次の通りである。第 2 章では, SuperSQL システムを用いて適応型表示ビューを自動生成する方法である ACTIVIEW について述べる。第 3 章では ACTIVIEW における適応化について述べ, 第 4 章では ACTIVIEW のレイアウト変換手法を, 第 5

章ではこの変換手法の適用戦略による適応型表示ビューの生成に関して述べる．提案するレイアウト変換手法の実験と検討，考察を第 6 章で述べ，第 7 章では，関連研究の紹介とこれら従来研究と比較した本研究の特徴を述べる．最後に第 8 章で結論を述べる．

第 2 章

ACTIVIEW の実現

ACTIVIEW に関して述べる前に、その基盤技術である SuperSQL について述べる。

2.1 SuperSQL の概要

あらゆる出版メディアは、Structure (固有の構成子) と Value (データ/コンテンツ) 構造をもっている。この構造は、一般に関係データベーステーブルがもつ平坦な構造とは異なっている。したがって、関係データベースシステムからの検索結果を SuperSQL システムを用いて電子文書に変換する場合には、これらの構成子や構造を付け加えなければならない。SuperSQL によって定義された問合せ文を基に実際のメディアを生成する SuperSQL システムは、図 2.1 のように 3 つの部分から成る Trinity Data Model[9] を基にデータベース出力処理を行う。

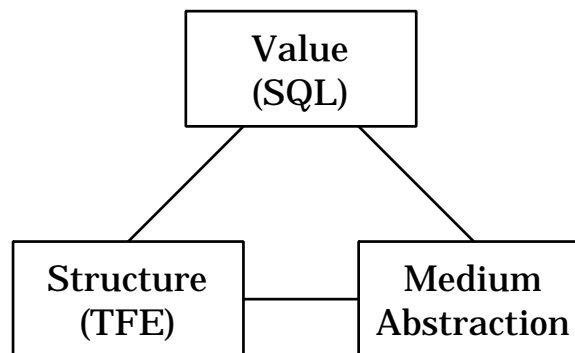


図 2.1: Trinity データモデル

本論文では、SuperSQL システムによって生成されるメディアをターゲットメディアと呼び、次のように定義する。

定義 2.1 (ターゲットメディア)

SuperSQL システムによって支援するメディアのうち、SuperSQL の問合せ文で指定されているメディア、すなわち関係データベースの検索結果から生成したいメディアを Web 開発者が指定し、この指定されたメディアがターゲットメディアとなる。

Trinity Data Model の 3 つの部分について述べる。

Value : 関係データベースからの問合せ結果で、平坦な構造をもつ検索結果。SuperSQL の問合せ文では、検索結果を用いて生成したい電子文書 (メディア)

を指定することができる¹。SuperSQL の問合せ文で定義したターゲットメディア向けの構造に変換する前、すなわち構造化を行っていないものである。

Structure：関係データベースからの平坦な構造の出力結果を SuperSQL の問合せ文の TFE 定義に従い構造化する機能。メディア依存的な構成子や装飾子の処理を Medium Abstraction に分けることで各ターゲットメディア別に存在したメディア構造生成部を 1 つに統合する事が可能になる。

Medium Abstraction：SuperSQL システムによって生成するターゲットメディアに依存する構成子と装飾子の定義

SuperSQL の問合せ文は、SQL の SELECT 句を“ GENERATE medium TFE ”の構文をもつ GENERATE 節で置き換えたものである。次に SuperSQL の問合せ文の文法を EBNF[17] を用いて記述する。FROM 句と WHERE 句の記述は標準 SQL の記述に従う。

```
SuperSQL Query = generate , [ from ] , [ where ] ;
generate       = “ GENERATE ” , medium , TFE ;
medium        = “ ACTIVIEW ” | “ XML ” | “ HTML ”
               | “ PDF ” | “ EXCEL ” | “ SWF ”
               | “ X3D ” ;
```

medium は、SuperSQL システムを用いてデータベースからの検索結果から生成可能なターゲットメディアの種類である。現在、Java 版の SuperSQL システムで支援可能なメディアを示している。

次に、TFE (結合子と反復子と装飾子) について簡単に述べる。

2.1.1 TFE (Target Form Expression) とは

本論文では、SuperSQL システムで生成する Web ビューの構造について、レイアウト、結果レイアウト、候補レイアウト、最終結果レイアウトという用語を使っている。まずそれぞれの用語を次のように定義する。

¹電子文書以外にも SuperSQL が支援するアプリケーションの指定が可能である。ACTIVIEW はターゲットメディアであるが、電子文書の種類ではない。このように、他の開発者が開発したアプリケーションを SuperSQL システムに統合することが可能である。

定義 2.2 (レイアウト)

SuperSQL システムによって定義する構造，すなわち関係データベースの検索結果から SuperSQL の問合せ文の定義に従った構造（生成されるテーブルの構造）を本論文ではレイアウトと呼ぶ．

定義 2.3 (初期レイアウト)

Web ビューの開発者が定義した元の問合せ文に従って生成されるオリジナルレイアウトを初期レイアウトと呼ぶ．

定義 2.4 (結果レイアウト)

関係データベースからの検索結果を SuperSQL システムによって構造化した結果を結果レイアウトと呼ぶ．

定義 2.5 (最終結果レイアウト)

SuperSQL システムによって生成されるレイアウトのうち，ユーザに最終的に提供される結果レイアウトを最終結果レイアウトと呼ぶ．

定義 2.6 (候補レイアウト)

ユーザ環境に最適化されたレイアウトを提供するために，SuperSQL システムによって生成される複数の結果レイアウトのうち，最終結果レイアウトの候補になるレイアウトを候補レイアウトと呼ぶ．

定義 2.7 (Web ビュー)

SuperSQL システムによって，関係データベースの検索結果から生成される Web コンテンツ，すなわち最終結果レイアウトによる結果 HTML 文書を意味する．関係データベースの検索結果を元にした Web コンテンツであるため，Web ページとは異なる Web ビューという用語を定義する．

TFE は，SuperSQL で構造化を定義するために導入したもので，定義は次のようになる．

定義 2.8 (TFE)

SuperSQL システムによって生成される結果レイアウトの構造を定義する構造化演算子である．SQL のターゲットリストを拡張したもので，TFE 特有のオペレータ（結合子と反復子）を用いてレイアウトを定義する．

TFE = connector | repeater ;

通常の SQL に TFE を導入することによって元の問合せ文を定義する開発者は、自分の意図通りに関係データベースからの検索結果を構造化することができる。SuperSQL システムは、TFE を用いた構造定義に従い、関係データベースからの平坦な構造から開発者が定義したレイアウト構造へのレイアウト変換を行う。TFE 特有のオペレータである結合子と反復子はそれぞれレイアウトの次元に対応している。

HTML 文書を生成する場合、1, 2 次元は、表の行と列に対応し、3 次元はハイパーリンクに対応する。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。

2.1.2 結合子

結合子 (Connector) は次のように定義される。

定義 2.9 (結合子)

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。結合子は、水平結合子 (,) , 垂直結合子 (!) , 深度方向への結合子 (%) がある。

connector expression = (attribute | TFE) , [decorator] ,
connector ,
(attribute | TFE) , [decorator] ;
connector = “ , ” | “ ! ” | “ % ” ;
attribute = 通常の SQL 文の SELECT 句の属性 ;

表 2.1 にその種類と意味、略記法を示す。なお、表中の A と B はオペランドであり、属性または任意の TFE である。

図 2.2 に結合子を使った SuperSQL 問合せ文と、この問合せ文による結果テーブルの例を示す。

- 水平結合子 (,) : 両オペランドのデータを横に連結して出力する。図 2.2 の A
- 垂直結合子 (!) : 両オペランドのデータを縦に連結して出力する。図 2.2 の B
- 深度方向への結合子 (%) : 両オペランドのデータを深度方向に連結して出力する。図 2.2 の C

表 2.1: 結合子の種類と意味

次元	結合子	名前	問合せ文の表記
1	,	水平結合子	A, B
2	!	垂直結合子	$A!B$
3	%	深度方向への結合子	$A \% B$

	問合せ文の例	結果テーブル			
A	GENERATE Actiview p.team , p.name FROM player p	<table border="1"> <tr> <td>Japan</td> <td>Koji NAKATA</td> </tr> </table>	Japan	Koji NAKATA	
Japan	Koji NAKATA				
B	GENERATE Actiview p.team ! p.name FROM player p	<table border="1"> <tr> <td>Japan</td> </tr> <tr> <td>Koji NAGATA</td> </tr> </table>	Japan	Koji NAGATA	
Japan					
Koji NAGATA					
C	GENERATE Actiview p.team % p.name FROM player p	<table border="1"> <tr> <td>Japan</td> <td rowspan="2">Koji NAKATA</td> </tr> <tr> <td>Hyperlink</td> </tr> </table>	Japan	Koji NAKATA	Hyperlink
Japan	Koji NAKATA				
Hyperlink					

図 2.2: 各結合子の問合せ文の例と結果テーブル

ターゲットメディアが ACTIVIEW になっている場合は HTML 文書を出力するため、深度方向への結合子はハイパーリンク機能になる²。

2.1.3 反復子

反復子 (Repeater) は次のように定義される。

定義 2.10 (反復子)

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。大括弧と連結子の組み立てによる、水平反復子 ([],) , 垂直反復子 ([] !) , 深度方向への反復子 ([] %) がある。

²他のメディアへの応用も可能だが、現時点で ACTIVIEW が支援しているメディアは HTML 文書のみである。

repeater = “ [”, (attribute | TFE) , “] ”, connector , [decorator] ;

表 2.2 に反復子の種類と意味，略記法を示す．結合子と同様，表中の A は属性か任意の TFE である．

表 2.2: 反復子の種類と意味

次元	反復子	名前	問合せ文の表記
1	[],	水平反復子	[A],
2	[]!	垂直反復子	[A]!
3	[]%	深度方向への反復子	[A]%

たとえば，

[選手の国籍 , 選手の名前]!

と記述することで，横方向に連結された選手の国籍とその選手の名前をインスタンス数だけ縦方向に反復して出力する．

また反復子は単に反復構造を指定するだけでなく，入れ子構造の生成ができる．関係データベースからの検索結果から，反復して現れる属性とこの属性を含んで反復しない属性間の関連を指定することで，反復して現れる属性にネストされた構造のレイアウト定義ができる．

たとえば，

[チーム名]! , [選手の名前]! , [ポジション]!

とすると，単に各々の一覧が表示されるだけ（図 2.3）で互いの関連は失われるが，

[チーム名 !
 [選手の名前 , ポジション]!
]!

と反復子を入れ子状にすることで，そのチームにおける選手の名前とポジション一覧が図 2.4 のようにネスト構造で表示される．

Japan	Koji NAKATA	GK
England	Shinji ONO	DF
...	Yasuhito ENDO	MF
	...	FW
	David BECKHAM	
	Michael OWEN	
	...	

図 2.3: 単に各々の一覧が表示されるレイアウトの例

Japan	Koji NAKATA	DF
	Shinji ONO	MF
	Yasuhito ENDO	MF

England	David BECKHAM	MF
	Michael OWEN	FW

...

図 2.4: ネストされた構造のレイアウトの例

	問合せ文の例	結果テーブル								
A	GENERATE <i>Actiview</i> [p.name], FROM player p	<table border="1"> <tr> <td>Koji NAKATA</td> <td>Shinji ONO</td> <td>...</td> </tr> </table>	Koji NAKATA	Shinji ONO	...					
Koji NAKATA	Shinji ONO	...								
B	GENERATE <i>Actiview</i> [p.name]! FROM player p	<table border="1"> <tr><td>Koji NAKATA</td></tr> <tr><td>Shinji ONO</td></tr> <tr><td>...</td></tr> <tr><td>KAWAGUCHI</td></tr> </table>	Koji NAKATA	Shinji ONO	...	KAWAGUCHI				
Koji NAKATA										
Shinji ONO										
...										
KAWAGUCHI										
C	GENERATE <i>Actiview</i> [p.team , [p.name]!]! FROM player p	<table border="1"> <tr> <td rowspan="3">Japan</td> <td>Koji NAKATA</td> </tr> <tr> <td>Shinji ONO</td> </tr> <tr> <td>...</td> </tr> <tr> <td>England</td> <td>BECKHAM</td> </tr> <tr> <td>...</td> <td>...</td> </tr> </table>	Japan	Koji NAKATA	Shinji ONO	...	England	BECKHAM
Japan	Koji NAKATA									
	Shinji ONO									
	...									
England	BECKHAM									
...	...									

図 2.5: 各反復子の問合せ文の例と結果テーブル

図 2.5 に反復子を使った SuperSQL 問合せ文と、この問合せ文による結果テーブルの例を示す。

- 水平反復子 ([],) : オペランドのデータがある限り、そのデータを横に繰り返し結合する。図 2.5 の A
- 垂直反復子 ([]!) : オペランドのデータがある限り、そのデータを縦に繰り返し結合する。図 2.5 の B
- 深度方向への反復子 ([]%) : オペランドのデータを深度方向に繰り返し結合する。

図 2.5 の C は垂直反復子と水平結合子を組み合わせて用いる例である。このような反復子のネストによってグルーピングを直感的に指定できる。

2.1.4 装飾子

SuperSQL には、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する装飾子がある。装飾子は、次のように定義される。

```

decorator          = “ @ ”, “ { ”, decorator expression ,
                    { “ , ”, decorator expression } , “ } ” ;
decorator expression = item name , “ = ” , value ;
item name          = “ bgcolor ” | “ width ” | “ height ”
                    | “ font color ” | “ font size ” | “ align ”
                    | “ valign ” | “ divalign ” | ... ;

```

定義のように、装飾子 (decorator) は問合せ文に定義される属性名、すなわち関係データベースのカラム名や文字列定数、あるいは反復子に対して指定する。装飾指定式 (decorator expression) の定義のように、“項目名 = 値”に従う指定によって生成する文書に様々な装飾をする。たとえば、関係データベースからの検索結果のうち、チーム名 (team.name) を検索した結果値の表すセルの背景を赤にして、セル内の横幅を 120pixel にしたい場合は次のように記述する。

```
team.name@{bgcolor=red, width=120}
```

項目名 (item name) はターゲットメディアによって異なる指定なので、ターゲットメディアの追加とともに増える。

2.1.5 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、特定の処理を行う機能である。

SuperSQL における関数呼出しの構文は、次のように定義される。

```

SuperSQL function = function name , ( “ ( ”, TFE , option list “ ) ”
                    | “ [ ”, TFE “ ] ” ) ;
function name     = “ imagefile ” | “ link ” | “ sum ” | “ max ”
                    | “ min ” | “ avg ” | “ count ” ;
option list       = path , | query file , “ , ”, TFE ;
path              = “ path= ”, ( 絶対パス指定 | 相対パス指定 ), “ ” ;

```

現在の SuperSQL システムで利用可能な関数 (function name) は、問い合わせ結果中に画像を表示する imagefile 関数、HTML 文書出力においてハイパーリンクを動的に構築する link 関数と集約関数 (sum, max, ...) がある。option list は、関数の実行に必要な条件やディレクトリパス等の指定を“,”で区切ったリストである。

たとえば、link 関数を用いたリンク生成の定義は次のようになる。

```
link ( moviefile.title, file=" Q4-2.sql ", att=moviefile.id )
```

この定義によると、映画のタイトルに対し、映画の詳細情報ページへリンクされる Web ビューが生成される。SuperSQL システムは、file で指定した問合せ文ファイル名と“ att ”で指定した属性の値を用いてリンク先の URL を生成する。

2.1.6 文字列定数

文字列定数は、問合せ文による関係データベースからの検索結果ではなく、開発者の指定によって SuperSQL 問合せ文に含める文字列である。テーブルを生成する際、題目などを指定したい場合使われる。SuperSQL における文字列定数は、次のように定義する。

```
constant string = “ ”, string , “ ”
```

string は元の問合せ文を作成する開発者が定義する文字列である。2.1.4 項の装飾子で定義したように文字列定数にも、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指定が可能である。次の例は、文字列定数“ 選手の名前 ”が現れるセルの背景色が赤になるように定義したものである。

```
“ 選手の名前 ”@{bgcolor = red}
```

次のように選手の名前を検索する問合せ文に文字列定数“ 選手の名前 ”を赤いフォントで表すように指定したとすると、

```
“ 選手の名前 ”@{font color = red} ! [ player.name ] !
```

生成されるテーブルは図 2.6 のようになる。

選手の名前
Yoshikatsu KAWAGUCHI
Yasuhito ENDO
Hidetoshi NAKATA
...

図 2.6: 文字列定数を用いた例

2.2 ACTIVIEW

現在，デスクトップコンピュータ用に作成した Web ページをユーザ表示画面のサイズに合わせて変換するために様々な研究が行われている．しかし，現在までに行われている研究では Web コンテンツを縮小や要約したり，リンクを用いたり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，Web ページのレイアウト構造の変換は対象にしていない [18, 19, 20, 21, 22, 23]．多様な端末にエンベデッドされたアプリケーションからの Web アクセスも，特定の端末に限られている [24, 25, 26, 27, 28]．このように従来の Web 開発は元となる Web ページやユーザが利用している端末に依存した開発が続いている（図 2.7）．これは Web 開発者にとって負担になるものであり，多様化しているユーザ環境とともに変化するユーザの要求に動的に対応することは困難である．

一般的に，表³を Web 上で閲覧する際，表示画面より大きな表が表示されると非常に見にくいだけでなく，縦横にスクロールする不便が生じる．また，携帯端末においては縦へのスクロールが主流であり，横に広い表は望ましくない．

様々なユーザ環境，すなわち変化する端末機に動的に対応するため，ACTIVIEW は生成される Web ビューの構造定義に注目し，Web ビューのレイアウトを変換することで様々なユーザ画面表示に対応する手法を提案している．ACTIVIEW における適応化の基本概念を図 2.8 に示す．Web ページを提供する際，ACTIVIEW の導入によって，SuperSQL によって定義されている元のテーブル（Original View）から，ユーザ環境に適応した様々な型のテーブル（Adapted View）を自動的に生成する．

ACTIVIEW の問合せ文は，ユーザが ACTIVIEW によって生成された Web ビューをアクセスする際に受け取ったユーザ表示画面サイズの情報を基に，適応化処理部によって，ユーザ表示環境の画面サイズに適応するように変換される．ACTIVIEW の問合せ文は従来の SuperSQL 問合せ文と同じで，ターゲットメディアが ACTIVIEW になっているだけである．

本論文で提案する新しい ACTIVIEW での内部変換手法は次の通りである．

- (1) ユーザが ACTIVIEW によって生成された Web ビューにアクセスする時にユーザ表幅の値（表示画面サイズ）を受け取る．
- (2) 初期のレイアウトを定義した ACTIVIEW の問合せ文⁴から表の構造に関する情報と，装飾情報の“幅”に関する情報が混在する式（レイアウト式とサイズ情報）を生成し，表の幅を計算する．

³関係データベースからの検索結果から生成されたビュー

⁴初期のレイアウトは関係データベースの結果からデスクトップコンピュータ用向けに作成した Web ビューであるため，ユーザ表示画面のサイズに合わせる必要がある．

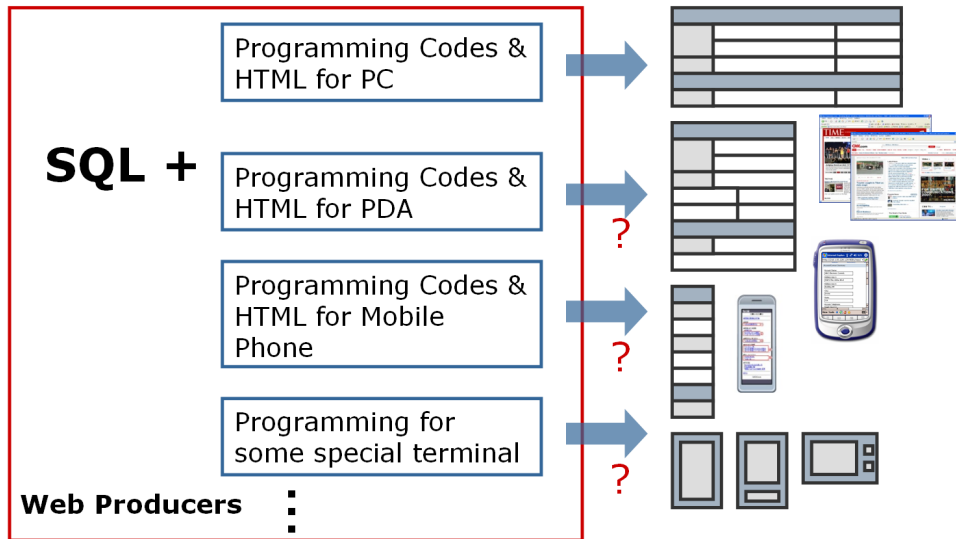


図 2.7: 従来の Web 開発

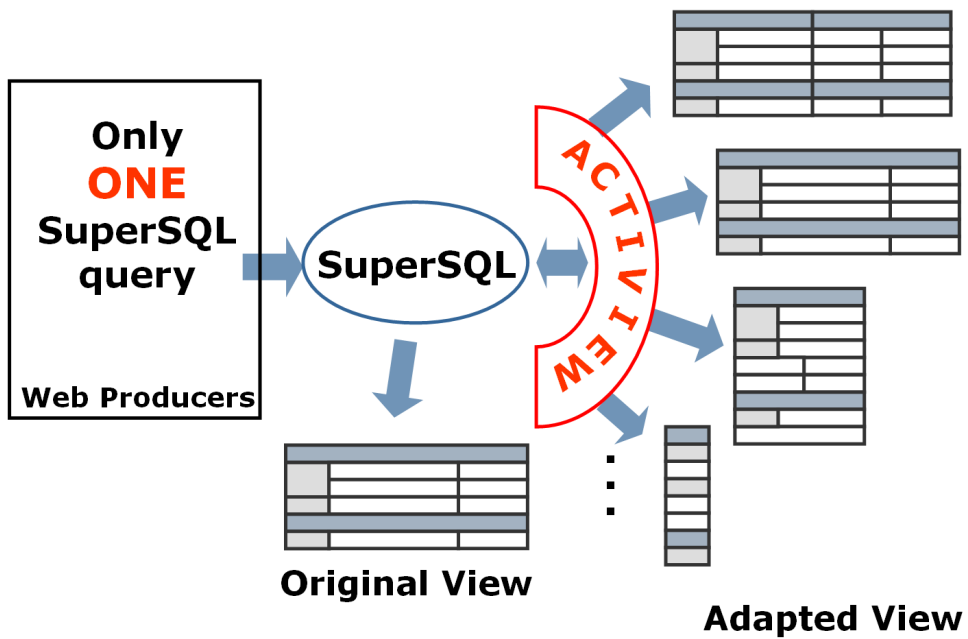


図 2.8: ACTIVIEW によるレイアウト生成の概念

- (3) 生成されるレイアウトが本論文で提案する制約と目標（第 5 章）を満たしたレイアウトではない場合は，提案手法に従ってレイアウト構造の変換を行う．
- (4) 各制約と目標を満たし，ユーザ表示画面に適応したレイアウトが生成されるまで，(2) (3) を繰り返す．

本論文で提案する制約と目標に関しては，第 5 章で詳細を述べる．

初期の ACTIVIEW システムは Lisp ベースで開発された．しかし，SuperSQL のシステムが Java ベースで再設計されたこととともに ACTIVIEW システムも作り直す必要が生じた．さらに，本論文で提案するレイアウト変換手法の実現と，より効率的な ACTIVIEW システムに対する要求から ACTIVIEW システムの再設計を行った．

2.3 システム構成

全般的なシステムの構成を図 2.9 に示す．図 2.9 において，太線で囲まれた部分が SuperSQL 処理系であり，点線で囲まれた部分が ACTIVIEW を実現するための処理部分である．

まず，フロントエンド（Front End）がブラウザからユーザ表示画面の幅と高さを受け取り，この情報を Adaptation Processor（以下，適応化処理部）に渡す．

ユーザ表示画面というのはユーザが利用している端末機の画面ではなく，端末上でアクセスした時利用しているブラウザを意味する．そのため，同じ端末機であっても異なる画面サイズである場合や，動的に変化する場合がある．

適応化処理部は，フロントエンドが受け取ったユーザ表示幅に従って ACTIVIEW Query⁵をユーザ表示画面に収まるように書き換える処理を行う．この処理によって，ユーザ表示画面のサイズに対応する．

Adaptation Processor： 適応化処理部は開発者によって定義されている元の問合せ文から適応化処理を効率的に行うために必要な情報を取り出す．元の問合せ文から構造情報のみ取り出した構造情報リストや，各ネスト別の属性のリストなど，レイアウト変換処理に使われる情報を先に獲得する．これらの情報を基に“Convert Processor”と“Filling rate Calculator”，“Width Calculator”処理部がユーザ表示画面に適応したレイアウトを生成する．

適応化処理部の各処理部について簡単に述べる．

⁵Web ページを提供する側の開発者が指定した元の問合せ文．

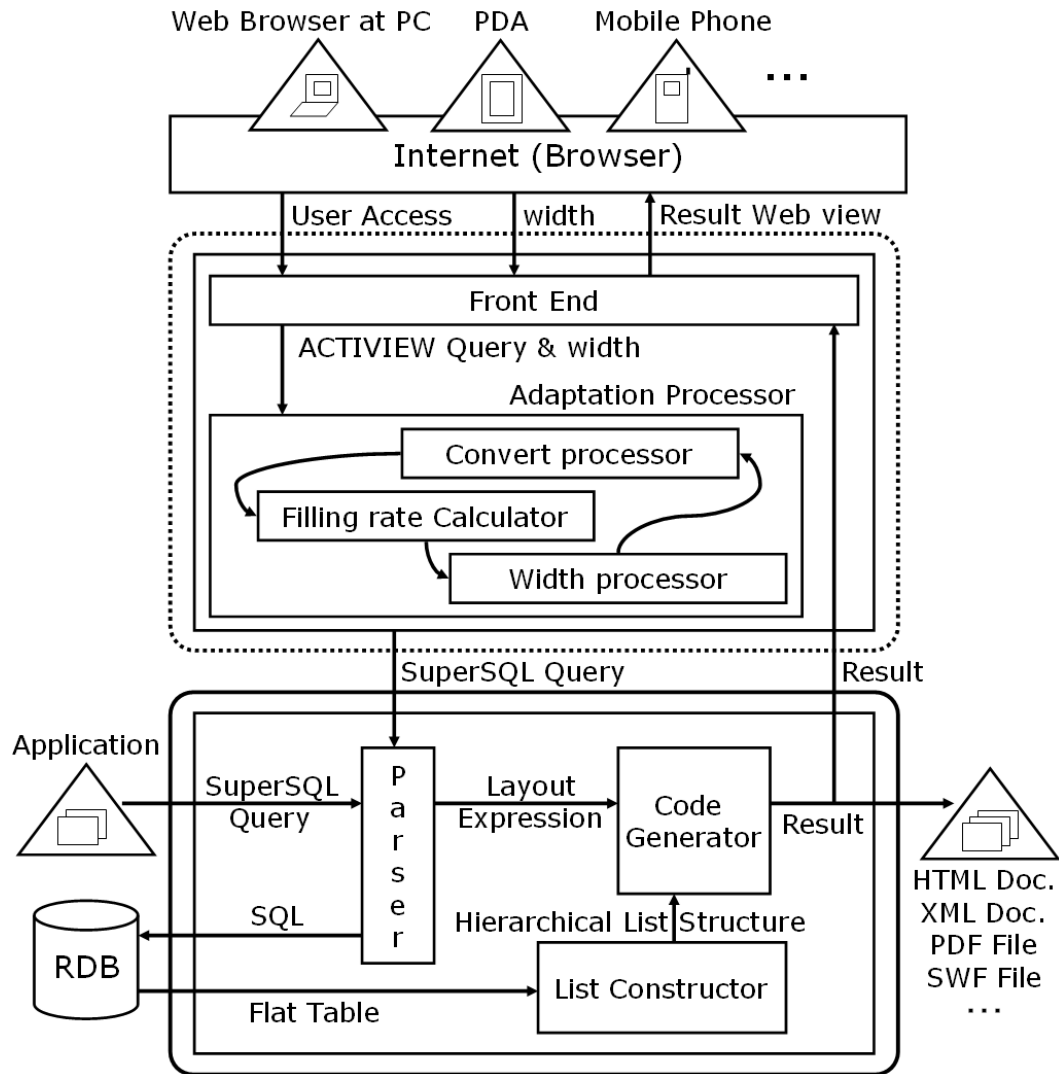


図 2.9: システムの概要図

Convert Processor : ユーザ表示画面の幅と高さを基準に、本論文で新しく提案するレイアウト変換手法(第5章で詳しく述べる.)に従ってユーザ表示画面の幅と高さに合わせた、候補レイアウトを生成する。変換を行って再構成された候補レイアウトは“ Filling rate Calculator ”と“ Width Calculator ”処理部で評価される。ユーザ表示画面に適合し、一貫性のあるレイアウトが生成されたと判断できるまでこの処理を行う。

Filling rate Calculator : “ Convert Processor ”の処理によって再構成された複数の候補レイアウトそれぞれがどれくらいユーザ表示画面を利用しているかを判断するために、ユーザ表示画面に対する候補レイアウトの利用率を計算し(計算方法は3.4節で詳しく述べる.)、利用率が高い候補レイアウトを推薦する。

Width Calculator : “ Convert Processor ”の処理によって再構成された複数の候補レイアウトのうち、幅制約(3.1節で詳しく述べる.)を満たした候補レイアウトを推薦する。

本論文では一貫性のあるレイアウトを次のように定義する。

定義 2.11 (一貫性のあるレイアウト)

関連性が高いと判断できる属性(同一中括弧内あるいは同一ネスト内の属性)間をつなげる演算子は、変換を行う際、同時に変換を行うことによって統一された構造をもつレイアウトを生成すると仮定する。このような一括変換の対象になる演算子を同時変換することによって生成されたレイアウト結果を、一貫性のあるレイアウトと呼ぶ。

適応化処理部によって書き換えた ACTIVIEW の問合せ文に基づき、SuperSQL システムは関係データベースから Web 上で利用可能な HTML 文書を生成する。

たとえば、図 2.10 のように各チームが所属するグループと各チーム名、所属している選手のポジション情報と選手個人情報を検索する通常の SQL 文(図 2.10 の上の部分)は、平坦な構造の出力結果(図 2.10 の下の部分)を返す。

図 2.10 の通常の SQL 文を SuperSQL の構造化定義に従って ACTIVIEW の問合せ文に書き直すと図 2.11 の上のような問合せ文が定義できる。この ACTIVIEW の問合せ文によって、平坦な構造の出力結果(図 2.10 の下の部分)から構造化された結果が図 2.11 の下の部分である。

ACTIVIEW におけるレイアウト適応化(第3章)や本論文で提案するレイアウト変更戦略(第5章)の説明を一貫性のあるものとするため、論文全般に使われる ACTIVIEW の問合せ文を図 2.12 のように定義する。


```
SELECT t.group, t.country, p.position ,
       p.name, p.birth, p.height, p.weight
```

GROUP	B	England	FW	Michael OWEN	...
GROUP	B	England	FW	Peter CROUCH	...
GROUP	B	England	FW	Wayne ROONEY	...
GROUP	B	England	MF	David BECHAM	...
GROUP	B	England	MF	Frank LAMPARD	...
...

図 2.10: 通常の SQL 文とその結果

```
GENERATE ACTIVIEW
[ "GROUP", t.group ,
  [ "COUNTRY"! t.country !
    [ "POSITION", p.position ,
      [ p.name , p.birth , p.height , p.weight ]!
    ]! ]! ]!
```

GROUP		POSITION		COUNTRY	Name	Birth	Height	Weight
B	FW	England (ENG)			Michael OWEN	14/12/1979	172	67
					Peter CROUCH	20/01/1981	198	69
					Wayne ROONEY	24/10/1985	178	78
	MF				David BECKHAM	02/05/1975	182	74
					Frank LAMPARD	20/06/1978	177	78

図 2.11: SuperSQL によって構造化された Web ビューの例

POSITION	NAME	DLM	BIRTHDAY	HEIGHT	WEIGHT	COUNTRY	NATIONALITY
	RICARDO		13/11/1975	184	76	Portugal (POR)	Portugal (POR)
	RICARDO		11/02/1976	188	80	Portugal (POR)	Portugal (POR)
	RICARDO		11/02/1976	188	80	Portugal (POR)	Portugal (POR)
	RICARDO		22/02/1977	176	74	Portugal (POR)	Portugal (POR)
	HUGO WANA		15/01/1983	179	77	Portugal (POR)	Portugal (POR)
	MANICHE		11/11/1977	187	86	Portugal (POR)	Portugal (POR)
	PETIT		25/09/1976	173	73	Portugal (POR)	Portugal (POR)
	TILAO		02/05/1981	183	83	Portugal (POR)	Portugal (POR)
	TILAO		02/05/1981	183	83	Portugal (POR)	Portugal (POR)
	MANE ANKLOVSKI		06/05/1977	184	79	Czech Republic (CZE)	Czech Republic (CZE)
	MANE ANKLOVSKI		06/05/1977	184	79	Czech Republic (CZE)	Czech Republic (CZE)
	Martin BRANEK		25/05/1979	183	77	Czech Republic (CZE)	Czech Republic (CZE)
	Pavel BABES		19/01/1976	185	78	Czech Republic (CZE)	Czech Republic (CZE)
	Tomaz UPALUS		24/03/1978	186	76	Czech Republic (CZE)	Czech Republic (CZE)
	Tomaz UPALUS		24/03/1978	186	76	Czech Republic (CZE)	Czech Republic (CZE)
	BRANKO		30/03/1975	202	100	Czech Republic (CZE)	Czech Republic (CZE)
	LADY SKANO		01/02/1977	175	70	Czech Republic (CZE)	Czech Republic (CZE)
	MANE HENZ		04/03/1977	188	75	Czech Republic (CZE)	Czech Republic (CZE)
	MANE HENZ		04/03/1977	188	75	Czech Republic (CZE)	Czech Republic (CZE)
	MARZ BAROS		28/10/1981	181	77	Czech Republic (CZE)	Czech Republic (CZE)
	MARZ BAROS		28/10/1981	181	77	Czech Republic (CZE)	Czech Republic (CZE)
	Jozef BLAZEK		30/06/1975	187	82	Czech Republic (CZE)	Czech Republic (CZE)
	Jozef BLAZEK		30/06/1975	187	82	Czech Republic (CZE)	Czech Republic (CZE)
	Petr Cech		20/05/1982	197	87	Czech Republic (CZE)	Czech Republic (CZE)
	Petr Cech		20/05/1982	197	87	Czech Republic (CZE)	Czech Republic (CZE)
	Davik JARDUM		17/05/1979	172	69	Czech Republic (CZE)	Czech Republic (CZE)
	Davik JARDUM		17/05/1979	172	69	Czech Republic (CZE)	Czech Republic (CZE)
	Tomaz MALISEK		27/11/1979	182	77	Czech Republic (CZE)	Czech Republic (CZE)
	Tomaz MALISEK		27/11/1979	182	77	Czech Republic (CZE)	Czech Republic (CZE)
	Radostav Kovac		15/01/1973	189	83	Czech Republic (CZE)	Czech Republic (CZE)
	Radostav Kovac		15/01/1973	189	83	Czech Republic (CZE)	Czech Republic (CZE)
	Pavel Nedved		30/06/1972	175	73	Czech Republic (CZE)	Czech Republic (CZE)
	Pavel Nedved		30/06/1972	175	73	Czech Republic (CZE)	Czech Republic (CZE)
	Karel Poborsky		30/03/1972	185	82	Czech Republic (CZE)	Czech Republic (CZE)
	Karel Poborsky		30/03/1972	185	82	Czech Republic (CZE)	Czech Republic (CZE)
	Jiri Stastny		22/05/1962	185	82	Czech Republic (CZE)	Czech Republic (CZE)
	Jiri Stastny		22/05/1962	185	82	Czech Republic (CZE)	Czech Republic (CZE)
	Jan Kocul		14/03/1981	181	77	Czech Republic (CZE)	Czech Republic (CZE)
	Jan Kocul		14/03/1981	181	77	Czech Republic (CZE)	Czech Republic (CZE)
	Armed Issah		24/05/1982	186	80	Ghana (GHA)	Ghana (GHA)
	Armed Issah		24/05/1982	186	80	Ghana (GHA)	Ghana (GHA)
	Samuel Adu-Poku		09/09/1976	179	75	Ghana (GHA)	Ghana (GHA)
	Samuel Adu-Poku		09/09/1976	179	75	Ghana (GHA)	Ghana (GHA)
	John Pantisil		15/09/1981	173	73	Ghana (GHA)	Ghana (GHA)
	John Pantisil		15/09/1981	173	73	Ghana (GHA)	Ghana (GHA)
	Haris Sappi		26/10/1982	188	81	Ghana (GHA)	Ghana (GHA)
	Haris Sappi		26/10/1982	188	81	Ghana (GHA)	Ghana (GHA)
	Haris Sappi		25/06/1976	179	68	Ghana (GHA)	Ghana (GHA)
	Haris Sappi		25/06/1976	179	68	Ghana (GHA)	Ghana (GHA)
	Hamid Mohamed		10/12/1983	179	76	Ghana (GHA)	Ghana (GHA)
	Hamid Mohamed		10/12/1983	179	76	Ghana (GHA)	Ghana (GHA)
	Enmanuel Pappoe		03/03/1981	177	76	Ghana (GHA)	Ghana (GHA)
	Enmanuel Pappoe		03/03/1981	177	76	Ghana (GHA)	Ghana (GHA)
	Enmanuel Pappoe		03/03/1981	177	76	Ghana (GHA)	Ghana (GHA)
	Enmanuel Pappoe		03/03/1981	177	76	Ghana (GHA)	Ghana (GHA)
	Mani Tache-Adesani		15/02/1977	173	76	Ghana (GHA)	Ghana (GHA)
	Mani Tache-Adesani		15/02/1977	173	76	Ghana (GHA)	Ghana (GHA)

図 2.13: ACTIVIEW の問合せ文 (図 2.12) から生成される Web ビュー

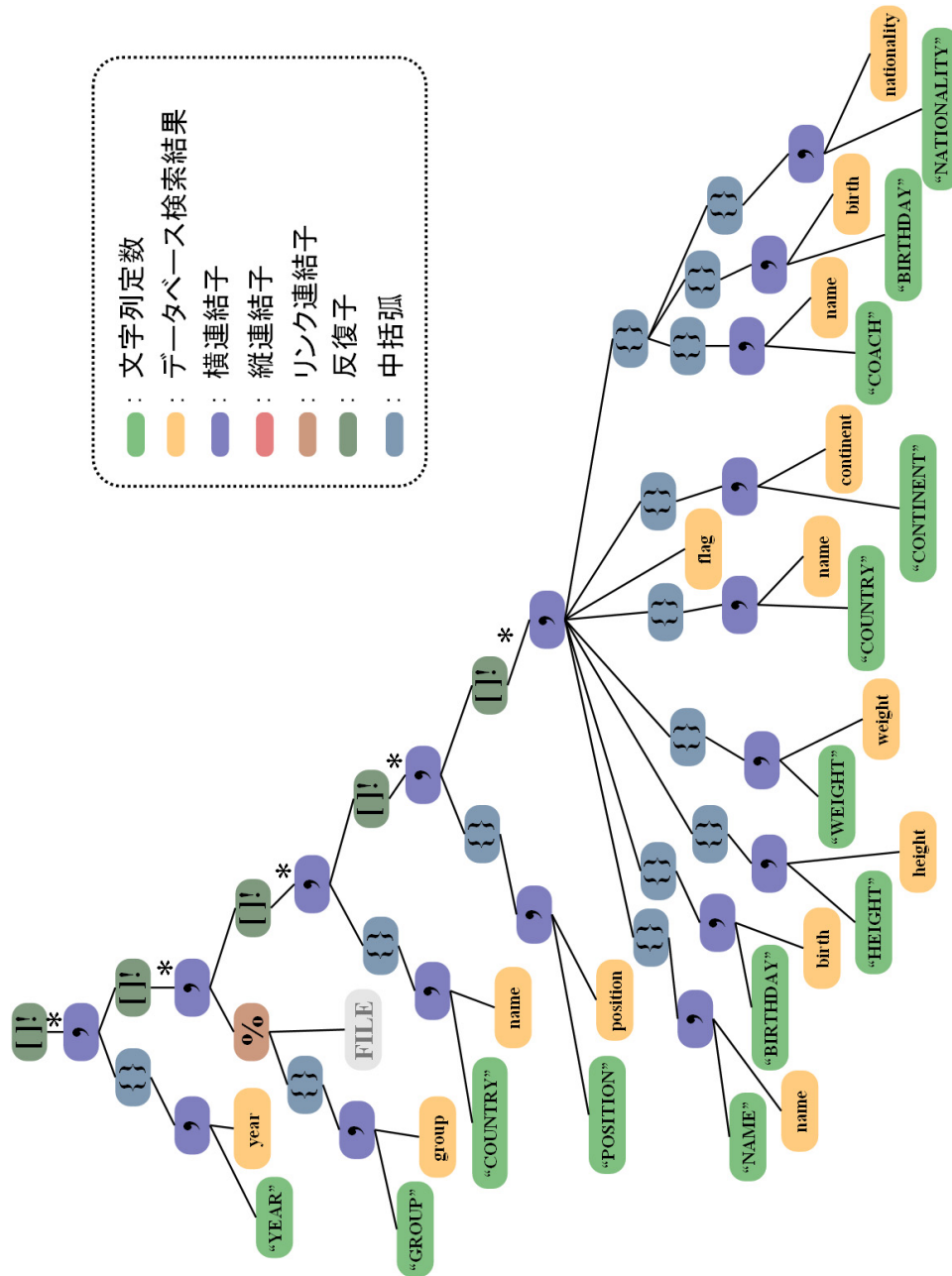


図 2.14: レイアウト情報の木構造表現

第 3 章

ACTIVIEW における適応化

ユーザ表示画面に適応化された Web ビューは、再構成されるレイアウトがユーザ表示画面の幅以内であるなどの“条件を満たしたレイアウト”，元の間合せ文に開発者が指定したレイアウトの構造定義による“開発者の意図の反映”，異なるユーザ環境に対する“ビュー表示領域の有効利用”，再構成されるレイアウトが“ユーザに使いやすい”レイアウトであることなど、様々な角度から考えることができる。

ユーザ表示画面のサイズに最も適応化されたレイアウトを求めるには、いくつかの条件を考え、それらを満たしたレイアウトの中で、どのレイアウトを提供するかを決定する必要がある。

本研究では、適応化されたレイアウトの指標として、

- 幅制約：ユーザ表示画面の幅サイズに対する制約
- 開発者制約：ACTIVIEW の間合せ文の定義による制約

という守るべき 2 つの制約と

- 幅占有率目標：効率的な幅の利用
- 充填率目標：効率的な空間の利用
- 長さ目標：極端な長さを避ける。

という 3 つの目標を提案する。

ACTIVIEW システムでは変換された複数のレイアウトのうち、制約を満たしたレイアウトだけを候補レイアウトとして評価関数によって評価し、ユーザ表示画面に最も適応したレイアウトの生成を目指す。各評価関数の結果値の比較を行い、最も評価が良い候補レイアウトを最終結果レイアウトとしてユーザに提供する。

3.1 幅制約

ユーザ表示画面に適応したレイアウトの指標の 1 つである幅制約を、次のように定義する。

定義 3.1 (幅制約)

構成されたレイアウトがユーザ表示画面の幅より狭い幅をもつ条件を幅制約とする。

幅制約は最も基本になる制約である。ACTIVIEW システムで提供される最終結果レイアウトはユーザ表示画面上に収めることを目指す。したがって、ACTIVIEW システ

ムが生成した SuperSQL の問合せ文によって再構成されたレイアウトの中で、ユーザ表示画面の幅に収まるレイアウトを、幅制約を満たしたレイアウトとする。

幅制約条件の充足可否を判断するため、関数 $RW(q)$ を定義する。 q を問合せ文、 n を属性の個数、 e_i を属性、 i を問合せ文に現れる属性の順序を表す番号、 $W(e_i)$ を属性 e_i の幅、 $W(U)$ をユーザ表示画面の幅とする。 次の式に従って、問合せ文に指定された各属性の幅の合計がユーザ表示画面の幅値、 $W(U)$ より小さい幅値をもつレイアウトが幅制約 $RW(q)$ を満たしたレイアウトになる。

$$RW(q) = \begin{cases} true & \text{if } \sum_{i=1}^n W(e_i) \leq W(U), \\ false & \text{if } \sum_{i=1}^n W(e_i) > W(U), \\ exception & \text{if } \exists i W(e_i) > W(U). \end{cases} \quad (3.1)$$

$$W(e_i) = \begin{cases} width(e_{max}) & \text{if 全ての結合子 = 縦連結子,} \\ sum(e_i, e_{i+1}) & \text{if 結合子 = 横連結子,} \\ max(e_i, e_{i+1}) & \text{if 結合子 = 縦連結子.} \end{cases} \quad (3.2)$$

すべての連結子が縦連結子であると、レイアウトの幅値は問合せ文にある属性中で一番幅広い属性 (e_{max}) の値になる。

本論文で提案するレイアウト変換戦略 (第 5 章で詳しく述べる。) に従って変換された結果レイアウトの幅を計算し、条件を満たすまでレイアウトの変換を行う。幅制約を満たしたレイアウトは最終結果レイアウトの候補レイアウトになる。しかしながら、属性の中に、ユーザ表示画面より広い幅をもつ属性が存在する場合があります。この場合は、レイアウトを変換してもユーザ表示画面の幅に収まるレイアウトは生成できない。本論文で再設計した ACTIVIEW システムでは、これは例外としてユーザ表示画面の幅に合わせて属性の幅値の変換を行う。幅制約を適用することによって横スクロール操作が無くなる。これは、携帯端末など非常に幅の狭い画面でのユーザ操作の便利性を高める。

図 2.14 のレイアウト情報とテーブルの幅 (width 属性) に関する装飾情報を合わせて得られる木構造表現を図 3.1 に示す。図 3.1 はレイアウトの変換によって、レイアウトの幅値が変わる一例である。

ユーザ表示画面の幅が 1,200pixel であると、図 3.1 の左側に示した元のレイアウト幅 (2,690pixel) では幅制約を満たせない。幅制約を満たすために、木構造の 3 カ所で横連結子から縦連結子に変換を行ったレイアウト (右側) の幅値は 1,150pixel になり、幅

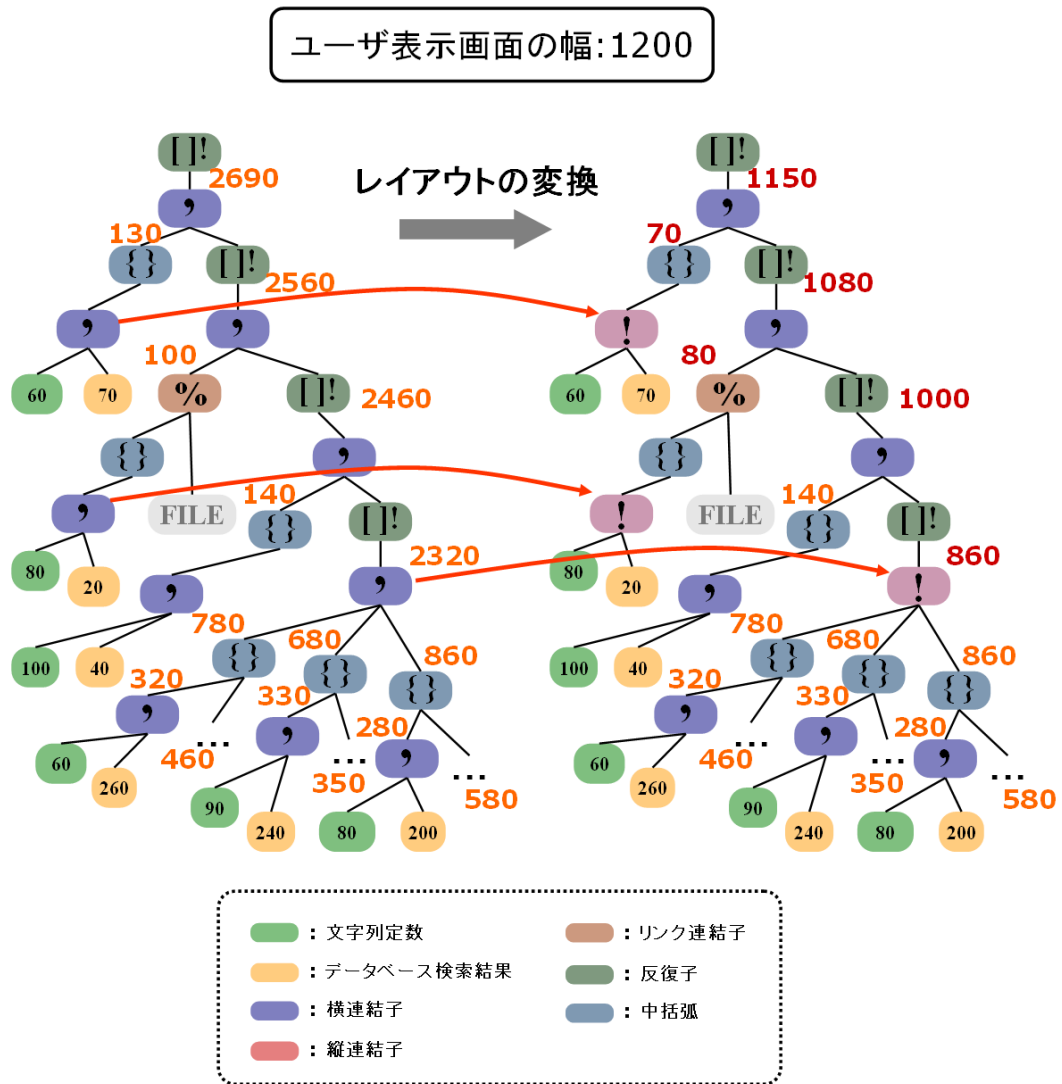


図 3.1: レイアウトーサイズ式の木構造表現

制約を満たしたレイアウトになる．実際は 860pixel の幅になるネストでは 780pixel と 680pixel と 860pixel の幅をもつ 3 つの中括弧の間の 2 カ所が変換されるので，全部 4 カ所での変換が行われる．

表 3.1 に示す計算方法を用いてレイアウト式から生成されるレイアウトの幅が求まる¹． E_i と E_j は属性または任意の TFE である．また， $sum()$ ， $max()$ は関数であり，次の計算を行う．

$sum()$ ： 検索結果が横に連結されるレイアウトを生成する場合であり () 内の値の合計を計算する．

$max()$ ： 検索結果が縦に連結されるレイアウトを生成する場合であり () 内の最大値を返す．

表 3.1: 結合子ごとの計算

次元	結合子	計算法
1	E_i, E_j	$(sum\ width(E_i)\ width(E_j))$
2	$E_i ! E_j$	$(max\ width(E_i)\ width(E_j))$
3	$E_i \% E_j$	$width(E_i)$

この計算方法を図 3.1 に適用すると

- レイアウト変換の適用前：

$$\begin{aligned} & (sum (sum 60 70) (sum (sum 80 20) \\ & (sum (sum 100 40) (sum \\ & sum ((sum 60 260) 460) sum ((sum 90 240) 350) \\ & sum ((sum 80 300) 580))))) \\ & = 2,690 \end{aligned}$$

- レイアウト変換の適用後：

$$\begin{aligned} & (sum (max 60 70) (sum (max 80 20) \\ & (sum (sum 100 40) (max \\ & sum ((sum 60 260) 460) sum ((sum 90 240) 350) \\ & sum ((sum 80 300) 580))))) \\ & = 1,150 \end{aligned}$$

と計算でき，レイアウトの変換後の幅が小さくなっていることが分かる．

¹File はハイパーリンク先のファイル， $width()$ は指定された TFE の表示幅を求める関数

3.2 開発者制約

ユーザ表示画面サイズの動的な変化に対して、データを提供する側、すなわち Web ページのデザイナー（開発者）が、元の問合せ文に意図的に与えたレイアウトとは異なるものが提供されることは望ましくない。そこで、開発者制約を次のように定義する。

定義 3.2 (開発者制約)

元の問合せ文における構造化定義による開発者の意図の反映を開発者制約とし、レイアウトの変換を行う際に尊重する。

開発者制約では、まず開発者が与える問合せ文をどこまでレイアウトに関する制約ととらえるかが問題となる。開発者が与えた入れ子構造をシステムが変更してしまっても、意味的に開発者が意図していないレイアウトができあがってしまう可能性がある。

開発者は元になる問合せ文を作成する段階で、次の 3 つの事項をレイアウトに盛り込むことで、レイアウトの構造に自分の意図を反映できる。

- 属性の順序
- ネスト（入れ子構造）
- グループینگ

属性の順序は、開発者によって定義される元の問合せ文に現れる属性の順序を意味する。問合せ文に定義される属性の順序は開発者の意図的な指定であると仮定する。

ネストは、SuperSQL の反復子のネスト定義によって、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。すなわち、属性間の包含関係が開発者によって指定できる。SuperSQL のネストの適用によってひとまとめになる属性と、これらの属性によって入れ子構造にされたレイアウトが生成できる。すなわち、入れ子構造が定義できることで、実際の Web サイトの項目別の分類ページなどのカテゴリ化したレイアウトを生成することが可能になる。

また、グループینگは中括弧（{ }）で複数の属性を含む部分式をくくることで、くくられた属性の関連性が他の属性より高いことを指示する。中括弧によるグループینگは TFE の大括弧の指定による検索結果のグループینگ、すなわちネスト化とは異なる属性の集合を定義することである。続いて、この 3 つの開発者制約について詳しく述べる。

3.2.1 開発者制約 - 順序

開発者が元になる問合せ文に定義する属性の順序による制約を，次のように定義する．

定義 3.3 (開発者制約 - 順序)

元になる問合せ文に現れる属性の順序を，開発者制約の 1 つである順序制約とする．

図 3.2 のように選手と国に関する属性の順に定義された問合せ文から，システムが幅制約などを満たすために属性の順序を変更した場合 (*p.height* と *n.name* 属性の位置変更)，開発者が意図した意味とは異なるレイアウトが生成される恐れがある．

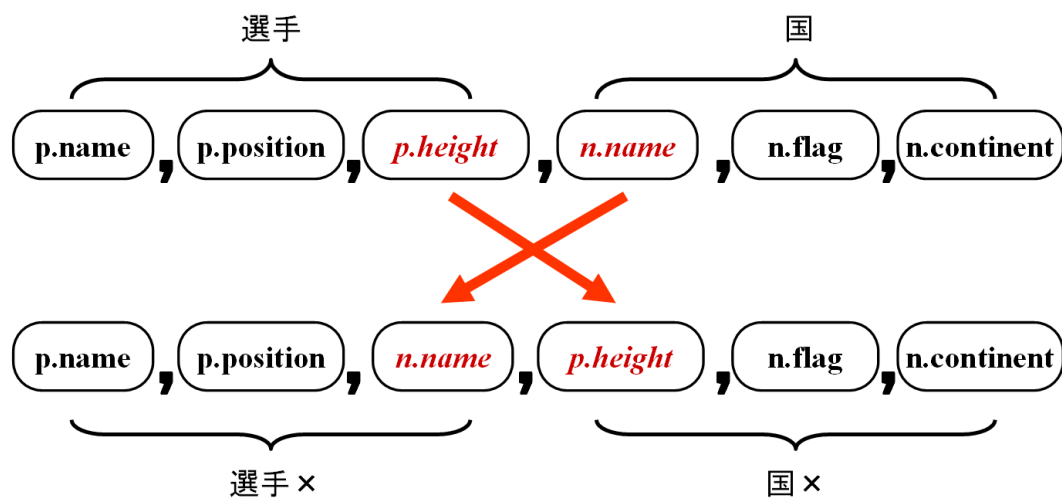


図 3.2: 開発者による属性の順序定義

属性の順序を変更すると，開発者が意図した属性間にあった関連性が崩れてしまい，開発者が意図した意味がなくなってしまうと判断する．すなわち，元の問合せ文に定義されている属性の順序は開発者が意図的に定義したものであり，ACTIVIEW で変換を行う際でも守るべきである．したがって，提案システムでは属性の順序を変換する処理は行わない．

3.2.2 開発者制約 - 入れ子

2.1.3 項で述べたように，SuperSQL では問合せ文に反復子の指定によるネスト構造を定義することによって，同じ属性値をもつタプルをひとまとめにすることができる．すなわち，ACTIVIEW は開発者が指定した属性間の包含関係に従って入れ子構造の Web ビューを生成する．

England	Michael OWEN	FW
	Peter CROUCH	FW
	David BECKHAM	MF

Japan	Shinji ONO	MF

...

[*team_country* ,
[*player, position*]!
]!

図 3.3: チームによるネスト

England	Michael OWEN	FW
England	Peter CROUCH	MF
England	David BECKHAM	
England
...	...	
Japan	Shinji ONO	MF
Japan	...	
...

[[*team_country* ,
player]! , *position*
]!

図 3.4: ポジションによるネスト

本論文では SuperSQL のネストの適用によってひとまとめになる属性と、これらの属性をまとめて入れ子構造にする属性は強い関連性をもっていると仮定する。

どの属性を用いてネストを行うかによって、同じデータに対するものであっても異なる意味をもつレイアウトが生成される。

図 3.3 は、チームごと (*team_country*) にネストされた各選手の名前とポジションを表示するレイアウトであり、図 3.4 は各ポジション別 (*position*) にネストされたチーム名と選手を表示している。したがって、レイアウトの変換を適用する際、開発者の意図、すなわち、あらかじめ開発者によって与えられたネスト構造に関する条件は満たすべきである。

したがって、入れ子制約の定義は次のようになる。

定義 3.4 (開発者制約 - 入れ子)

元の問合せ文に現れる反復子によるネスト指定に従って入れ子構造を維持する条件を、開発者制約の 1 つである入れ子制約とする。

3.2.3 開発者制約 - グループング

SuperSQL では演算子の適用順序を明示する目的で問合せ文にグループング ({ } 演算子) を指定することができる。ここで、中括弧でくくられた属性の間は他の属性との間より高い関連性をもっていると判断する。開発者制約の中のグループング制約の定義は次のようになる。

定義 3.5 (開発者制約 - グループング)

属性を中括弧でくくることによって、元の見合せ文に指定されているグループング指定による構造制約を、開発者制約の 1 つであるグループング制約と定義する。グループングされた属性の間は、他の属性との関係より高い関連性を持っていると判断し、レイアウトの変換を行う際に尊重する。

たとえば、図 3.5 の①のように中括弧でくくられた属性 *p.name* , *p.position* , *p.height* の 3 つの属性の関係は、他の *g.group* の②や *n.country* などの③の方に含まれる属性との関係より関連性が高いことを意味する。したがって、開発者による中括弧の指定は意図的な意味の反映であると判断する。ACTIVIEW システムでは、要素が中括弧でくくられると 1 つの要素としてまとめて処理する。

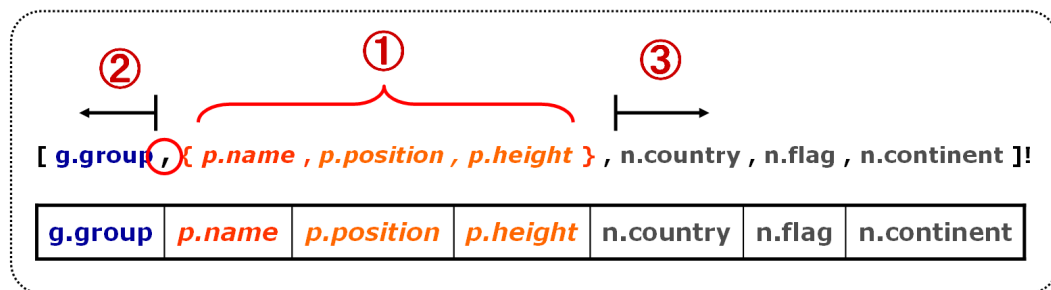


図 3.5: 中括弧との横連結による結果レイアウト

図 3.5 では中括弧で属性がくくられているが、横連結子と結合されており、レイアウトの構造では影響がないように見える。しかしながら、システム内では①の部分をもとめて処理を行っている。

中括弧のレイアウトへの影響は、中括弧との連結子が縦連結子である場合、はっきり分かる。図 3.5 の丸の中の横連結子 (,) が縦連結子 (!) になる場合、中括弧が使用されていない場合 (図 3.6) のレイアウト変換結果と、中括弧が使用されている場合 (図 3.7) のレイアウト変換結果を比較して見る。

[**g.group**! *p.name* , *p.position* , *p.height* , n.country , n.flag , n.continent]!

g.group	<i>p.position</i>	<i>p.height</i>	n.country	n.flag	n.continent
<i>p.name</i>					

図 3.6: 中括弧が使用されていない場合の問合せ文と結果レイアウト

[**g.group**! { *p.name* , *p.position* , *p.height* } , n.country , n.flag , n.continent]!

g.group			n.country	n.flag	n.continent
<i>p.name</i>	<i>p.position</i>	<i>p.height</i>			

図 3.7: 中括弧が使用されている場合の問合せ文と結果レイアウト

図 3.6 は問合せ文に中括弧が使用されていない場合で，*g.group* と横連結子 (,) でつながってある *p.name* のみに *g.group* 属性と縦連結するレイアウト変換を行う。しかし，図 3.7 の問合せ文に中括弧が使用されている場合は，3 つの属性が同じ集合として扱われて同時に移動するレイアウト変換を行う。

3.2.4 中括弧の導入

基本的に，開発者が問合せ文に指定している構造化指定以外をシステムが自動的に書き込むことは，開発者が意図しているレイアウトと異なるものを生成してしまう恐れがある。しかしながら，3.2.3 項で述べた中括弧によるグルーピング指定は，利用する手法によってレイアウトを変換する際に非常に役に立つことが予備実験によって判明した (4.4 節)。開発者による意味反映が崩れない範囲での中括弧の自動的な導入によって，意味的に関連性が高い属性同士に一括変換されたレイアウトの生成がより簡単にでき，変換コストを削減できる (5.4 節)。

たとえば，図 2.14 の木構造の下の部分を簡単に示すと図 3.8 のようになる。元の問合せ文 (図 2.12) は，図 3.8 のように *coach* の情報を検索した結果のみが中括弧でくくられる問合せ文である。

図 3.8 の木構造から，関係データベース中の同一テーブルに格納されている属性どうしは，他のテーブルに格納されている属性との関係より，意味的に関連性が高いと仮定

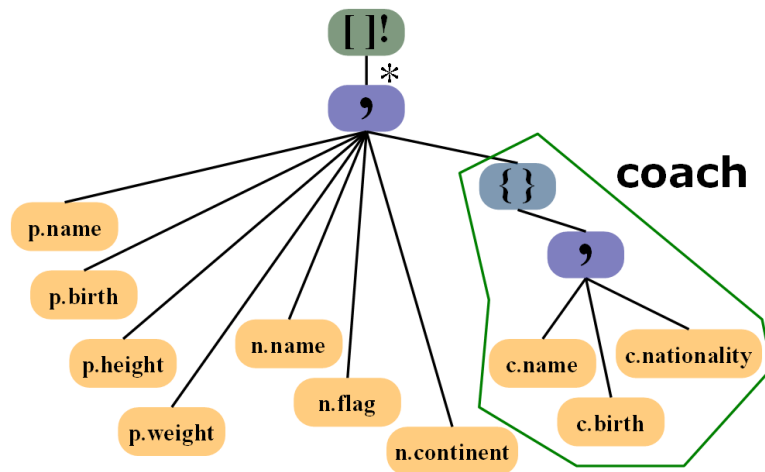


図 3.8: coach の情報のみが中括弧でくくられている場合

する．この仮定を基に，同一テーブルからの検索結果は1つの中括弧に纏めることが可能であると仮定する．関係データベースの同一テーブルに格納されていることは，属性の定義から判断できる．すなわち，隣接した属性，“ p.name ”，“ p.birth ”，“ p.height ”，“ p.weight ”は同一のテーブルからの検索結果であることが分かる．

したがって，中括弧を導入することによって図 3.9 のように選手とチームの情報に分けて集められた構造を定義することが可能になる．これは例外的な処理であり，適用する際は注意が必要である．

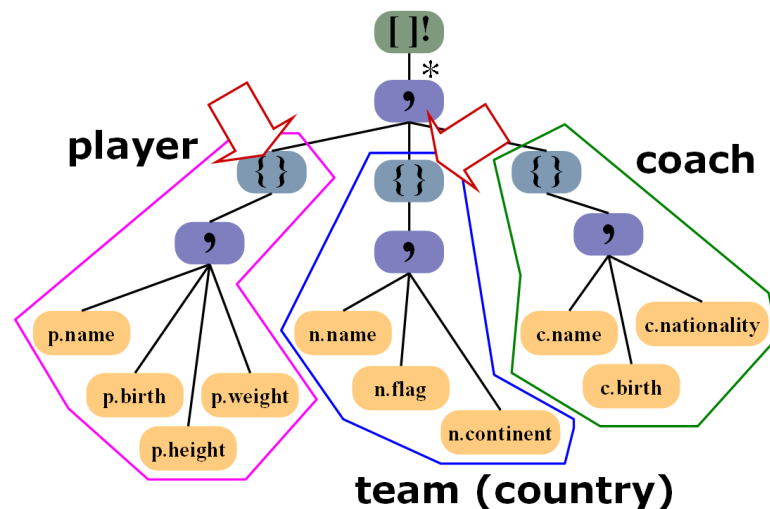


図 3.9: 中括弧を自動的に導入した場合

3.3 幅占有率目標

元の問合せ文に対し構造の変換を行うことで生成されたレイアウトの中で、制約を満たしたレイアウトが複数存在する場合、ユーザ表示画面に最適化されたレイアウトを選択するためには、複数存在する候補レイアウトから結果レイアウトを決定するための指標が必要になる。本論文では、幅制約と開発者制約の 2 つの制約とともに適応化の指標として“幅占有率目標”、“充填率目標”、“長さ目標”の 3 つの目標を提案している。

まず、画面表示に対して再構成されたレイアウトの幅がどれくらい効率的かを判断する幅占有率目標を次のように定義する。

定義 3.6 (幅占有率)

ユーザ表示画面に対して構成されたレイアウトの幅の割合を幅占有率とする。

すなわち、幅占有率はユーザ表示画面の幅 $W(U)$ に対する結果レイアウトの幅 $W(R)$ の比率 (図 3.10) である。

制約を満たしたレイアウトが複数存在する場合は、指標の 1 つである幅占有率を基準とした評価関数によるランキングを行う。問合せ文 q から生成されたレイアウトの幅占有率は次のような評価関数 $TW(q)$ によって求める。

$$TW(q) = \frac{W(R)}{W(U)} \times 100 \quad (3.3)$$

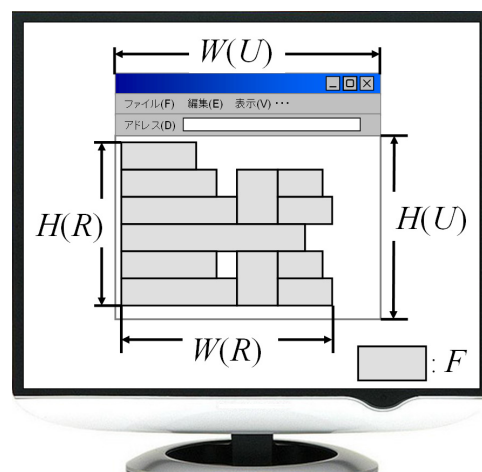


図 3.10: ユーザ表示画面に対する結果レイアウト

3.4 充填率目標

複数存在する候補レイアウトの中で充填率が高いレイアウトは表示空間を有効に利用していると考えられる。すなわち、画面表示に対して再構成されたレイアウトがどれくらい効率的に空間を利用しているかを判断する指標である。充填率は次のように定義する。

定義 3.7 (充填率)

ユーザ表示画面に対して構成されたレイアウトの占める面積の割合を充填率とする。

すなわち、ユーザ表の大きさである { 表示画面の横幅 ($W(U)$) × 表の長さ $H(R)$ } の内部における実質データの表示に要する面積 (F) の比率を充填率 (図 3.10) とし、候補レイアウトの中からより充填率が高いレイアウトを優先する。充填率の評価関数 $TF(q)$ は次のように定める。

$$TF(q) = \frac{F}{W(U) \times H(R)} \times 100 \quad (3.4)$$

図 3.11 は充填率による評価の 1 つの例である。網かけ部分はデータの表示されていない空白を表している。図 3.11 のように同じ幅をもつレイアウトであってもレイアウトの構造によって異なる充填率をもつ場合がある。この場合はより高い充填率をもつ、上のレイアウトが効率的に空間を利用していると判断され、優先される。図 3.11 が示すように連結子 1 つが変換されることによって充填率は大きく変わる。

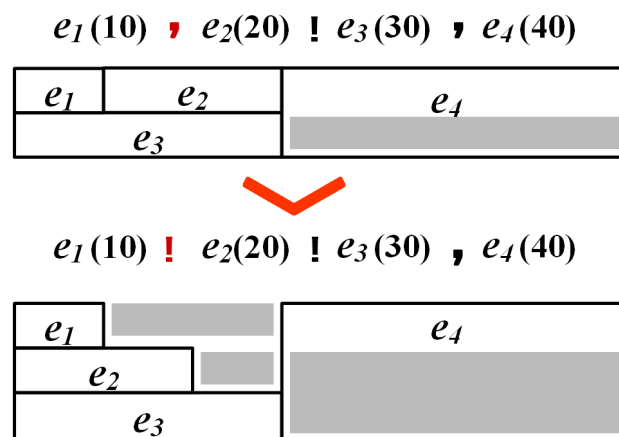


図 3.11: 充填率の比較

幅占有率は問合せ文に指定されている属性の幅値に従うことが多い。

連結子の変換によって影響を受けるのは属性の幅値であり、属性の幅値が大きく異ならなければ、計算結果の差はそれほど大きくないと考えられる。

しかし、1つの連結子の変換であっても充填率は大きく変わる。特に、ネスト化された構造は反復的に現れるため、間違っただ変換による低い充填率をもつ変換手法は結果レイアウトに大きい影響を与えてしまう。このように、変換手法の最適化の重要性は高い。

3.5 長さ目標

幅制約を満たしたレイアウトを生成する際、縦に非常に長いレイアウトのテーブルが生成されてしまう恐れがある。この問題を解決するため、長さ目標を定義し、この条件を満たしたレイアウトを候補レイアウトとする。

定義 3.8 (長さ目標)

構成されたレイアウトの長さ、ユーザ表示画面の長さとの割合が式 $TH(q)$ を満たしている場合、長さ目標を満たしたレイアウトとする。

問合せ文 q に対する長さ目標関数 $TH(q)$ は、次のように定義する。

$$TH(q) = \begin{cases} true & \text{if } \sum_{i=1}^n H(e_i) \leq H(U) \times d(q), \\ false & \text{if } \sum_{i=1}^n H(e_i) > H(U) \times d(q). \end{cases} \quad (3.5)$$

$H()$ はレイアウトの長さを計算する関数である。生成されるレイアウトの長さは、縦に連結されている各要素のセルの長さ（高さ、HTML の height で指定する値で以後“長さ”に統一する）を足し合わせることで計算できる（図 3.12 のボードのイタリック体で示した要素）。

構成されたレイアウトの長さ（ $\sum_{i=1}^n H(e_i)$ ）がユーザ表示画面の長さ（ $H(U)$ ）に、ある倍率 $d(q)$ を掛けた長さに比べて短いレイアウトでないと、長さ目標は満たされない。

$d(q)$ を計算するためには、検索結果がどれぐらいの長さを持っているかを計算し、ユーザ表示画面との割合を計算する必要がある。

関係データベースから検索結果をもらってから計算すると正確な値を計算できるが、それではレイアウトを変換するたびに繰り返して検索を行い、この検索結果を再計算

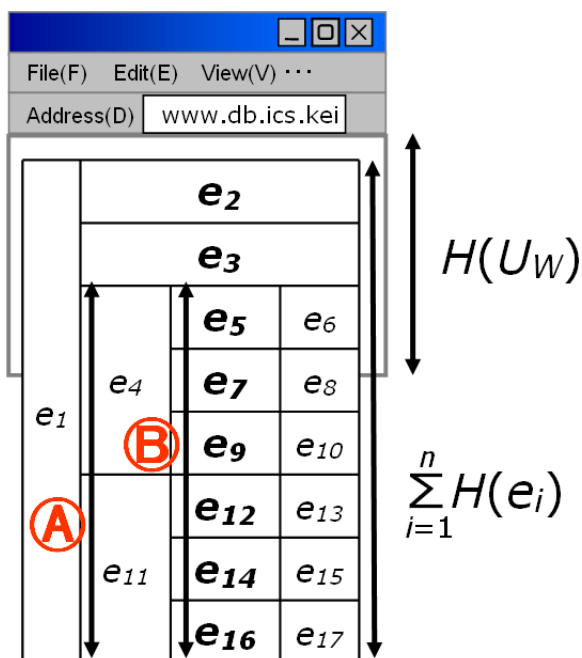


図 3.12: 長さ目標を満たす前のレイアウト

する必要が生じる．この手法は性能面で問題があるため，元の問合せ文に定義された初期のレイアウトが複雑なレイアウトであると，ユーザに動的な対応ができない場合が生じる．

そこで，元の問合せ文から結果レイアウトの長さを推測する手法を考える．

まず， $longest(H(n_x))$ を入れ子構造を構成している各ネスト n_x (図 3.12 のⒶとⒷ) の中で最も長いネストの長さとして定義する．グルーピングで含まれる結果が何回反復されるか計算した反復回数を IC (データ代表要素個数，5.2.5 項で詳しく述べる．) と定義する．予想された結果レイアウトの中で一番長いネストの長さに対するユーザ表示画面の長さと反復回数を掛けた値の割合の値 $p(x)$ を計算する．

$$p(x) = \frac{longest(H(n_x))}{H(U) \times IC_x} \quad (3.6)$$

図 3.12 のⒶとⒷの長さは構造化を行った後，実際に現れる属性の長さを合計して計算する．すなわち，ネストⒶの長さは e_4 と e_{11} が縦に連結されるレイアウトで，セルの長さが 30pixel であるとすれば，サブレイアウトの長さは $2 \times 30 = 60\text{pixel}$ になり，ネストⒷの長さは $6 \times 30 = 180\text{pixel}$ になってネストⒷが $longest(H(n_x))$ になる．各ネスト別に計算することは長さ目標を満たすため，リンク変換が適用されると，リンク

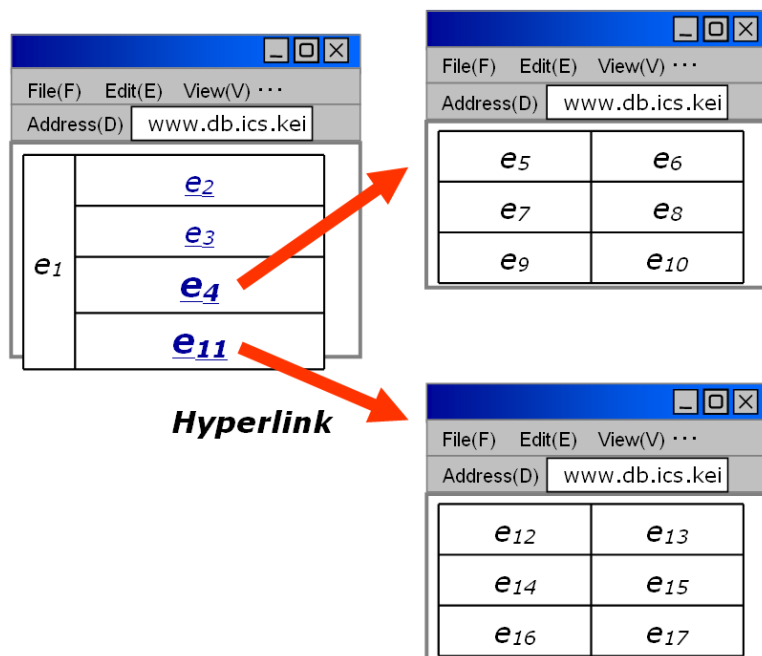


図 3.13: 長さ目標を満たしたレイアウト

が付けられた各ネスト別のレイアウトが生成されるからである。
問合せ文が

$$[A , [B] !] !$$

の場合、 B の構造は A の値の個数だけ反復される。そこで、 B の反復回数 IC_B は A の個数²になる。すなわち、ネストと属性の間の連結子にリンク変換が行われると、ネストの長さを属性の反復回数で割った長さのレイアウトが生成される。

図 3.12 で示すように、180pixel だったサブレイアウト⑥の長さが図 3.13 では 2 つのページに分けられ、90pixel になるのが分かる。

$p(x)$ の計算結果から、 $d(q)$ を次のように定義する。 y は開発者が指定した初期値である。

$$d(q) = \begin{cases} p(x) & \text{if } 1 < p(x) \leq y, \\ y & \text{if } p(x) > y, \\ 1 & \text{if } p(x) \leq 1. \end{cases} \quad (3.7)$$

²ここでの個数は属性に対する検索結果の個数ではなく、データ反復回数 (IC_x)。5.2.1 項によって計算された値である。

最も長いサブレイアウトを，ユーザ表示画面の長さで反復回数で割り算した結果が，開発者指定の変数 y より長い場合は，開発者指定の変数 y が基準になる．この場合は，リンク指定で複数になったレイアウトの長さが，開発者指定の変数 y を満たさない場合がありうるため，可能な範囲での長さ目標を許容する．

たとえば，開発者が指定した y が 3 であるとき，ACTIVIEW システムは開発者指定に従って結果レイアウトの長さがユーザ表示画面の長さの 3 倍以内であると，長さ目標を満たしたレイアウトであると判断する．

$longest(H(n_x))$ が 1,200pixel で，ユーザ表示画面の長さが 100pixel，反復回数が 3 である場合はリンク変換によって生成されるレイアウトの長さが 400pixel と予想され，開発者の基準 300pixel では (100×3) 長さ目標は満たされない．この場合の $d(q)$ は定義に従い，計算結果ではなく開発者指定の 3 になり，例外のサブレイアウト以外のサブレイアウトは開発者が指定した基準で再構成を行う．これは，サブレイアウトの長さを短くしても目標基準を満たすことができなくなるという問題が生じるためであり，例外として扱う．しかし， $longest(H(n_x))$ が 600pixel であれば開発者指定は超えるが，反復回数で計算すると 200pixel になり，開発者指定より短くすることが可能なレイアウトである．この場合は定義の通り，200pixel が長さ目標の基準になる．

元の問合せ文からの各ネストの長さを予想する手法については 5.2.1 項で詳細を述べる．

第 4 章

ACTIVIEW によるレイアウト変換

本章では、元の間合せ文によって生成される初期レイアウトから、制約や目標を満たすレイアウトを生成するレイアウト変換手法について述べる。

4.1 レイアウト変換の基本概念

SuperSQL の間合せ文によって生成される Web ビューは、2.1 節で説明したとおりに間合せ文に指定する TFE の定義によって構造化される。ACTIVIEW による構造の変換は、元の間合せ文に指定されている TFE の変換による 2 つの手法が考えられる [11]。

- (1) 横連結 (,) を縦連結 (!) に変換する。
- (2) 横連結 (,) か縦連結 (!) をハイパーリンク連結 (%) に変換する。

(1) の手法は、横に並んで表示されるべきデータが縦に並べられるので、表としてデータ間の関係が理解しやすく、ハイパーリンクで辿るよりも手間が少ない。しかしながら、大量のデータによって非常に長いレイアウトが生成される場合や、携帯端末など表示画面のサイズが極端に小さいものでは (2) の手法も採用する。携帯端末に代表される小型端末の特徴を次にあげる。

- 小さい表示画面
- 上下左右ボタンによるスクロール操作

これらの特徴を考慮し (2) の手法を採用する理由は、まず、ユーザが必要とする情報を素早く提供するため、限られた画面に多くの種類のデータを表示する必要がある。

次に、スクロールに大きな手間がかかる場合、ハイパーリンクで情報に飛ぶときの手間の方が少なくなることがあげられる。

この基本概念に従ってレイアウト変換手法を“横縦変換”と“リンク変換”の 2 つの具体的手法として定義する。これらのレイアウト変換手法を用い、ユーザ表示画面に適応したレイアウトを提供する手法を提案する。

4.2 横縦変換による幅変化

横連結から縦連結にレイアウトを変換することで“幅制約”を満たしたレイアウトが生成できる。さらに、この変換の適用によって“幅占有率目標”や“充填率目標”に従ったレイアウトの生成が可能になる。横縦変換を次のように定義する。

- ① GENERATE ACTIVIEW [**group , country , player**]! FROM... WHERE...
- ② GENERATE ACTIVIEW [**group ! country , player**]! FROM... WHERE...
- ③ GENERATE ACTIVIEW [**group ! country ! player**]! FROM... WHERE...

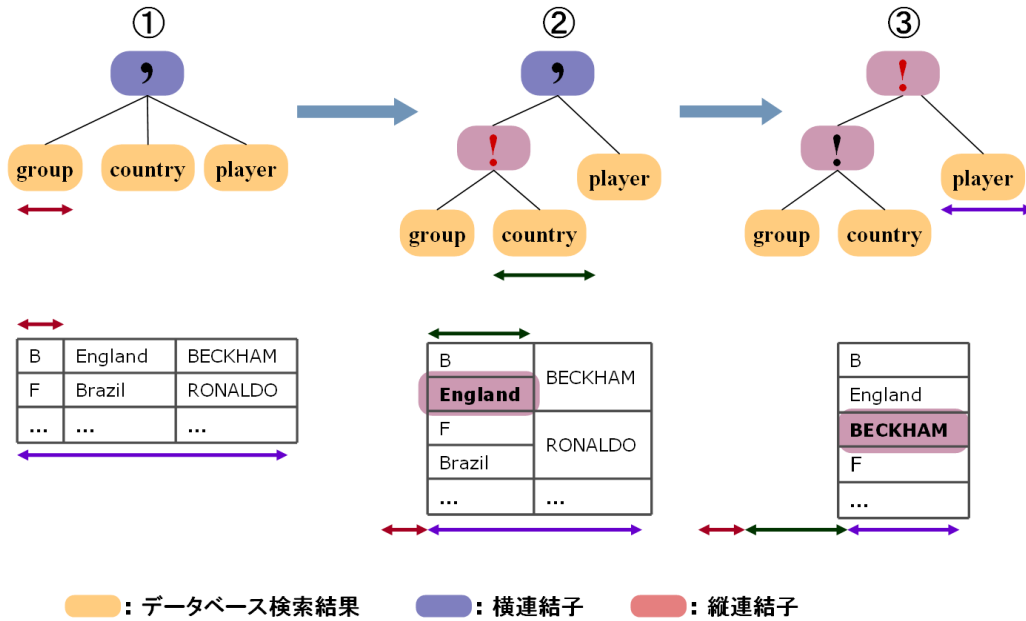


図 4.1: 横縦変換

定義 4.1 (横縦変換)

ACTIVIEW 問合せ文に指定されている横連結 (,) を 1 つ , または複数選び , 縦連結 (!) に変換する手法を横縦変換とする .

すなわち , 図 4.1 のように元の問合せ文にある横連結 (,) が 1 つずつ縦連結 (!) に変換されることで , 縦に変換される連結子の右辺の分だけ幅が狭いレイアウトが生成される .

図 4.1 で見ると , まず group と country を結合していた横連結子 (,) が縦連結子 (!) に変換されることで , 全体幅は group の幅の分だけ全体幅は狭くなる . また , country と player を結合していた横連結子 (,) が縦連結子 (!) に変換され , country の幅の値分 , 結果レイアウトは縮められる . 図 4.1 では順番通りに変換を行ったが , 変換の手法は色々あり , 実際のユーザには最適化手法によって選択された結果レイアウトが提供される .

4.3 リンク変換による長さ変化

“ 長さ目標 ”を満たしたレイアウトを生成するため、リンク変換を定義する。

定義 4.2 (リンク変換)

ACTIVIEW 問合せ文に指定されている結合子を1つずつ選び、リンク連結(%)に変換する手法をリンク変換とする。

図 4.2 のようにリンク変換を適用することによって、リンク元の Web ページとリンク先の Web ページの複数のレイアウトが生成される。生成された複数のレイアウトはそれぞれ制約と目標を満たさなければならない。図 4.2 の結果から見ると、group にネストされた country 属性の検索結果がリンク先になり、全体的な長さが縮められた。

① GENERATE ACTIVIEW [**group** , [**country**]!]! FROM... WHERE...

② GENERATE ACTIVIEW [**group** % [**country**]!]! FROM... WHERE...

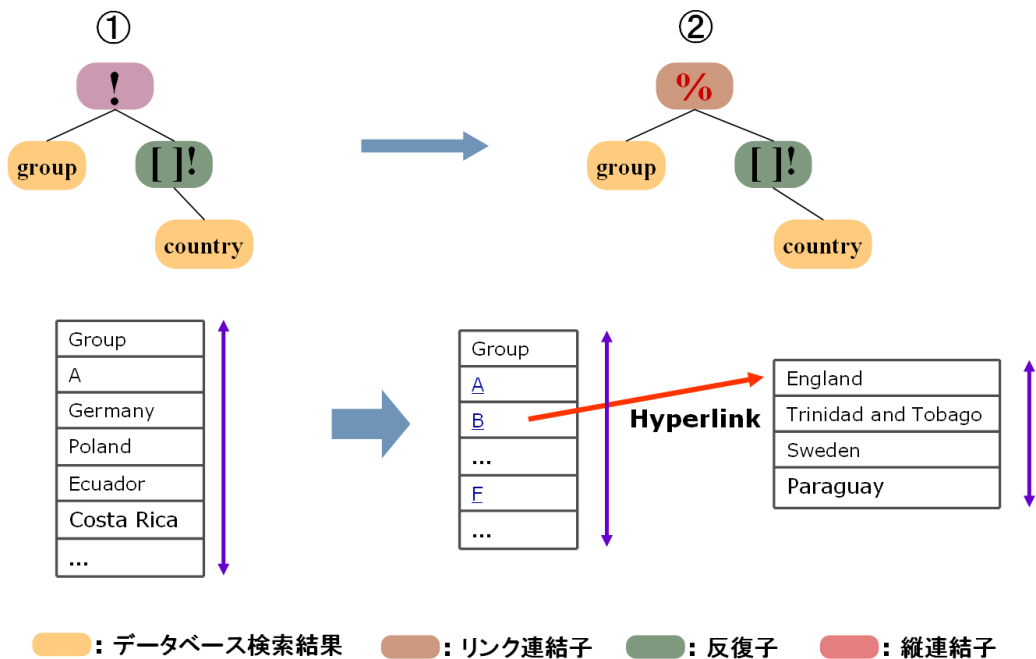


図 4.2: リンク変換

4.4 予備実験

ACTIVIEW では、レイアウトの変換手法に従いつつ、初期レイアウトを制約と目標に合わせて変換することで、ユーザ表示画面への適応を実現する。ここで問題となるのが、どの連結子を縦連結またはリンク連結に変換するのかという点である [12, 10]。これらの問題を解決するために様々な研究を行った。これらの研究の結果から得た知見を述べ、この知見に基づいて新しい変換手法を提案する。

4.4.1 従来のレイアウト変換

従来のレイアウト変換手法 [16] として、結合子が現れる順序どおりにレイアウト変換を適用する“単純順序探索”と、適用可能なすべての結合子に対して変換を行うことによって生成される、全レイアウトを対象にする“全数探索”の 2 つの手法がある。

“単純順序探索”によるレイアウト変換手法は、問合せ文に属性が現れる順番通りにレイアウト変換を適用する手法である。

m 個の属性に対し、レイアウト変換の適用が可能なのは $m - 1$ の連結子であり、生成可能なレイアウトの個数は $m - 1$ 個になる。元の問題文に現れる順序通りに (図 4.3 の矢印の順番) 横連結子から縦連結子に変換を行う。

これは、必ず満たすべきである制約条件を満たしたレイアウトを生成可能だが、目標に従った最適化は行われない。そのため、“単純順序探索”によるレイアウト変換手法では、ユーザに望ましくない (見にくい) レイアウトが生成される可能性がある [16]。

“全数探索”の手法は元の問題文から生成可能な全レイアウトパターンを対象にしている。変換する連結子は水平結合子 (,) , 垂直結合子 (!) , 深度方向への結合子 (%) の 3 つが対象になるので、 n 個の演算子をもつ問合せ文から構成されるレイアウトの個数 $P_B(n)$ は、

$$P_B(n) = 3^n \tag{4.1}$$

になる。さらに、自動的に中括弧を適用して候補のレイアウトを生成する場合に生成されるレイアウトパターンの個数 $P_M(n)$ は非常に増えてしまう。

$$P_M(n) = \sum_{k=1}^{n-1} P_M(k) \cdot P_M(n-k) \tag{4.2}$$

生成可能な全レイアウトパターンを対象にしているため、最適な目標の結果値をもつレイアウトを生成することができる。しかしながら、図 4.4 の実験結果から分かるよ

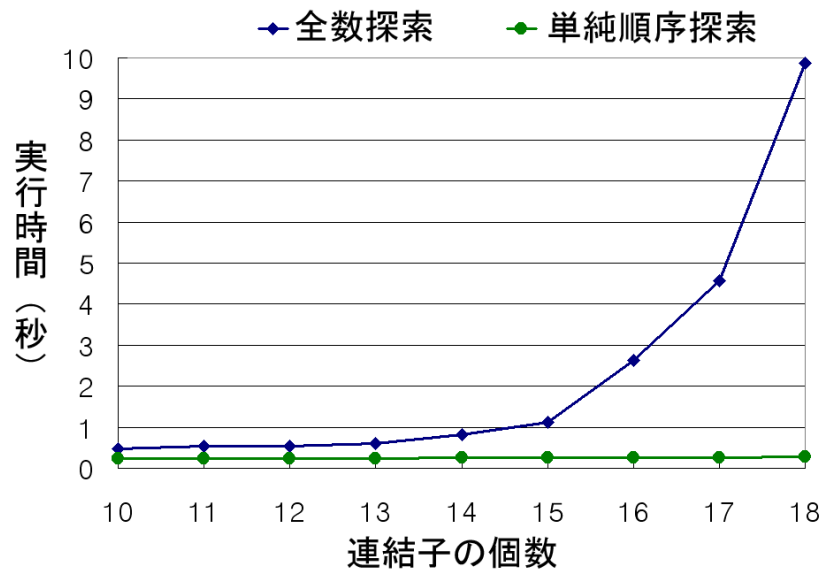


図 4.4: 単純順序探索と全数探索の実行時間

うに、属性数が多いときに処理時間が非常に大きくなる問題があり、Web 上でのユーザに対する動的な対応が困難になる。図 4.4 を見ると、Web 上でのユーザに対する動的な対応を可能にするためには、全数探索の対象になる連結子の数を 15 個以下に減らさなければならない。

さらに、属性間の関連性を考えていないレイアウトは、高い結果値で目標を満たしたレイアウトを生成していても、ユーザに見やすいレイアウトであるとはいえない。

そこで、計算コストを下げながら、制約と目標に従った変換を求める適用手法が必要になる。新しい変換手法を提案するために、今まで述べたレイアウト変換手法に対して様々な実験を行い、その結果の比較を行う。この比較から取り出した発見的な手法を基に、新しいレイアウト変換戦略を提案する。

4.4.2 予備実験による比較

2つの基本変換手法を適用したレイアウトを生成する場合に関して、ユーザ表示画面の幅に対するレイアウト結果の幅占有率と充填率を比較する。

予備実験では実際の Web サイトを、ACTIVIEW の問合せ文の定義によって生成可能なレイアウトの対象になる、カテゴリ化可能なページ、テーブル型でコンテンツを表示するデータ集約的な Web ページ、複雑なテーブル型ページの 3つの種類に分類したうえで、2つの基本変換手法を適用したレイアウト変換結果の分析を行った。

```

GENERATE ACTIVIEW
[ { "YEAR" @{\width = 60}, h.year@{\width=70} },
  [ { "GROU" @{\width=80}, t.group@{\width=20} },
    [ { "COUNTRY" @{\width=90}, t.country@{\width=240} },
      { "COACH" @{\width=80}, c.name@{\width=200} },
      { "NATIONALITY" @{\width=130}, c.nationality@{\width=190} }
    ]!
  ]!
]!
]!
FROM host h, teams t, coach c
WHERE h.id = t.id and h.id = c.entry ...

```

■ : 文字列定数 ■ : オペレータ ■ : 装飾子

図 4.5: 予備実験に使われた問合せ文

予備実験の結果では、単純なレイアウト（入れ子構造がないレイアウトなど）をもつテーブルに対して、4.2、4.3 節で述べる 2 つの基本変換手法を適用したレイアウトの変換結果は元のレイアウトと大して異ならなかった。

そこで、本論文で提案するレイアウト変換戦略の有効性を示すために予備実験で行った様々なレイアウト構造を基に、より大きく構造変換が行われレイアウト構造を取り出した。予備実験によるレイアウト構造の分析を基に定義した問合せ文が図 2.12 (p.22) である。予備実験の結果、入れ子構造や中括弧によるグルーピングが定義されている問合せ文の方が、レイアウト変換手法の適用結果の差を認識しやすい。そこで、図 2.12 の問合せ文を比較実験の一例として取り上げた。

本項では、図 2.12 の問合せ文を簡単にした問合せ文（図 4.5）による実験結果を一例として、予備実験で行ったレイアウト変換手法による実際のレイアウトの変換結果を示す。実際のレイアウトの変換結果を示しながら、変換される連結子の違いによるレイアウトへの影響について述べる。図 4.5 の問合せ文から生成される初期レイアウトの幅は 1,150pixel であり、その幅を 240pixel まで縮めるレイアウト変換を行う。他の様々なレイアウトに対するレイアウトの変換結果の比較は、第 6 章で提案手法によるレイアウト変換との比較を行う際に示す。

まず、元の問合せ文の要素が現れる順通りに横縦変換手法を適用した場合 (*SEQ*) に対して実験を行った。文字列定数に関しては、次の 3 つの適用手法に対して予備実験を行った。

1. 各基本変換手法に従ってレイアウトを変換し、文字列定数が現われた場合も連結子として取り扱って連結子変換を適用した場合 (*SEQ(Same)*)

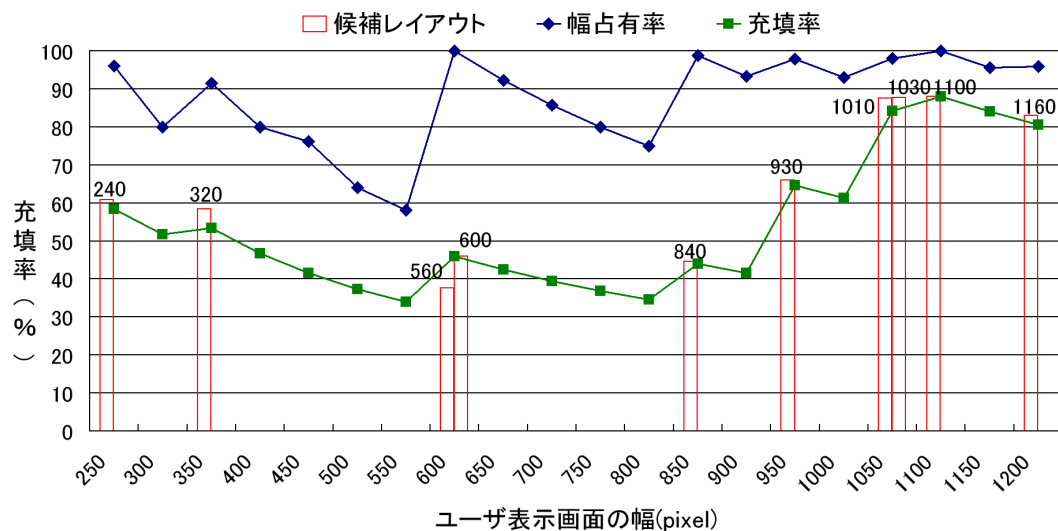


図 4.6: 結果レイアウトの幅占有率と充填率の割合 ($SEQ(Same)$)

2. 結合子の順通りに連結子変換ルールを適用し，かつ文字列定数が現われた場合には，同じネストに含まれるすべての文字列定数に対して，変換を適用した場合 ($SEQ(Group)$)
3. すべての文字定数変換に対し，変換を適用した後，連結子変換を結合子が現れる順に適用した場合 ($SEQ(AllStrFirst)$)

4.4.3 $SEQ(Same)$ 適用手法

図 4.5 の問合せ文に現れる順序通りに変換を行うことで生成される候補レイアウトの幅と充填率，これらの候補レイアウトが最終結果レイアウトとして提供される際のユーザー表示画面のサイズに対する幅占有率と充填率を図 4.6 に示す．図 4.6 の赤い縦棒が単純順序探索によって生成された結果（候補）レイアウトであり，縦棒の高さは生成されたレイアウトの充填率，縦棒の数字は生成されたレイアウトの幅を示す．

図 4.6 の実験結果から，単純順序探索はユーザー表示画面の幅に対するレイアウト結果の割合，すなわち幅占有率が 90% を下回る場合が多く，さらに充填率は，ほぼ 60% を下回るレイアウト結果を生成していることが分かる．図 4.6 から結合子が現れる順にレイアウト変換を適用する手法はあまり効果的ではないと判断できる．

特に，単純順序探索では，生成される 10 種類の非常に少ない数の候補レイアウトから結果レイアウトを選んでユーザに提供するしかないので，ユーザー表示画面の幅に対

(a) ユーザ表示画面の幅が 590pixel である場合の候補レイアウト

YEAR	590	
2000		
GROUP	560	
A		
COUNTRY		
Costa Rica		
COACH		
GUIVARAES Alexandre		
COUNTRY		
Ecuador		
COACH		
COUNTRY		

(b) ユーザ表示画面の幅が 550pixel である場合の候補レイアウト

YEAR	550	
2000		
GROUP		
A		
COUNTRY	320	
Costa Rica		
COACH		
GUIVARAES Alexandre		
NATIONALITY	Costa Rican	
COUNTRY		
Ecuador		
COACH		

(a) ユーザ表示画面の幅が 590pixel である場合の候補レイアウト

(b) ユーザ表示画面の幅が 550pixel である場合の候補レイアウト

図 4.7: $SEQ(Same)$ 手法によって生成される結果レイアウトの例

する結果レイアウトの幅占有率が極端に減少する（ユーザ表示画面の幅が 650pixel から 550pixel に減少する時）。すなわち、ユーザ表示画面の変換に従って生成されるレイアウトの変換結果が安定した結果を生成できないと判断される。

最終結果レイアウトの幅がこのように極端な値を示す理由は、以下の例で示される。たとえばユーザ表示画面の幅が 600pixel 以下では、図 4.7 の (a) の通りにユーザ表示画面の幅が 590pixel の良いレイアウト結果の生成ができた。しかし、ユーザ表示画面の幅が 550pixel になる（図 4.7 の (b)）と次に連結子の変換が行われるのが（中括弧でくくられている。）“ NATIONALITY ”と“ Costa Rican ”のペアの部分であり（図 4.7 の矢印）、レイアウトの変換を行うと次に縦連結される属性の幅の 320pixel 分、急に候補レイアウトの幅が縮まる。

図 4.6 から分かるようにユーザ表示画面の幅が 320pixel 以上から 560pixel 以下になる場合に提供できる候補レイアウトは幅 320pixel の候補レイアウト 1 つしかない。さらに、ユーザ表示画面の幅が 600pixel 以上から 840pixel 未満である場合も幅 600pixel の候補レイアウト 1 つしかないので、最終結果レイアウトとしてこの候補レイアウトを提供するしかない。

単純順序探索は、場合によって高い割合の結果を示すが（ユーザ表示画面の幅が 1,100pixel である時）、これは偶然であり、このような単純なレイアウト変換の適用ではユーザ表示画面の幅に適應したレイアウトの生成手法としては望ましくないと判断する。

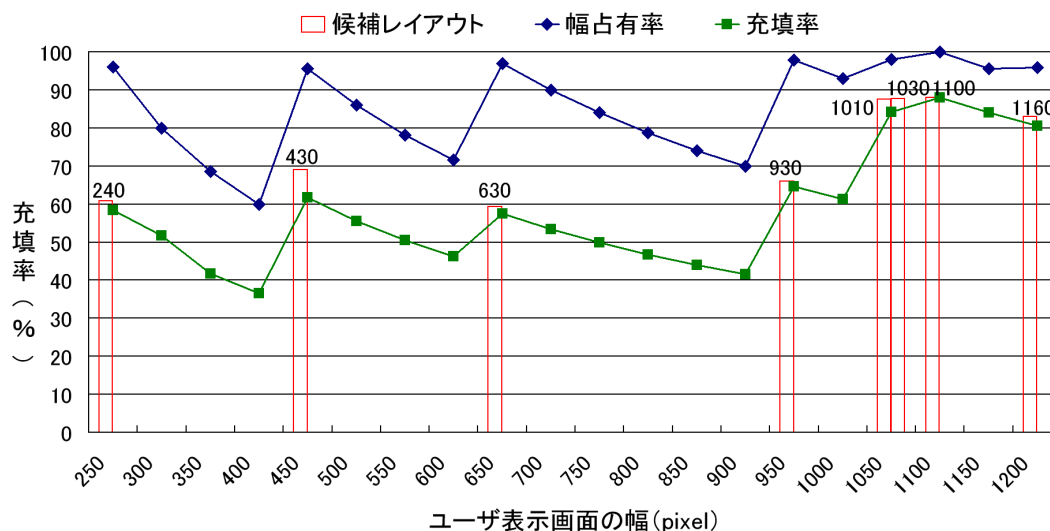


図 4.8: 候補レイアウトの幅占有率と充填率の割合 ($SEQ(Group)$)

4.4.4 $SEQ(Group)$ 適用手法

同一ネスト内の文字列定数に対して同時変換を行うことで生成される候補レイアウトの幅と充填率，この候補レイアウトが最終結果レイアウトとして提供される際のユーザ表示画面のサイズに対する幅占有率と充填率を図 4.8 に示す。

図 4.5 の問合せ文の“ GROUP ”と“ t.group ”の間の連結子に変換を行う際，“ COACH ”と“ c.name ”，“ NATIONALITY ”と“ c.nationality ”の間の連結子も同時変換を行う。この 3 つの連結子で変換を行って生成される候補レイアウトが存在するため，ユーザ表示画面の幅が 320pixel から 840pixel の間で図 4.6 とは異なる結果を示している。

図 4.9 の (a) の $SEQ(Same)$ 手法によって生成される 600pixel の幅をもつ候補レイアウトと比べて，図 4.9 の (b) の 630pixel の幅をもつ候補レイアウトのほうが統一的な構造をもつ一貫性のあるレイアウトを生成していることが分かる。しかし，生成可能なレイアウトの種類が少ないという問題によって，ユーザには幅占有率や充填率が低い結果レイアウトを提供することになってしまう。

4.4.5 $SEQ(AllStringFirst)$ 適用手法

問合せ文に現れるすべての文字列定数を先に変換を行った後，他の連結子に対して順序通りに変換を行うことで生成される候補レイアウトの幅と充填率，これらの候補レイアウトが最終結果レイアウトとして提供される際のユーザ表示画面のサイズに対

YEAR		
2006		
GROUP		
A		
COUNTRY	NATIONALITY	COACH
Costa Rica	Costa Rican	GUIMARAES Alexandre
COUNTRY	NATIONALITY	COACH
Ecuador	Colombian	KLINSMANN Juergen
COUNTRY	NATIONALITY	COACH
Ecuador (ECU)	SUAREZ Luis	SUAREZ Luis
COUNTRY	NATIONALITY	COACH
Germany (GER)	German	KLINSMANN Juergen
COUNTRY	NATIONALITY	COACH

(a) SEQ(Same) 手法によって生成される 600pixel の幅をもつ候補レイアウト

(b) SEQ(Group) 手法によって生成される 630pixel の幅をもつ候補レイアウト

図 4.9: SEQ(Same) と SEQ(Group) で生成される候補レイアウトの比較

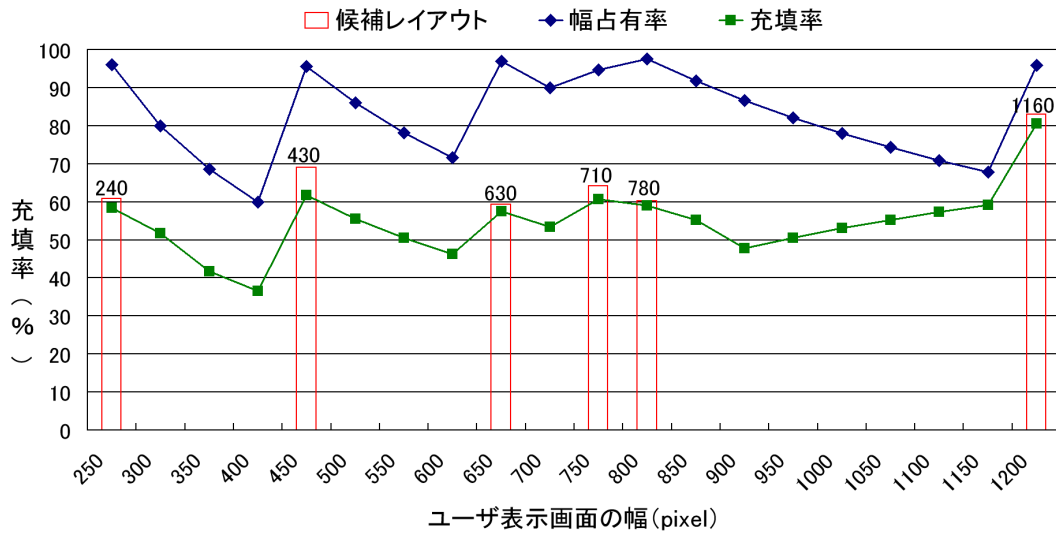


図 4.10: 候補レイアウトの幅占有率と充填率の割合 (SEQ(AllStringFirst))

YEAR				
2006				
GROUP				
A				
COUNTRY	COACH	NATIONALITY		
Costa Rica	GUIMARAES Alexandre	Costa Rican		
COUNTRY	COACH	NATIONALITY		
Ecuador (ECU)	SUAREZ Luis	SUAREZ Luis		
COUNTRY	COACH	NATIONALITY		
	KLINSMANN Juergen	German		

図 4.11: $SEQ(Same)$ 手法によって生成される 840pixel の幅をもつ候補レイアウト

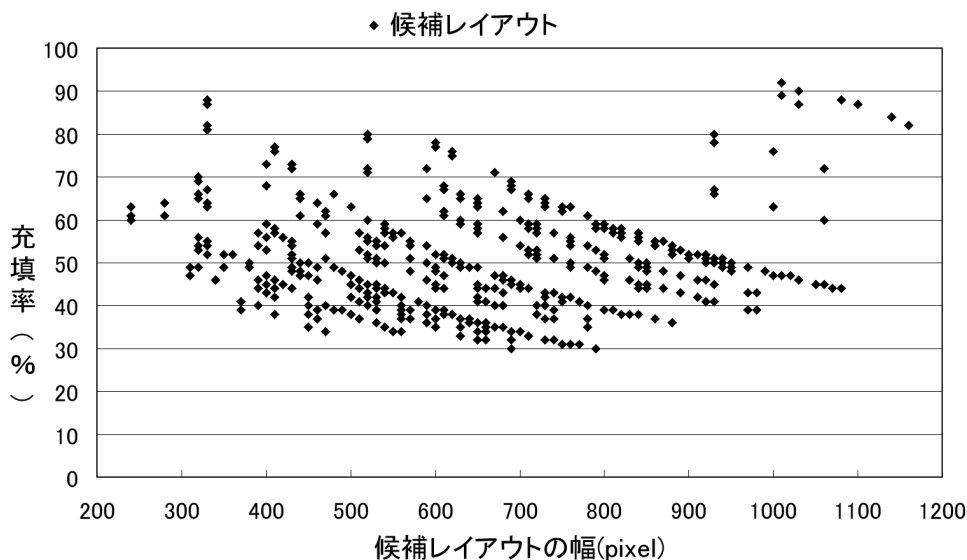
YEAR				
2006				
GROUP				
A				
COUNTRY	COACH	NATIONALITY		
Costa Rica	GUIMARAES Alexandre	Costa Rican		
COUNTRY	COACH	NATIONALITY		
Ecuador (ECU)	SUAREZ Luis	SUAREZ Luis		
COUNTRY	COACH	NATIONALITY		
Germany (GER)	KLINSMANN Juergen	German		
COUNTRY	COACH	NATIONALITY		
Poland (POL)	JANAS Pawel	Polish		
COUNTRY	COACH	NATIONALITY		

図 4.12: $SEQ(AllStringFirst)$ 手法によって生成される 780pixel の幅をもつ候補レイアウト

する幅占有率と充填率を図 4.10 に表す。

$SEQ(AllStringFirst)$ の手法は先に文字列定数に対して変換を行うために、最初から大きく縮まった候補レイアウト (780pixel の幅をもつレイアウト) を生成している。すなわち、図 4.10 から分かるように 800pixel から 1,160pixel 未満の間には候補レイアウトが生成されないため 780pixel の幅をもつ候補レイアウトを提供するしか方法がない。

しかし、ユーザ表示画面の幅が 850pixel である場合、提供される図 4.6 の 840pixel の幅をもつ候補レイアウト (図 4.11) と比べて、図 4.10 の $SEQ(AllStringFirst)$ 手法による 780pixel の幅をもつ候補レイアウト (図 4.12) のほうが幅占有率は低い、高い充填率をもっている。また構造的にも統一的にそろった一貫性のあるレイアウトを生成している。ユーザにとっては一貫性のあるレイアウトのほうが見やすいと感じると考えられる。

図 4.13: 候補レイアウトの幅の種類と充填率 ($P_B(Same)$)

4.4.6 $P_B(Same)$ 適用手法

全数探索を行うことによって図 4.5 の問合せ文から生成可能な候補レイアウトの種類を図 4.13 に示す．図 4.5 の問合せ文に現れる連結子の個数が 9 個であるので，512 種類のレイアウトを生成できる．図 4.13 から同じ幅の候補レイアウトであっても，変換された連結子の違いによって異なる充填率をもつレイアウトであることが分かる．

図 4.13 から分かるようにほぼすべてのユーザ表示画面の幅に，複数の候補レイアウトを提供することができるので，より良い結果を選択することが可能である．すなわち，よりユーザ表示画面の幅に適応したレイアウト結果を提供可能になる．

予備実験での P_B では生成できるレイアウトパターンから条件を満たすレイアウトが複数存在する場合，複数のレイアウトから一番幅広いレイアウトを結果として抽出した．同じ幅をもつレイアウト結果が現れた場合は，充填率が一番高いレイアウトを最後の結果にした．しかし，ユーザ表示画面の幅に収めることだけを基準にしているため，一貫性のあるレイアウト結果であるとは限らない．

たとえば， P_B によるレイアウト変換結果の一例である図 4.14 は幅 620pixel で 76% の高い充填率をもつ候補レイアウトであるが，ユーザに見やすいレイアウトであるとはいえない．したがって，幅占有率よりは意味的な関連性を生かした結果レイアウトのほうがユーザには見やすいレイアウトであると仮定できる．

YEAR	2006				
GROUP	A	COUNTRY	Costa Rica	NATIONALITY	Costa Rican
		COUNTRY	Ecuador (ECU)	NATIONALITY	SUAREZ Luis
		COACH	SUAREZ Luis	SUAREZ Luis	
		COACH	GUIMARAES Alexandre	GUIMARAES Alexandre	

図 4.14: $P_B(\text{Same})$ によって生成されるレイアウト結果の例

4.4.7 $P_B(\text{Group})$ 適用手法

同一ネスト内の文字列定数に対して同時変換を行って生成した候補レイアウトの種類を図 4.15 に表す。図 4.5 の問合せ文では、同一ネスト内に複数の文字列定数をもっている一番奥のネスト ($\{\{ \text{ " COUNTRY " } \dots \{ \text{width=190} \} \} \}$!) の 3 つの中括弧内の連結子は同時に変換されるので 1 つの連結子として扱える。したがって、変換対象になる連結子の個数は 7 になるので、生成される候補レイアウトの種類は 2^7 の 128 個になる。

図 4.15 は図 4.13 と比べて少ない数の候補レイアウトを生成しているが、同一変換による一貫性のあるレイアウトを提供する可能性が高くなると考えられる。さらに、充填率の観点から見ても 60% を超えるレイアウトが数多く出現しているので、一貫性を持ちながら高い充填率と幅占有率¹をもつレイアウトを提供することが可能であると考えられる。

$P_B(\text{Group})$ によって生成される候補レイアウトは図 4.16 と図 4.17 のように同一ネスト内で同時に変換を行い、統一されたレイアウトを生成する。3 つの連結子に同時に変換を行わない場合について図 4.18 に示す。

図 4.17 は 35% の低い充填率を示しているが、これは上位のネスト、すなわち“ YEAR ”と“ 2006 ”のペアおよび“ GROUP ”と“ A ”のペアのネスト部分が非効率的に空間を利用しているためである。このように反復的に現れるネストをもつ上位のネストは横連結になると非効率的な空間利用になってしまう。そこでレイアウト変換では、図 4.16 の“ GROUP ”と“ A ”のペアとネストが縦連結されているように上位ネストを縦連結に変換することを優先したほうが良いと判断した。

¹240pixel から 1160pixel まで全体的な幅に対して候補レイアウトを生成している。

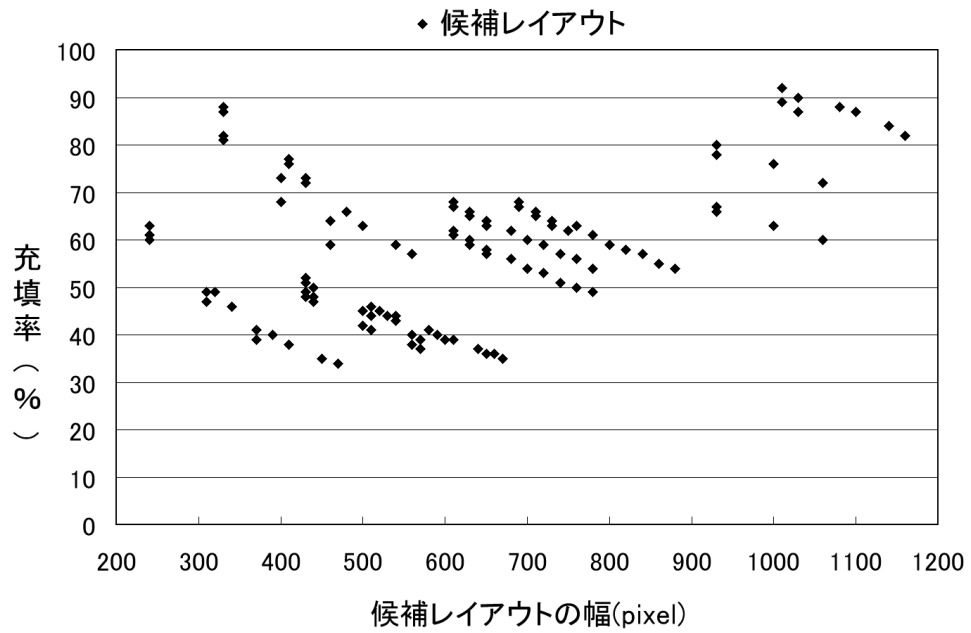


図 4.15: 候補レイアウトの幅の種類と充填率 ($P_B(Group)$)

		GROUP		
		A		
		COUNTRY	COACH	NATIONALITY
		Costa Rica	GUIMARAES Alexandre	Costa Rican
		COUNTRY	COACH	NATIONALITY
YEAR	2006	Ecuador (ECU)	SUAREZ Luis	SUAREZ Luis
		COUNTRY	COACH	NATIONALITY

図 4.16: $P_B(Group)$ によって生成されるレイアウト結果の例 (760pixel の幅)

YEAR	2006	GROUP	A	COUNTRY	COACH
				Costa Rica	GUIMARAES Alexandre
				NATIONALITY	
				Costa Rican	
				COUNTRY	COACH
				Ecuador (ECU)	SUAREZ Luis
				NATIONALITY	

図 4.17: $P_B(Group)$ によって生成されるレイアウト結果の例 (670pixel の幅)

YEAR				
2006				
GROUP	COUNTRY	Costa Rica	COACH	GUIMARAES Alexandre
	NATIONALITY	Costa Rican		
	COUNTRY	Ecuador (ECU)	COACH	SUAREZ Luis
	NATIONALITY	SUAREZ Luis		
	COUNTRY			COACH

図 4.18: $P_B(Group)$ によって生成されるレイアウト結果の例 (690pixel の幅)

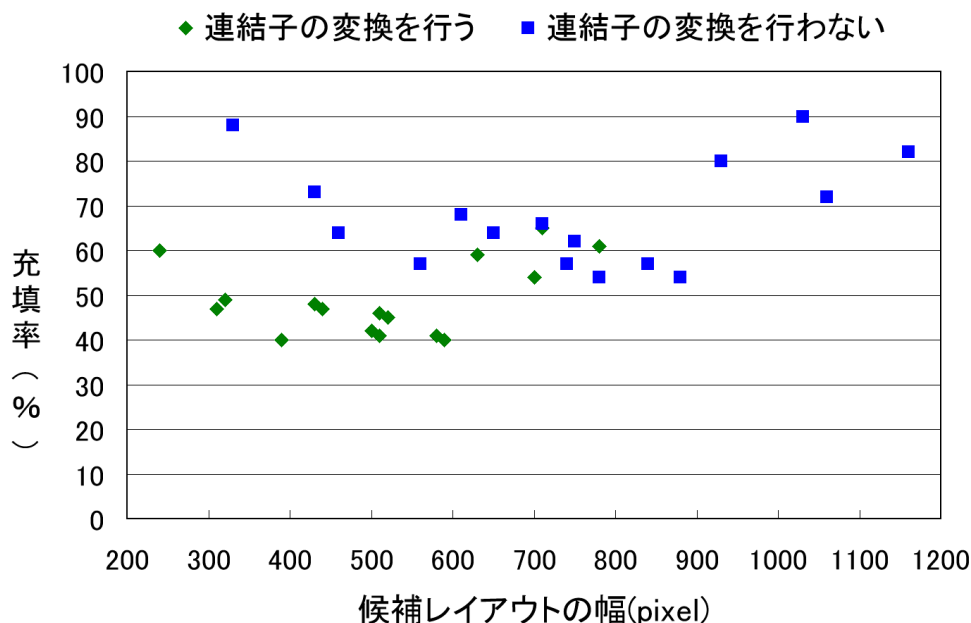


図 4.19: 候補レイアウトの幅の種類と充填率 ($P_B(AllString)$)

4.4.8 $P_B(AllString)$ 適用手法

図 4.19 はすべての文字列定数に対し、同時にレイアウト変換を行った場合と行わなかった場合の 2 種類のパターンに対して他の連結子に対して全数探索を行った ($P_B(AllString)$) 結果、生成される候補レイアウトの種類を表している。 $SEG(AllStringFrist)$ では、すべての文字列定数に対して連結子変換を行ったが、予備実験の $P_B(AllString)$ ではもっと多様な候補レイアウトを生成して結果レイアウト同士を比較してみるために、同時変換をした場合としなかった場合の 2 通りのパターンを用意してみた。

中括弧内の文字列定数との連結子を抜くと、残る連結子の個数は 4 つになり、 2×2^4 で 32 種類の候補レイアウトが生成可能になる。

図 4.19 は図 4.15 の $P_B(Group)$ によって生成される候補レイアウトの個数と比べて、よりと少ない数の候補レイアウトが生成されるのは当然である。しかし、関連性が高いと仮定した属性 (中括弧でくくられる \cdot) は構造的には統一した変換をおこなったレイアウトであるので、 $P_B(Group)$ での例のようにユーザには見やすいレイアウトであると考えられる。

たとえば、図 4.19 での文字列定数に対して連結子変換を行った場合 (図 4.20) と行っていない場合 (図 4.21) で同じ幅 (710pixel) になっている 2 つの候補レイアウトを比較してみる。ここでは、2 つとも同時変換を行っているために、構造的にはそろったレイアウトであると考えられる。このように、あるユーザ表示画面に対して候補レイア

YEAR			
2006			
GROUP	COUNTRY	COACH	NATIONALITY
	Costa Rica	GUIMARAES Alexandre	Costa Rican
	COUNTRY	COACH	NATIONALITY
	Ecuador (ECU)	SUAREZ Luis	SUAREZ Luis
	COUNTRY	COACH	NATIONALITY
	Germany (GER)	KLINSMANN Juergen	German
	COUNTRY	COACH	NATIONALITY
A	Poland (POL)	JANAS Pawel	Polish

図 4.20: $P_B(AllString)$ によって生成されるレイアウト結果の例 (710pixel の幅)

YEAR					
2006					
GROUP	COUNTRY	Costa Rica	COACH	GUIMARAES Alexandre	
	NATIONALITY	Costa Rican			
	COUNTRY	Ecuador (ECU)	COACH	SUAREZ Luis	
	NATIONALITY	SUAREZ Luis			
	COUNTRY	Germany (GER)	COACH	KLINSMANN Juergen	
	NATIONALITY	German			
	COUNTRY	Poland (POL)	COACH	JANAS Pawel	
	NATIONALITY	Polish			
		COUNTRY	England (ENG)	COACH	ERIKSSON Sven Goran

図 4.21: $P_B(AllString)$ によって生成されるレイアウト結果の例 (710pixel の幅)

ウトが複数存在すると, 2つの候補レイアウトから最終結果レイアウトを選択する基準が必要になる. 次の第5章で詳しく述べるように, この場合は2つの候補レイアウトの充填率や変換された連結子の回数と, 第5章で提案する変換戦略の優先順位の比較によって最終結果レイアウトを選ぶ.

今まで述べたように, 全数探索によるレイアウト生成手法であってもユーザにとっては望ましくない結果を生成する場合が多くある. 文字列定数を用いた例で中括弧によるグルーピングに関して述べたように, 関連性が高い属性は統一的に変換を行うほうがユーザに見やすいレイアウトを提供できると考えられる. 予備実験で用いた文字列定数だけでなく, 中括弧でくれる属性間の変換については深く考えるべきであり, 第5章で新しいレイアウト変換手法を提案する.


```

GENERATE ACTIVIEW
[ { "GROUP"@{width=80} , t.group@{width=20} } ,
  [ { "COUNTRY"@{width=90} , t.country @{width=240} } ,
    [ { "NAME"@{width=60} , p.name@{width=260} } ,
      { "BIRTHDAY"@{width=100} , p.birth@{width=100} } ,
      { "CLUBS"@{width=70} , p.clubs@{width=260} } ,
      { "WEIGHT"@{width=80} , p.weight@{width=40} } ,
      { "HEIGHT"@{width=80} , p.height@{width=60} } ,
    ]!
  ]!
]!
FROM teams t, player p
WHERE ...

```

█ : 文字列定数 █ : オペレータ █ : 装飾子

図 4.22: 予備実験に使われた選手に関する問合せ文

4.4.9 Web 開発者の意味反映

図 4.22 の問合せ文からレイアウトを生成する際、図 4.23 (a) は開発者が指定した構造の意味を反映しながらレイアウト変換を適用した結果レイアウトで、図 4.23 (b) はレイアウトの変換によるユーザ表示画面の幅に収めることだけを基準にした場合である。図 4.23 (a) の開発者の構造化定義に従ってレイアウト変換を適用した結果レイアウトのほうが、より整理された感じの表を生成する。この変換によるレイアウト変換がユーザに対してより見やすいレイアウトの提供が可能であることは確実である。

図 4.24 はリンク変換の適用によって作成されたマルチページの例である。図 4.23 のように生成されたレイアウト結果が、開発者によって指定されている長さの範囲を超えないレイアウトの生成を目指す長さの条件を満たすために、このようにハイパーリンクによってマルチページが作成される場合が多い。

実験の結果から見るように、開発者のレイアウトの構造化定義に従って変換を適用した結果レイアウトがより見やすいレイアウトを生成するのは当然な結果であり、問合せ文の属性間の相関関係を考慮しない場合はパターンを基に変換を適用する手法がよりユーザ表示画面の幅に適応したレイアウトを生成する。これらの実験結果から、ユーザが見やすいように結果レイアウトが一貫性をもつようにレイアウトを変換する戦略について第 5 章で詳しく述べる。

(a) レイアウト変換に意味を反映した場合の結果レイアウト

GROUP	B		
COUNTRY	England(ENG)		
POSITION	FW	NAME	Michael OWEN
		BIRTHDAY	14/12/1979
		CLUBS	Newcastle Utd. (ENG)
		WEIGHT	67
		HEIGHT	172
		NAME	Peter CROUCH
	BIRTHDAY	30/01/1981	
	CLUBS	Liverpool (ENG)	
	WEIGHT	69	

(b) レイアウト変換に意味を反映しなかった場合の結果レイアウト

GROUP	B		
COUNTRY	England(ENG)		
POSITION	FW		
NAME	Michael OWEN	HEIGHT	172
BIRTHDAY	14/12/1979		
CLUBS	Newcastle Utd. (ENG)		
WEIGHT	67		
NAME	Peter CROUCH	HEIGHT	198
BIRTHDAY	30/01/1981		
CLUBS	Liverpool (ENG)		
WEIGHT	69		

(a) レイアウト変換に意味を反映した場合の結果レイアウト

(b) レイアウト変換に意味を反映しなかった場合の結果レイアウト

図 4.23: レイアウト変換を行う際の意味を反映

図 4.24: リンク変換の適用によって作成されたマルチページ

GROUP	B
COUNTRY	England(ENG)
POSITION	GK
	DF
	MF
	FW
COUNTRY	Sweden
POSITION	GK
	DF

NAME	Michael OWEN
BIRTHDAY	14/12/1979
CLUBS	Newcastle Utd. (ENG)
WEIGHT	67
HEIGHT	172
NAME	Peter CROUCH
BIRTHDAY	30/01/1981
CLUBS	Liverpool (ENG)
WEIGHT	69

図 4.24: リンク変換の適用によって作成されたマルチページ

第 5 章

レイアウト変換適用の戦略

ACTIVIEW では多様なユーザのアクセス環境，すなわち様々なユーザ表示画面のサイズに対して，属性間の関連性に基づいたユーザに見やすいレイアウトの動的な生成を目指す．まず，計算コスト削減による動的なユーザ対応を実現するため，ACTIVIEW システムで前処理を行うことによってレイアウト変換の候補個数を減らす手法を提案する．さらに，属性間の関連性を基に，制約と目標に従うレイアウトの変換戦略を用いて，ユーザに見やすいレイアウトの生成を実現する．

5.1 ACTIVIEW によるレイアウト変換

予備実験の結果によると，単純順序探索は速い時間で結果レイアウトの生成が可能であるが，生成する候補レイアウトの個数が少なくユーザに望ましい，見やすいレイアウトではなかった．これに比べて全数探索手法は，ユーザ表示画面の幅に最適化されたレイアウトの生成が可能だった．しかし，実行時間が非常にかかるのでユーザの要求に動的な対応ができない問題が生じた．さらに，幅制約だけを基準としているため，ユーザに見やすいレイアウトを提供するとは限らない．

そこで，本論文では，

- Web 開発の際，大きな負担になった多様なレイアウトの動的な生成を可能にするレイアウト変換コストの削減
- ユーザ環境に適応した見やすい Web ビューであると判断できる一貫性のあるレイアウトの提供

を目指したレイアウト変換手法を提案する．本論文での提案手法と従来のレイアウト変換手法の比較を表 5.1 に示す．

表 5.1: 各変換手法の比較

変換手法	制約	目標	生成可能な全レイアウト
単純順序探索		×	×
全数探索			
提案手法			

表 5.1 に示したように，本論文の提案手法は全数探索と比べて，生成可能な全レイアウトを対象にしていない．しかし，制約と目標を強化することで変換コストの削減

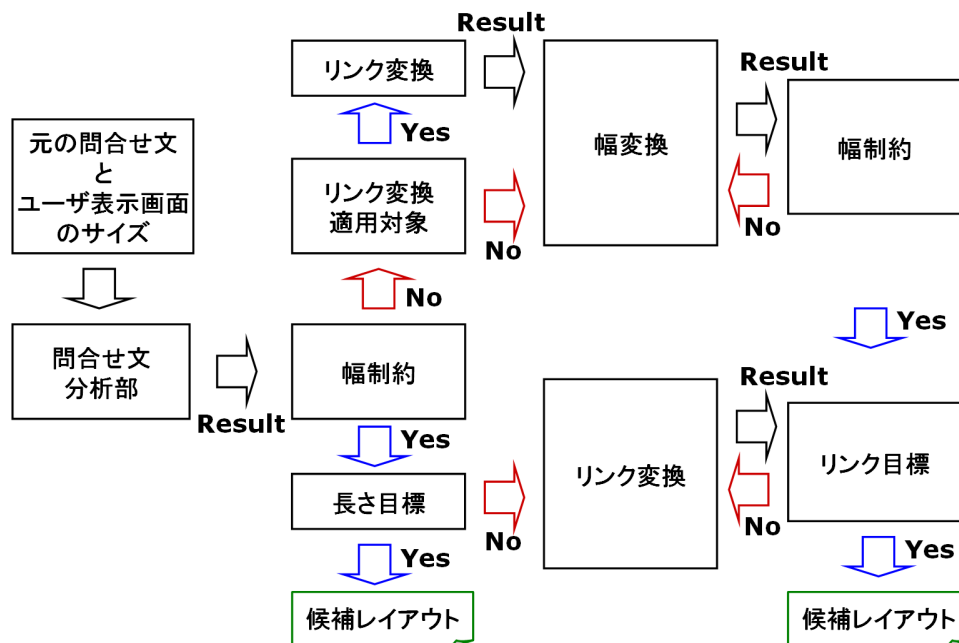


図 5.1: レイアウト変換の流れ

と、レイアウト変換戦略を用いて、ユーザにより良い結果レイアウトを適用する手法である。

本論文の ACTIVIEW システムでのレイアウト変換処理の流れを図 5.1 に示す。図 5.1 は 2.3 節 (p.2.3) で述べたシステムの“適応化処理部”内の処理を詳しく示したものである。

元の問合せ文とユーザ表示画面のサイズは“問い合わせ分析部”によってレイアウトの幅や長さ、充填率の計算に必要な情報を抽出する。これらの情報は、元の問合せ文を書き換えた結果から実際に生成されるレイアウト結果を予測する際に利用する。

元の問合せ文を分析した結果、幅制約を満たしていないレイアウトであれば、リンク変換を先に適用する対象になるかの判断を行う。次の条件を満たすとリンク変換を先に適用する対象になる。

- 開発者が指定している幅目標を超える。
- 最も多い子要素をもつネストの幅が初期レイアウトとユーザ表示画面の幅との差より広い。

この条件を満たすと、レイアウトの変換の際、いずれかリンク変換を適用することになる。そこで、先にリンク変換を行うことでレイアウト変換の範囲を分割し、変換

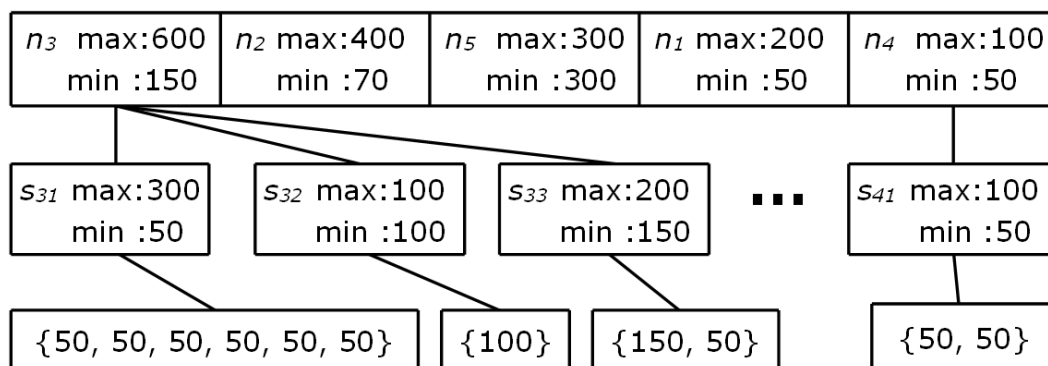


図 5.2: 変換対象の探索

対象になる連結子の個数を減らすことができる．リンク変換の戦略については5.2節で述べる．リンク変換を行ったレイアウトはリンク元とリンク先の複数レイアウトに分けられるため，それぞれのレイアウトに対し，幅変換を行う．幅変換を行った結果レイアウトが長さ目標を満たさない場合は，再びリンク変換を行う．

レイアウトの幅変換は，問合せ文分析部で生成した各ネスト別の情報をソーティングしたリスト（図 5.2）を基に変換対象になる連結子の探索を行う．たとえば，初期レイアウトとユーザ表示画面の幅が700pixelであると， n_3 と n_4 か， n_2 と n_5 ， n_2 と n_1 と n_4 の組のいずれかに変換を行うことになる．どの枝に探索を行うことがユーザにとって見やすいレイアウトを提供できるかを判断するレイアウト変換戦略は，5.3節と5.4節で述べる．

5.2 ACTIVIEW のリンク変換

ACTIVIEW では，計算コストを下げる手法として，長さ目標を満たさないレイアウトに対して“リンク変換”を先に行う手法と，“統一変換”が可能な連結に対しては同時に変換を行う手法を提案する．そして，一貫性のあるレイアウトを生成するために，2つの条件と3つの発見的な戦略を提案する．

5.2.1 リンク変換戦略の基本概念

初期レイアウトにリンク変換を先に適用することによって，変換の対象となる連結子の個数を分割により減らすことができる．リンク変換の適用は次の条件および優先順位に従う．

- リンク変換の前提条件

- (1) 反復子との連結子に適用することを前提にする。
- (2) 中括弧内のリンク変換は禁止する。

- リンク変換の優先順位

- (1) 多くの子要素をもつネストからの変換を優先する。
- (2) 長い結果テーブルをもつネストからの変換を優先する。
- (3) 深いレベルにあるネストからの変換を優先する。

これらのリンクへの変換手法は 4.4 節の実験結果に基づいた発見的な戦略である。実験から生成されるレイアウトの結果の検討に基づき、ユーザに一貫性のあるレイアウトが提供できる効率的な戦略を採用した。続いて各戦略を 1 つずつ詳しく述べながら、これらのリンク戦略の実際的な処理アルゴリズムを示す。

5.2.2 反復子との連結子にリンク

開発者が問合せ文に反復子を指定することによってネスト構造を定義する場合は、このネスト構造と連結されている属性は、他の属性の間より高い関連性をもつと判断する。すなわち、一般的にネストの木構造での親子関係では、子供ノードが意味的に、あるいは構造的に親ノードに包含される。さらに、関連性が高い属性が同一ネストに含まれることは意味的に近い内容であると判断できるので、この関連性を維持しなければならない。そこで、リンク変換を行う場合でもこの反復子による要素の集合に対しては同時変換を適用する。この関連性と包含関係を維持するためにリンク変換条件の (1) を前提とする。

たとえば、World Cup に出場したチームのトーナメント組 (t.group)、チーム名 (t.name)、選手のポジション (p.position)、選手名 (p.name)、選手の誕生日 (p.birth)、選手の身長 (p.height)、選手の体重 (p.weight) を検索した結果データを開発者が図 5.3 のように構造化したとする。図 5.3 の構造から判断できることは、開発者が選手に関する個人情報 (名前、生年月日、身長、体重) 間の関連性が一番高いと判断していることである。あるいは、選手に関する個人情報は同じ構造 (テーブル) の中でくりたい属性である。そして、図 5.3 のように各チーム (図 5.3 の②) はトーナメント組でネストされ (図 5.3 の①)、またチームに所属する選手の個人情報 (図 5.3 の④) は各ポジション別 (図 5.3 の③) にネストされるテーブルになる。この構造の包含関係が

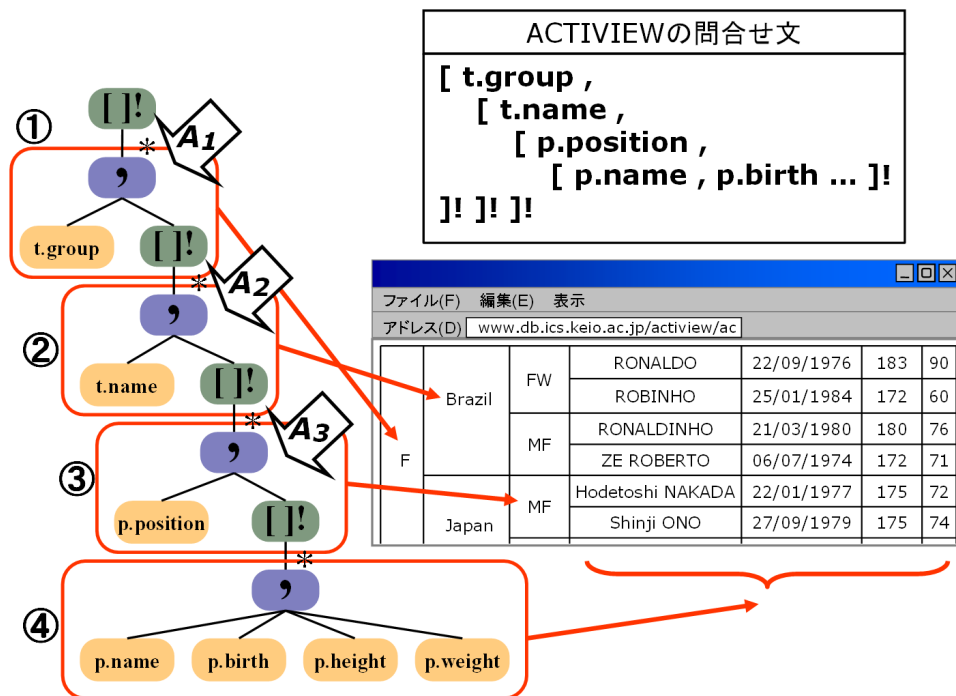


図 5.3: ネスト構造でのリンク優先

らリンク変換条件 (1) に基づくと、リンク連結子に変換される連結子の候補は図 5.3 の A_1, A_2, A_3 のいずれかになる。

図 5.3 の結果レイアウトから判断できるように、ネスト別にリンク変換が行われて新しいページになる場合のほうが、この新しく生成されるページの内容が 1 つのまとまりとなるので、ユーザにとっては見やすいページになる。したがって、関連性が高い属性は同時にレイアウト変換を行う戦略を適用することでユーザにとって直感的なレイアウトの提供ができると仮定する。

本論文での再設計を行った ACTIVIEW システムでは、適応化処理部を通じて元の問合せ文の分析を行い、ネスト定義 (反復子) と連結している連結子の情報のみを取り出したリスト (*connect_with_nest_list*) を用意する。

ネスト指定 ([]) と連結している演算子の情報をもつ *connect_with_nest_list* は、ネスト指定と連結している連結子の問合せ文への位置と、その位置¹の現れる連結子の種類、深さ、ネスト内に含んでいる属性の個数、包含しているネスト情報 (*subnest_info_list*) をもつリストである。

反復子との連結子に変換を行う際はこの *connect_with_nest_list* の情報を基にする。

¹ACTIVIEW システムによる元の問合せ文の分析によって生成される構造情報リスト内の位置を表す。

すなわち、このリストにある連結子に変換の候補になる。図 5.3 の問合せ文から生成される *connect_with_nest_list* は次のようになる。

```
connect_with_nest_list = [ [ 3, " , ", level1, 1, subnest_info_list ],  
                          [ 6, " , ", level2, 1, subnest_info_list ],  
                          [ 9, " , ", level3, 1, NULL ] ]
```

5.2.3 中括弧内でのリンク変換の禁止

中括弧内の属性は関連性が高いと仮定しているため、この間にはリンク変換を行うと開発者が意図した意味的な構成が崩れる。ACTIVIEW システムでは、中括弧でくくられてグルーピングされた属性の場合は、これらの属性を 1 つの集合要素として扱う。

図 5.4 に示すように中括弧でくくられている属性は意味的に関連性が高いと仮定する。各中括弧は選手、国（チーム）、コーチに対する情報別の集合である。したがって、選手に対する検索結果の集合である中括弧内でリンク変換を行う（図 5.4 の①）と意味関連性が崩れたレイアウトを生成してしまう。

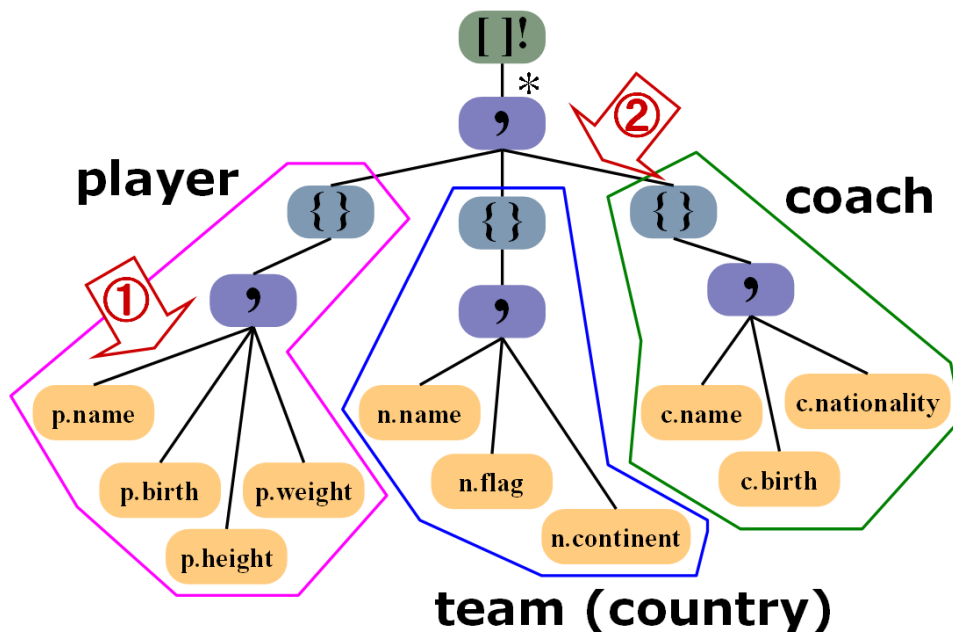


図 5.4: 中括弧内でのリンク変換

ほかに、同一ネスト内の中括弧の間の連結子にリンク変換を適用する場合（図 5.4 の②）は、リンク元になる属性をどう決めるかの問題が生じてしまう。この場合は前にある属性にリンクを貼り付ける。ネスト内でのリンク変換はリンク先になる属性が判別できない場合、すなわち、横連結などに単純に連結されている場合はネスト内でのリンク変換を行わないことを戦略とする。

ACTIVIEW システムでは中括弧内に現れる連結子の情報を別のリストで扱っているので、このリストにある連結子はリンク変換の対象外になる。

5.2.4 多くの子要素をもつネストの優先

構造化される各ネストの包含関係によって、代表ネスト、サブネスト、コンテンツネストと分類する。各ネスト分類は次のように定義する。

定義 5.1 (代表ネスト)

木構造での親ノードに当たるネストである。これに含まれるネストの内容を代表するコンテンツ、あるいは属性を持っているか、分類の上位概念のコンテンツ、あるいは構造的に子供ネストを含むネストを意味する。

定義 5.2 (サブネスト)

木構造での子供ノードに当たるネストである。あるネストに含まれ、反復的に現れるネストを意味する。サブネストは自分が含む子供ネストの親ネストにもなる。

定義 5.3 (コンテンツネスト)

重複して現れる検索結果をもっていない、あるいは重複して現れる検索結果が数少ないネストを意味する。検索した結果がほぼ反復されないことから、情報になるコンテンツ、あるいは下位概念の細かいコンテンツをもっていると判断されるネストである。たとえば、論文の目次より内容をもつ部分に当たるネストである。コンテンツネストは自分を含むネストのサブネストにもなる。

この定義から、開発者が問合せ文を作成する際に、多くの子要素をもつネストを定義する場合は、このネストが代表ネストであるよりもコンテンツネストである確率が高いと判断する。

図 5.3 の選手に関する情報の集まりであるネスト（図 5.3 の④）は、多くの情報をもつネストであることが分かる。初期のページでは多くの内容を並べるより、直感的に分かる代表性をもつ情報を前面に取り出して見せる方が良いと仮定している。したがっ

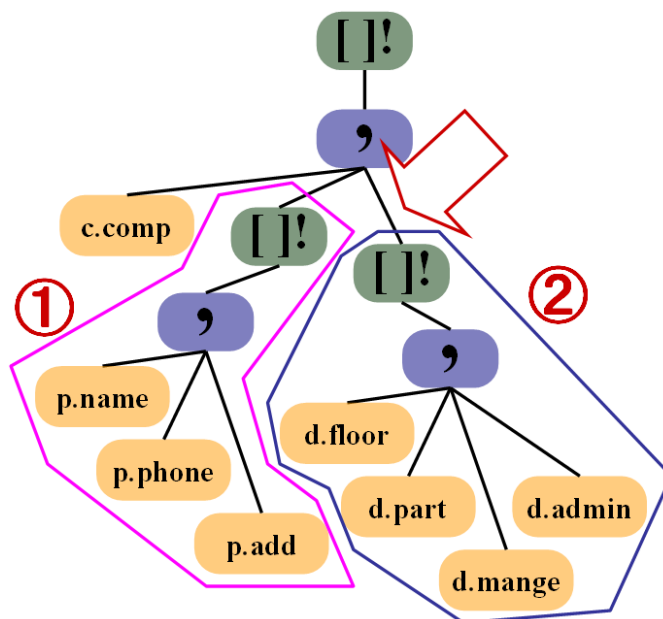


図 5.5: 多くの子要素をもつネスト優先変換戦略

て、図 5.5 の①と②のネストが同じ長さをもつネストだとすると属性数が多い図 5.5 の②ネストを優先にリンク変換を行う（図 5.5 の矢印）。

これによって、複雑な内容をもつレイアウトはリンク先のページに分けられ、コンテンツの量を落とした省略されたレイアウトの作成が可能になる。この戦略によると少ない変換で幅を大きく縮めることもでき、内容的に統一された結果レイアウトの生成が可能になる。

多くの子要素をもつネストは 5.2.6 項のアルゴリズム 5.1 で述べる各ネストがもつ連結子の情報を含むリスト, *nest_attribute_list* から計算できるので、この計算結果にあたるネストとの連結子の情報を 5.2.2 項で述べた *connect_with_nest_list* から用いることで戦略を適用できる。

5.2.5 長いネストの優先

反復子との連結子にリンク変換を適用する戦略で、複数の候補連結子を取り上げられた場合、これらの候補からどの連結子を優先するかを決めなければならない。

入れ子構造から長いネストを優先することで少ない変更によって長さ目標を満たすことが期待できる²。さらに、リンク元のページでは多くの内容が並べられるより、代

²少ない変換回数でレイアウトの長さが大きく変更される。

id	position	name	birth	height	weight
1	GK	Yoshikatsu KAWAGUCHI	15/08/1975	179	78
2	DF	Teruyuki MONIWA	08/09/1981	181	77
3	DF	Yuichi KOMANO	25/07/1981	171	71
4	DF	Tsuneyasu MIYAMOTO	07/02/1977	176	72
5	DF	Koji NAKATA	09/07/1979	182	74
6	FW	Naohiro TAKAHARA	04/06/1979	180	74
7	FW	Seiichiro MAKI	07/08/1980	184	77
8	FW	Atsushi YANAGISAWA	27/05/1977	177	75
9	MF	Yasuhito ENDO	28/01/1980	177	75
10	MF	Hidetoshi NAKATA	22/01/1977	175	72
...
23	MF	Shunsuke NAKAMURA	24/06/1978	178	69

↓	↓	↓	↓	↓
4	23	23	14	13

図 5.6: データ代表要素個数

表性をもつコンテンツを見せた方が直感的である。すなわち、リンク変換によってリンク先になるページはコンテンツネストのほうが望ましい。あるネストからコンテンツネストと予想されるネストを抽出するため、元の問合せ文から生成される各ネストのデータの代表要素個数を計算して各ネストの長さを予測する。

そこで、データ代表要素個数、 IC を導入する。リレーション r にある属性 t における検索結果のうち、重複して出現する要素の個数 IC_t は、

$$IC_t = \mathcal{G}_{count-distinct}(t)(r) \quad (5.1)$$

となる。関係代数演算 \mathcal{G} は集約演算で、下付き添字では適用される演算の種類を示す [29]。 IC_t リレーション r に現れる属性 t の個数に関係なく、重複した結果は削除した値のみ返す。すなわち、属性 t にある検索した結果値の種類が検索される。選手リレーション (player) の検索結果である図 5.6 のデータから position カラムデータの代表要素個数の計算式は、

$$IC_{position} = \mathcal{G}_{count-distinct}(position)(player)$$

になり、position カラムデータの中で重複して現れる結果から重複を削除した、“GK”、“DF”、“FW”、“MF”の個数、“4”が結果値として返される。他のカラムもそれぞれデータ代表要素個数を計算した値を図 5.6 に示す。

このテーブルから、position カラムによると 23 件のレコードを 4 つに分類できると考えられる。すなわち、position カラムが“代表ネスト”になる可能性があることを意味する。他の height カラムなどは 23 件が 14 件に分類されることになり、このカラム

表 5.2: 各ネストにある属性 t の IC_t 計算結果

属性 t	検索結果	IC_t
t.group	A, B, ... , H	8
t.name	Korea, Japan, ... , England	32
p.position	GK, MF, FW, DF	4
p.name	NAKATA, ...	736

向けにほかの情報 (カラム) をネストする構造は望ましくない。すなわち、データの省略率 $TS(t)$ が高くないことが分かる。

$$TS(t) = \left(1 - \frac{IC_t}{\mathcal{G}_{count(t)}(r)} \right) \times 100 \quad (5.2)$$

$\mathcal{G}_{count(t)}(r)$ はリレーション r にある属性 t の数を返す。図 5.6 のデータから position カラムの省略率を計算すると、

$$\begin{aligned} TS(position) &= \left(1 - \frac{IC_{position}}{\mathcal{G}_{count(position)}(player)} \right) \times 100 \\ &= \left(1 - \frac{4}{23} \right) \times 100 \end{aligned}$$

で 83% になり、height カラムの省略率は 39% になる。

これらの省略率が低いカラム (ネスト) がコンテンツネストの候補になり、省略率が高いカラムが代表ネストの候補になる。

開発者は利用するデータベースの状態を良く知ったうえで、検索結果の構造化指定を行うと仮定する³。

図 5.3 のデータ代表要素個数を表 5.2 に示す。この表は FIFA ホームページ [30] のデータのうち、2006 年のデータに基づいて作成された結果表である。

5.2.6 ネストの長さ予測アルゴリズム

長いネストを優先する戦略を適用するためには、まず各ネストの長さを知らなければならぬ。各ネストの長さの計算はレイアウトの長さを計算する際に行うので、まずレイアウトの長さを計算するアルゴリズムを述べる。

³name カラムに position の結果がネストされる構造を指定することは妥当ではない。

本論文では ACTIVIEW システムを再設計して元の問合せ文から生成されるレイアウトの長さを予測する手法を提案する。

5.2.5 項で述べた *IC* を用いてレイアウトの長さを予測するアルゴリズムをアルゴリズム 5.1 に示す。

レイアウトの長さを予測するために適応化処理部 (2.3 節) は、まず問合せ文に現れる属性を各ネスト別にリスト化した *nest_attribute_list* を生成する。図 5.3 を基に作成した *nest_attribute_list* の例を次に示す。

```
nest_attribute_list =  
    [ [ level0, t.group ], [ level1, t.name ], [ level2, t.positon ],  
      [ level3, p.name, p.birth, p.height, p.weight ] ]
```

このリストから各ネスト別のリストを取り出した *sub_attribute_list* を生成する (1 行目)。各 *sub_attribute_list* は

```
[ level0, t.group ]  
[ level1, t.name ]  
[ level2, t.positon ]  
[ level3, p.name, p.birth, p.height, p.weight ]
```

のようになる。

5.2.5 項で述べた *IC* の計算手法に従って適応化処理部で生成する各属性の *IC* 値のリスト, *IC_list*

```
IC_list = [ [ t.group, 8 ], [ t.name, 32 ], [ t.positon, 4 ],  
            [ p.name, 736 ], [ p.birth, 685 ], [ p.height, 40 ], [ p.weight, 40 ] ]
```

から *sub_attribute_list* の各属性にあたる *IC* 値を取り出して *nest_attribute_IC_list* を生成する (2 行目から 3 行目)。

IC_list と各 *sub_attribute_list* から生成した *nest_attribute_IC_list* は次のようになる。

```
nest_attribute_IC_list =  
    [ [ level0, [ t.group, 8 ] ], [ level1, [ t.name, 32 ] ], [ level2, [ t.positon, 4 ] ],  
      [ level3, [ p.name, 736 ], [ p.birth, 685 ], [ p.height, 40 ], [ p.weight, 40 ] ] ]
```

nest_attribute_IC_list の level3 のリストのように同一ネスト内で存在しながら異なる *IC* 値をもつ属性が現れる場合がある。しかし、同一ネスト内の属性の検索結果はそ

アルゴリズム 5.1 ネストの長さを計算

```
1: nest_attribute_list より各リスト別のエレメントリスト sub_attribute_list を取り出す
2: IC_list より nest_attribute_list の各エレメントの IC 値を取り出す
3: 1 行目と 2 行目から nest_attribute_IC_list =  $\langle l_1, l_2, \dots, l_n \rangle$  を生成
4: for nest_attribute_IC_list 順に  $l_i$  をチェック do
5:   nest_attribute_IC_list の属性の IC 値を一番大きい IC 値に変換
6: end for
7: while nest_attribute_IC_list  $\neq \emptyset$  do
8:   ネスト  $l_i$  の親ネストのリストを parent_nest =  $\langle p_1, p_2, \dots, p_n \rangle$  とする
9:   while  $p_i \neq \emptyset$  do
10:    if  $l_i$  の IC 値  $>$  ( $p_i$  の IC 値  $\leftarrow$  sub_nest_attribute_repeat) then
11:      sub_nest_attribute_repeat  $\leftarrow$   $l_i$  の IC 値  $\div$   $p_i$  の IC 値
12:    else
13:      sub_nest_attribute_repeat  $\leftarrow$   $l_i$  の IC 値
14:    end if
15:  end while
16: end while
17: while sub_nest_attribute_repeat =  $\langle s_1, s_2, \dots, s_n \rangle$  の属性  $s_i \neq \emptyset$  do
18:  if  $s_i$  と連結される  $s_{i+1}$  が横連結である then
19:    sub_nest_length  $\leftarrow$   $s_i$  と  $s_{i+1}$  の中で大きい値
20:  else
21:    sub_nest_length  $\leftarrow$   $s_i + s_{i+1}$ 
22:  end if
23: end while
24: while sub_nest_length  $\neq \emptyset$  do
25:  if sub_nest_length の親ネストと子供ネスト (child_nest) が横連結である then
26:    layout_length  $\leftarrow$  parent_nest の値  $\times$  child_nest の値
27:  else
28:    layout_length  $\leftarrow$  parent_nest の値  $+$  (parent_nest の値  $\times$  child_nest の値)
29:  end if
30: end while
31: return layout_length  $\times$  実際のセルの高さ値
```

の中の最大値に従うため、最も大きい *IC* の値にすべて変換する（4 行目から 5 行目）。すなわち、“ p.height ”の *IC* 値は 40 であるが“ p.name ”と同じネスト内にいるので生成されるレイアウトでは“ p.name ”の検索結果と同じく 736 個のセルを生成する。したがって、level3 の *nest_attribute_IC_list* の値は次のように変換される。

```
[ level3, [ p.name, 736 ], [ p.birth, 736 ], [ p.height, 736 ], [ p.weight, 736 ] ]
```

ネスト構造を定義している元の問合せ文から生成されるレイアウトの長さを予測するためには、ネスト定義によってあるサブネストが何回に反復的に現れるかを予測する必要がある（7 行目から 16 行目）。ある *nest_attribute_IC_list* を包含する親ネストが存在すると（8 行目）、包含されるサブネストでは包含している代表ネストの *IC* の回数だけ反復されると判断する。すなわち、サブネストにある属性による検索結果は代表ネストの検索結果の *IC* 値によって割られて構造化される。

nest_attribute_IC_list の level0 は親ネストがないのでそのまま 8 の値になる。level1 の *IC* 値は親ネスト level0 の *IC* 値である 8 より大きい 32 であるので 8 で割り算した結果値 4 が *sub_nest_attribute_repeat* に入れられる。このように、*sub_nest_attribute_repeat* は入れ子構造によって反復される各サブネストの反復回数を予測した値のリストである。続いて、level2 の *IC* 値は 4 なので、親ネストの *IC* 値より大きい値ではない。この場合は 13 行目に従ってそのまま 4 になる。これは、包含関係で同じ値が反復的に現れる場合である（各チーム別にポジション名が反復的に現れる）。

level3 は *sub_nest_attribute_repeat* から再計算した親ネストの *IC* 値である 8, 4, 4 の値で割り算した 5.75 が *sub_nest_attribute_repeat* の値になる（11 行目）。

したがって、生成される *sub_nest_attribute_repeat* は次のようになる。

```
sub_nest_attribute_repeat =  
[ [ level0, [ t.group, 8 ] ], [ level1, [ t.name, 4 ] ], [ level2, [ t.positon, 4 ] ],  
  [ level3, [ p.name, 5.75 ], [ p.birth, 5.75 ],  
    [ p.height, 5.75 ], [ p.weight, 5.75 ] ] ]
```

初期レイアウトの長さを計算するために、各ネスト内の属性間の連結関係を調べて *sub_nest_length* を生成する（17 行目から 23 行目）。

複数の属性をもっているネスト level3 の長さはこのネストに含まれている属性がすべて横連結になっているため、アルゴリズム 5.1 の 19 行目に従う。level3 の属性はすべて横連結されているので、5.75 の値を返す⁴。したがって、生成される *sub_nest_length* は次のようになる。

⁴横連結であっても中括弧の定義によってくくられた部分に現れる属性が縦連結になっている場合は計算値が変わる

$$sub_nest_length = [[level0, 8], [level1, 4]], [level2, 4], [level3, 5.75]]$$

図 5.3 の p.name と p.birth 属性が縦連結に変換されると、アルゴリズム 5.1 の 21 行目に従って $5.75+5.75$ の値が返されて *sub_nest_length* は次のように変わる。

$$[[level0, 8], [level1, 4]], [level2, 4], [level3, 11.5]]$$

sub_nest_length を用いて初期レイアウト (図 5.3) の長さを計算すると (24 行目から 30 行目) 各ネストがすべて横連結されているので、

$$\begin{aligned} 8 \times 4 &\rightarrow 32, & 32 \times 4 &\rightarrow 128, \\ 128 \times 5.75 &\rightarrow 736 \times \text{実際のセルの高さ.} \end{aligned}$$

の値がレイアウトの長さになる。図 5.3 の A_3 が縦連結に変換されると、

$$\begin{aligned} 8 \times 4 &\rightarrow 32, & 32 \times 4 &\rightarrow 128, \\ 128 + (128 \times 5.75) &\rightarrow 864 \times \text{実際のセルの高さ.} \end{aligned}$$

の長さをもつレイアウトに変換される。

5.2.1 項で述べたレイアウトの変換にリンク変換戦略、

(2) 長い結果テーブルをもつネストからの変換を優先する。

を適用する際は、*sub_attribute_IC_list* から一番長いネストであると予想される level3、すなわち図 5.3 の④のネストをリンク先のページに分けるようになる。この戦略を適用すると、図 5.3 の A_3 の連結子がリンク連結子に変換されて、生成されるレイアウトのリンク元のレイアウトの長さは、

$$8 \times 4 \rightarrow 32, \quad 32 \times 4 \rightarrow 128 \times \text{実際のセルの高さ.}$$

に変わる (図 5.7)。

すなわち、実際のセル 1 つの長さが 30pixel と仮定すると、初期レイアウト長さ 22,080pixel (736×30) が、リンク変換の適用によってリンク元のレイアウトの長さ 3,840pixel (128×30) に縮まる。

このように、元の問合せ文からデータ代表要素個数を計算することによって実際のレイアウト結果を生成せずに、問合せ文から取り出した情報を組み立ててみる段階でレイアウトの長さを予測することで、処理コストの削減が可能になる。実際の ACTIVIEW 処理では TFE による長さ定義 (たとえば HTML の height タグの指定によるセルの高さ定義) を利用する。開発者による長さ定義がない場合は、検索結果の一文字の高さを計算した値を使う。

実際データによって定義されたセルの幅に収まらずに、1 行以上になる場合もありうる。この場合は予測した長さより長いレイアウトを生成してしまうが、本論文の ACTIVIEW では無視しているため、実際生成されるレイアウトの長さとは少し異なる場合が生じる。

[t.group , [t.name , [p.position %
[p.name , p.birth ...]!
]!]!]!

Position	Name	Birth	Value 1	Value 2
MF	Yasuhito ENDO	28/01/1980	177	75
FW	Hidetoshi NAKATA	22/01/1977	175	72
DF	Mitsuo OGASAWARA	05/04/1979	173	72
GK
MF	Junichi INAMOTO	18/09/1979	180	76
FW	Shinji ONO	27/09/1979	175	74

図 5.7: 長いネスト優先リンク変換戦略

5.2.7 奥にあるネストの優先

開発者は問合せ文を作成する際、分類性が高いコンテンツをもつ属性を前面に指定し、個別性が高い属性は奥の方に、あるいはネスト構造では上位カテゴリ（上位の意味性をもつ）の属性をネスト構造の親ノードの方に指定するのが一般的である。したがって、奥にあるネストを優先する戦略を設定した。

図 5.8 に複雑な深さをもつ構造のときのリンク変換の優先順位を示す。同じ深さ（図 5.8 の同じ番号）のネストに対しては今まで述べた、長いネストの優先変換、多くの属性をもつネストの優先変換に従って変換を行う。

ネストの深さは 5.2.6 項で述べた *nest_attribute_list* から獲得できる。

5.2.8 リンク変換適用アルゴリズム

本項では今まで述べたリンク変換戦略に従う変換処理のアルゴリズム 5.2 の詳細を示す。初期レイアウト及び結果レイアウトが長さ目標を満たしていないレイアウトである場合、5.2.1 項のリンク変換戦略の基本概念で述べたレイアウト変換の優先順位に従ってリンク変換を行う。

5.2.1 項の通りに、*nest_attribute_list* と *sub_attribute* の各サブリストを用いて（1 行目と 2 行目）リンク変換の対象になるネストを探索する。

ACTIVIEW システムは、長い結果テーブルをもつネストを予測する基準である *IC*

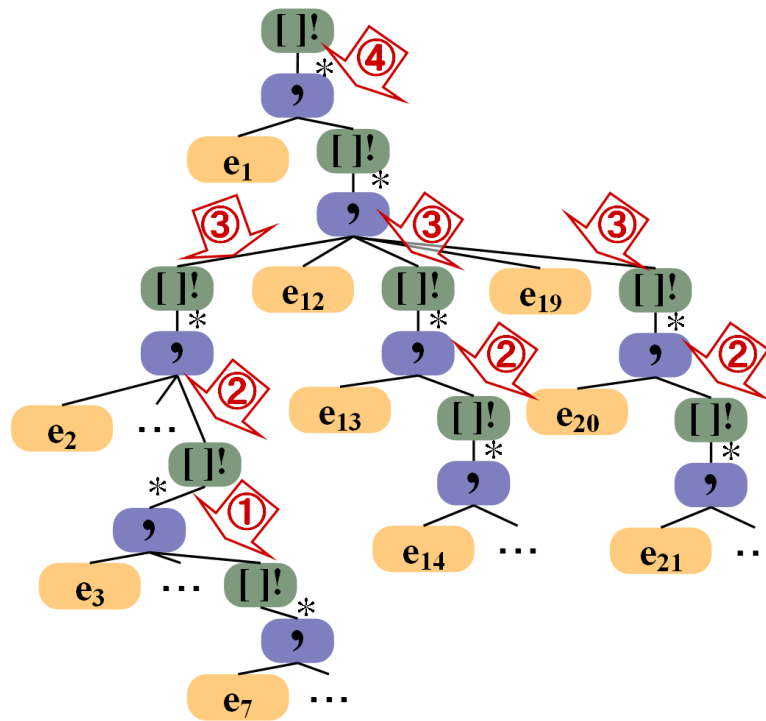


図 5.8: 奥のネスト優先変換戦略

アルゴリズム 5.2 リンク変換アルゴリズム

- 1: $nest_attribute_list$ から $sub_attribute_list = \langle l_1, l_2, \dots, l_n \rangle$ を生成
 - 2: $nest_attribute_IC_list = \langle i_1, i_2, \dots, i_n \rangle$ を生成
 - 3: **while** $i_i \neq \emptyset$ **do**
 - 4: $rank_{ICvalue} \leftarrow i_i$ の IC 値の大きい順に, ランキングする
 - 5: **end while**
 - 6: **while** $l_i \neq \emptyset$ **do**
 - 7: $rank_{AttributeNum.} \leftarrow l_i$ の属性の個数が多い順に, ランキングする
 - 8: $rank_{Nest} \leftarrow l_i$ のネストの深さが深い順に, ランキングする
 - 9: **end while**
 - 10: **while** 各 $rank \neq \emptyset$ **do**
 - 11: $condidacy_info_list \leftarrow$ 各 $rank$ を総合的にランキングする
 - 12: **end while**
 - 13: $change_to_link(condidacy_info_list$ のランク 1 位のネストとの連結子)
-

(データ代表要素個数)を長さ順でソーティングした結果(3行目から5行目)と、多くの子要素をもつネスト順にソーティングした結果(7行目)、深いネスト順にソーティングした結果(8行目)をそれぞれランキングしたものを用意する。

これらの各値によってランクされた結果から総合的にランクを行った結果リストを生成する(11行目)。結果リスト(*condidacy_info_list*)の一番目のランクのネストとの連結子(5.2.2項参照)にリンク変換を行う(13行目)。

続いて本論文で提案する統一変換と一貫性のあるレイアウトを生成する戦略とその処理アルゴリズムについて述べる。

5.3 統一変換可能な連結

開発者が指定した元の問合せ文に統一変換が可能な連結が存在すれば、同時に変換を適用することで、実質的に変換の対象になる結合子の個数を減らすことができ、計算コストの削減が可能になる。経験的に属性間の関連性が高い連結に対して統一変換を行うことにより、一貫性をもつレイアウトを生成する可能性が高い(4.4.2項)。たとえば、同一中括弧内の属性のように関連性が高い連結において、同時に変換を行って生成されたレイアウトが、連結子それぞれに対し、全数探索によって目標の結果値を最大化するよりも、構造的には一貫性のあるレイアウトとなる場合がある。

図 5.9 の場合、属性間の関連性を考えず全数探索の手法でレイアウトを生成すると、player 領域に含まれている 4 つの属性を連結する 3 つの連結子(図 5.9 の P_1, P_2, P_3)、nation の 2 つ(図 5.9 の N_1, N_2)、coach の 2 つ(図 5.9 の C_1, C_2)と、3 つの中括弧の間の 2 つ(図 5.9 の B_1, B_2)の連結子に対して変換を行い、 3^9 個のレイアウトパターンが生成される。4.4.1 節では n 個の演算子をもつ問合せ文から構成されるレイアウトの数 $P_B(n)$ は $P_B(n) = 3^n$ になるが、提案手法ではリンク連結子(%)の適用は特別の戦略に従うため、変換パターン数は横と縦変換の 2 種類として 2^9 の 512 パターンになる。しかし、中括弧に含まれている領域を 1 つのグループとして扱うと、各中括弧で含まれている領域に統一的に変換を適用するかしないかの各 2 通りが 3 つと、中括弧の間の 2 つの連結子に対する変換を含む 2^5 個でレイアウトの変換を行う。このため全数探索より計算コストの削減が可能になるとともに、一貫性のあるレイアウトが生成可能であると考えられる。

すなわち、統一変換可能な連結は、

同一ネスト・中括弧内の文字列定数：同一変換を行うことにより一貫性のあるレイアウトの生成ができると判断する。

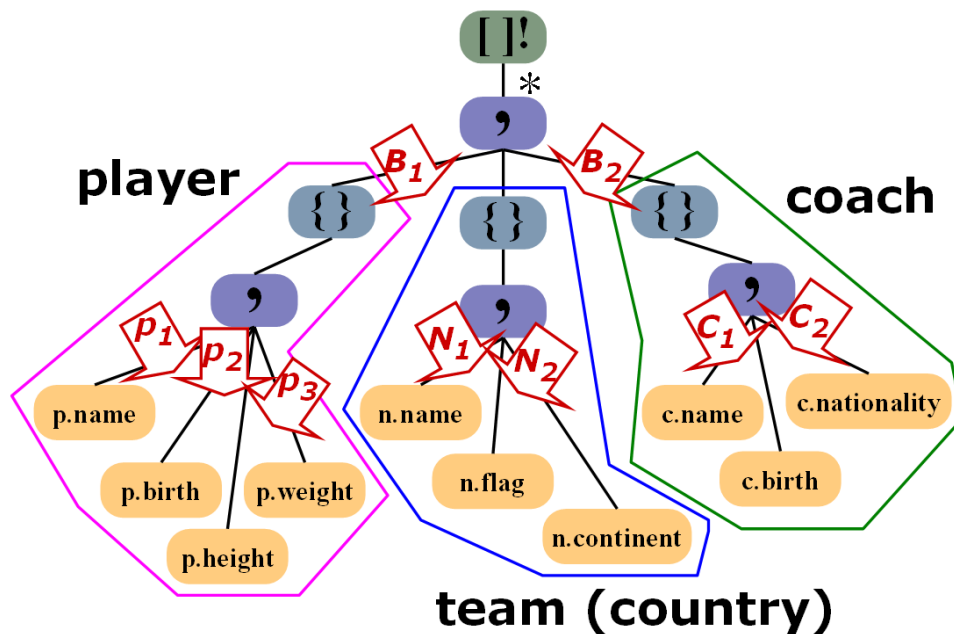


図 5.9: 統一変換可能な連結

同一中括弧内連結子：レイアウトの一貫性を維持できると判断する。

が考えられる。

ACTIVIEW システムでは統一変換を行うために、統一変換の候補になる連結子、すなわち同一中括弧内の連結子、同一ネスト内の連結子別にリストを用意する。レイアウトを変換するために本論文で提案するレイアウト変換戦略を適用する際にこのリスト情報に基づく。たとえば、図 5.9 のような木構造を生成する問合せ文が次のようにあるとする。

```
[ ... [ ... [ ... [ { p.name, p.birth, p.height, p.weight },
                    { n.name, n.flag, n.continent },
                    { c.name, c.birth, c.nationality } ]! ]! ]! ]!
```

この問合せ文から生成される同一中括弧内の連結子のリストは次のようになる。

```
connect_in_brace_list_each_nest = ...
[ level3, [ [ 14, " , " ], [ 16, " , " ], [ 18, " , " ] ],
          [ [ 24, " , " ], [ 26, " , " ] ],
          [ [ 32, " , " ], [ 34, " , " ] ] ]
```

connect_in_brace_list_each_nest は、元の問合せ文を ACTIVIEW システムがレイアウト変換戦略を適用する際に利用する情報のみを分析して、情報単位で分けて格納している *structure_info_list* (5.4.7 項で述べる。) から、ネスト別にネストに現れる中括弧内の連結子の情報のみを取り出して生成したリストである。

例のように *connect_in_brace_list_each_nest* では、ある連結子が *structure_info_list* で現れる順番と、連結子の種類をネスト (level 値) 別と中括弧別に格納している。たとえば、図 5.9 の P_1 , P_2 , P_3 に同時変換を適用する場合は *connect_in_brace_list_each_nest* から取り上げられる番号 14, 16, 18 番目にある連結子に同時変換を行うことになる。

連結子の統一変換はレイアウトの処理コストとの関係より、レイアウトの一貫性との関係を重要としているので、一貫性のあるレイアウトを生成する戦略について次の節で詳しく述べる。

5.4 一貫性のあるレイアウトの生成

ACTIVIEW での一貫性のあるレイアウトは、次の優先順位に従い、6 つの戦略によって生成される。これらの優先順位は予備実験での結果レイアウトの比較を行った結果に基づく発見的な手法である。

- (1) 複数の領域 (TFE 定義による) 内で行う変換より同一領域内 (同一ネスト内・同一中括弧) での変換を優先する。
- (2) 各属性間の変換よりも入れ子や中括弧との連結子への変換を優先する。
- (3) 同一ネスト内では親ノードの方を優先する。
- (4) 変換の回数が少ないレイアウトを優先する。
- (5) 中括弧内の連結子は統一変換を優先する。
- (6) これ以外は、全数探索によって生成されるレイアウトのうち、高い目標の結果値をもつレイアウトを優先する。

これらの戦略は制約を満たしながら適用される。これらの一貫性のあるレイアウト生成戦略について一つずつ詳しく述べる

5.4.1 同一領域内（同一ネスト内・同一中括弧）での変換の優先

制約を満たしたレイアウトであれば，複数のネスト内で行った変換によるレイアウト結果よりも，同一ネスト内での変換で構成されるレイアウト結果を優先する．

たとえば，目標の結果値が同じであるレイアウト結果が複数あれば，図 5.10 の A のように複数のネスト内で変換が行われたレイアウト結果よりも（図 5.11），B のように同一ネスト内で変換を行って構成されたレイアウト（図 5.12）を結果レイアウトとして提供する．

ACTIVIEW では，同一領域内での変換が，複数の領域での変換に比べて，全体レイアウトに与える影響が少ないと仮定している．

図 5.11 のように複数のネスト内で変換が行われた結果レイアウトは表構造が複雑になっており，ユーザにとって見やすい表とはいえない．それと比べて，図 5.12 は統一された感じの結果レイアウトを生成している．

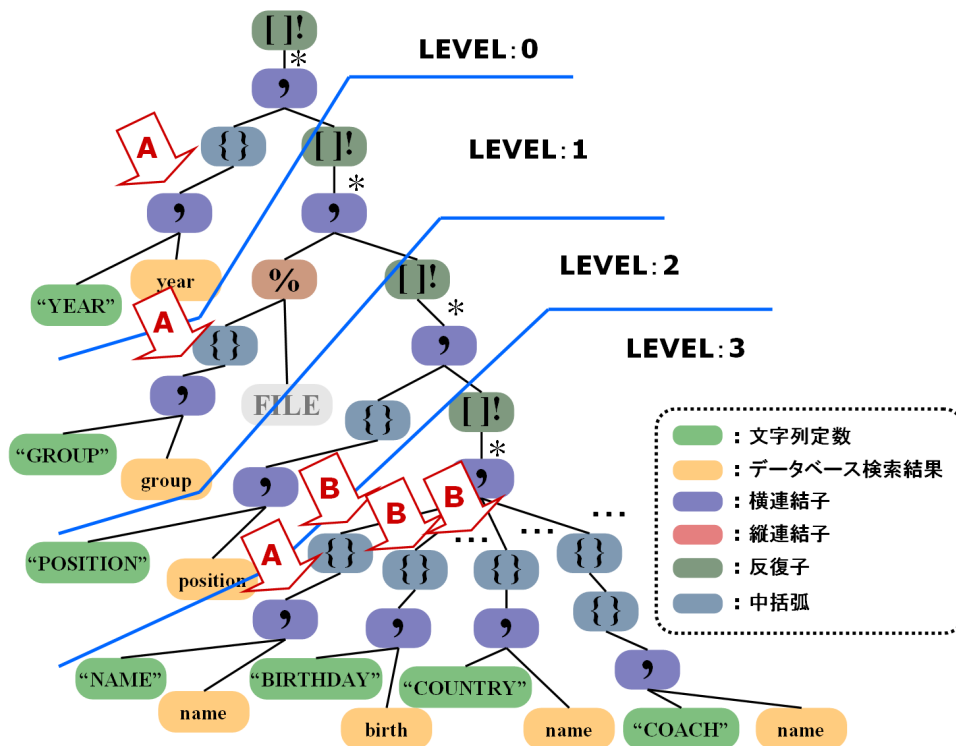


図 5.10: 複数のネスト内と同一ネスト内での変換の木構造例

YEAR	Group	POSITION	DF	NAME	BIRTHDAY	HEIGHT
2006	A		DF	Jervis DRUMMOND	08/09/1976	172
				Jorge GUAGUA	28/09/1981	179
				Marcell JANSEN	04/11/1985	190

図 5.11: 複数のネスト内で変換が行われた結果レイアウトの例

YEAR	2006	Group	A	POSITION	DF	NAME	BIRTHDAY	HEIGHT
						Jervis DRUMMOND	08/09/1976	172
						Jorge GUAGUA	28/09/1981	179
						Marcell JANSEN	04/11/1985	190

図 5.12: 同一ネスト内で変換が行われた結果レイアウトの例

5.4.2 反復子（ネスト指定）、グルーピングとの結合の優先

属性と属性の間の連結子で行われた変換よりも、反復子、連結子またはグルーピングされた属性を意味する中括弧との連結子に対し、変換を行うことを優先する。これは、入れ子内の属性の構造が変換され、レイアウト全般に反復されながら影響を与えることにより、反復子と中括弧との連結子に変換が適用され、入れ子や中括弧でくくられた属性が全体的に変換される方が、レイアウトへの影響は少ないという経験則に基づく。ACVTIVIEW では、属性間の連結子よりも入れ子や中括弧との連結子が構造的には弱いと判断することで、これを実現する。

図 5.13 は元の問合せ文（図 2.12）から生成される木構造でのネスト内にある連結子に変換を行う際（図 5.13 の矢印 A）と反復子との連結子に変換を行う際（図 5.13 の矢印 B）の例である。

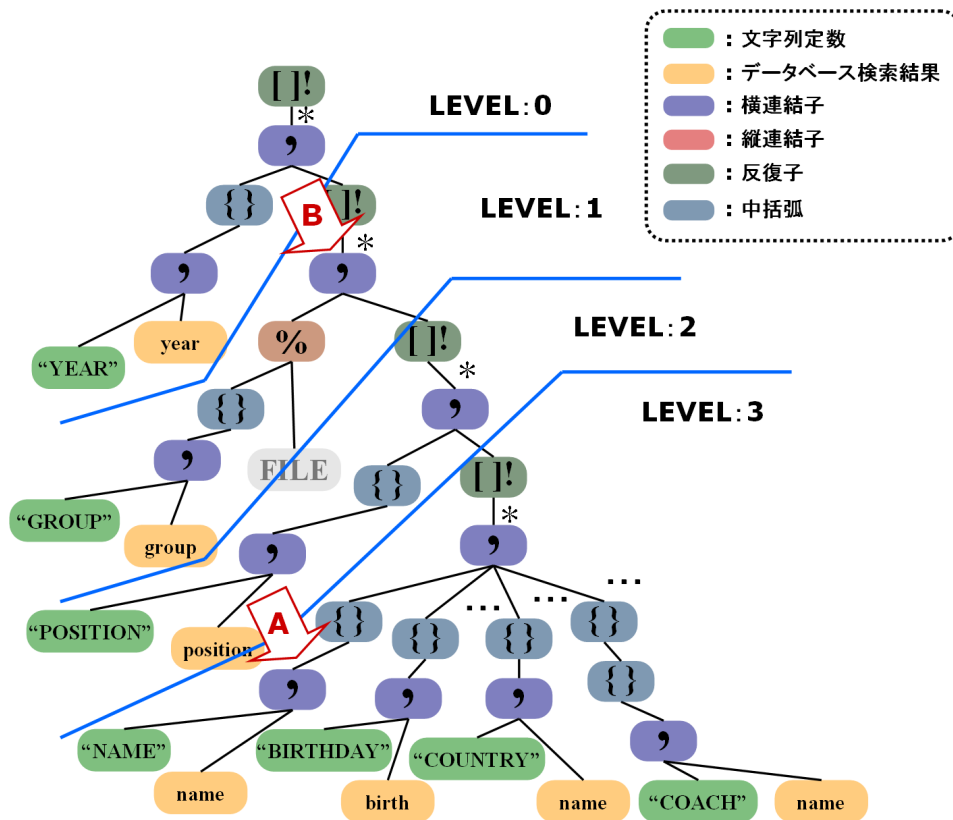
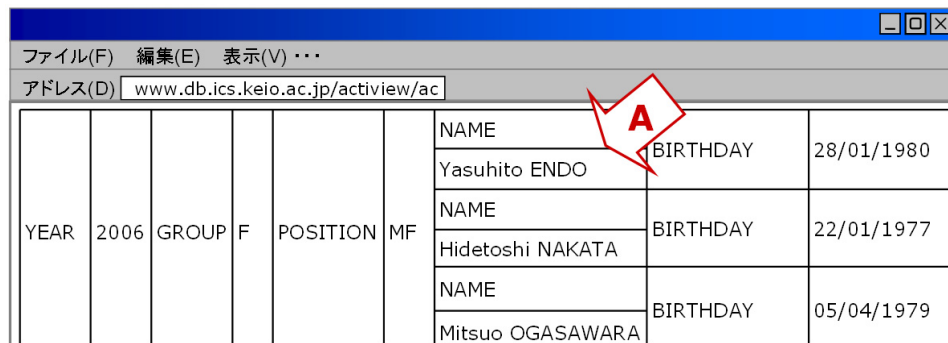


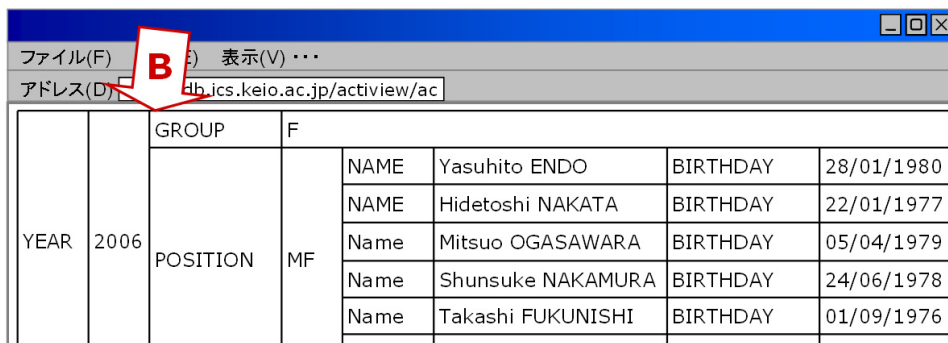
図 5.13: 反復子との連結子に変換を行う木構造例



The screenshot shows a web browser window with the address bar containing "www.db.ics.keio.ac.jp/actiview/ac". The main content area displays a table with columns for YEAR, 2006, GROUP, F, POSITION, MF, NAME, and BIRTHDAY. A red arrow labeled 'A' points to the 'NAME' cell of the first row.

YEAR	2006	GROUP	F	POSITION	MF	NAME	BIRTHDAY
						Yasuhito ENDO	28/01/1980
						Hidetoshi NAKATA	22/01/1977
						Mitsuo OGASAWARA	05/04/1979

図 5.14: ネスト内で変換が行われた場合の結果レイアウト



The screenshot shows a web browser window with the address bar containing "db.ics.keio.ac.jp/actiview/ac". The main content area displays a table with columns for YEAR, 2006, GROUP, F, NAME, and BIRTHDAY. A red arrow labeled 'B' points to the 'GROUP' cell of the first row.

YEAR	2006	GROUP	F	NAME	BIRTHDAY
				Yasuhito ENDO	28/01/1980
				Hidetoshi NAKATA	22/01/1977
				Mitsuo OGASAWARA	05/04/1979
				Shunsuke NAKAMURA	24/06/1978
				Takashi FUKUNISHI	01/09/1976

図 5.15: ネストとの連結子に変換が行われた場合の結果レイアウト

木構造でのこの 2 種類の変換によって生成される結果レイアウトを図 5.14 と図 5.15 に示す。

図 5.14 に示されるように文字列定数“ NAME ”と検索結果の間に変換を行った場合、一貫性のなく、充填率的にも悪いレイアウトを生成してしまう。

これに比べて、図 5.15 は全体的なレイアウトの変更も少なく充填率への影響も少ない、一貫性のあるレイアウトを生成している。

5.4.3 上位ノードの優先

候補レイアウトが複数存在する場合，より上位のノードの演算子に変換を行ったレイアウトの方を優先する．たとえば，図 5.16 の①と②で変換を行って再構成されたレイアウト結果の幅目標の評価関数の結果値が同じであれば，図 5.16 の②に変換を適用したレイアウトを結果レイアウトとする．

図 5.16 の木構造への変換によるレイアウト結果を図 5.17 と図 5.18 に示す．図 5.17 は図 5.16 の①にレイアウト変換を行った場合であり，図 5.18 は図 5.16 の②の方で変換を行った場合の結果レイアウトである．結果レイアウトが示すように，上位ノードの変換を優先した結果の方（図 5.18）がより一貫性のあるレイアウトを生成している．同一ネスト内でのノード関係は，5.4.2 項と同じく，上位ノードの方が弱い結合であると判断し，上位ノードの変換を優先する．

ネスト間での変換も，同様に上位のネストで行う変換が下位ネスト（深いネスト）で行う変換より優先される．

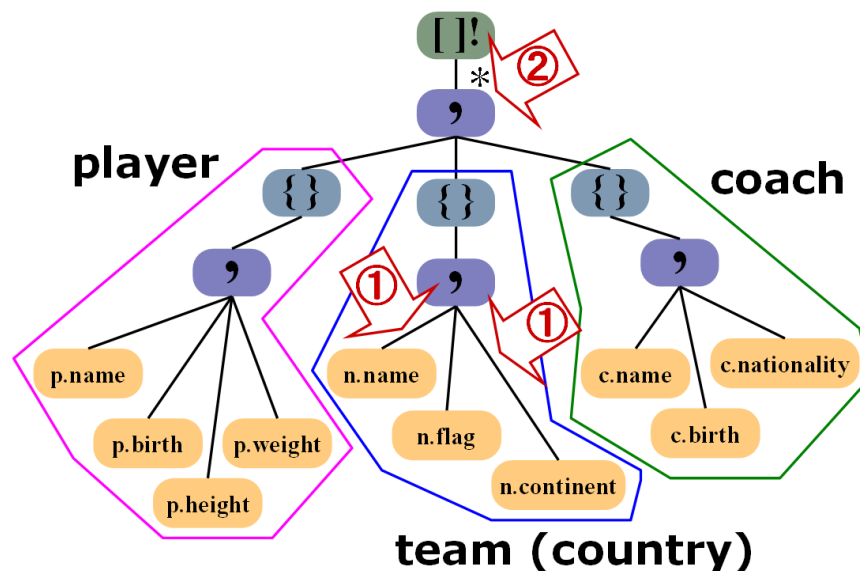


図 5.16: ネスト内の上位ノードを優先する木構造例

www.db.ics.keio.ac.jp/actiview/ac						
Yasuhito ENDO	28/01/1980	177	75	Japan	Zico	3 March
				●		
				Asia		
RONALDO	22/09/1976	183	90	Brazil	PARREIRA Carlos Alberto	27 Februar
				🇧🇷		
				South America		

図 5.17: 優先順位がない場合の結果レイアウト

www.db.ics.keio.ac.jp/actiview/ac			
Yasuhito ENDO	28/01/1980	177	75
Japan	●	Asia	
Zico	3 March 1953	Brazilian	
RONALDO	22/09/1976	183	90
Brazil	🇧🇷	South America	
PARREIRA Carlos	27 February 1943	Brazilian	

図 5.18: ネスト内の上位ノードを優先した場合の結果レイアウト

5.4.4 少ない変換適用の優先

ACTIVIEW では、全体レイアウトに与える影響が少ない方が元の開発者が指定したレイアウトに近いと仮定している。開発者によって定義された初期レイアウト構造が一番ユーザに直感的で見やすいレイアウトであるという前提に基づいている。したがって、変換の回数がより少ないレイアウトを元の構成により近いレイアウトであると判断している。たとえば、ユーザの画面表示幅と初期レイアウトの幅の差が 100pixel であるとする。100pixel を縮めるため、図 5.19 の①のように 3 カ所に変換を行う場合と、図 5.19 の②の連結子を変換する 2 つの手法がある。3 カ所に変換を行う場合は、まず横連結から縦連結子に変換し、Max⁵ 130pixel から Min 70pixel を差し引いた 60pixel を縮め、つづいて各ネストで縮められる 20pixel と 40pixel の合計の 120pixel を縮める。図 5.19 の②の 1 カ所に変換を行う場合は、横連結であった場合の幅値 (Max) 100pixel を縮める。

両方とも条件を満たしているが、変換の回数から見て、少ない変換で条件を満たした方が優先される。

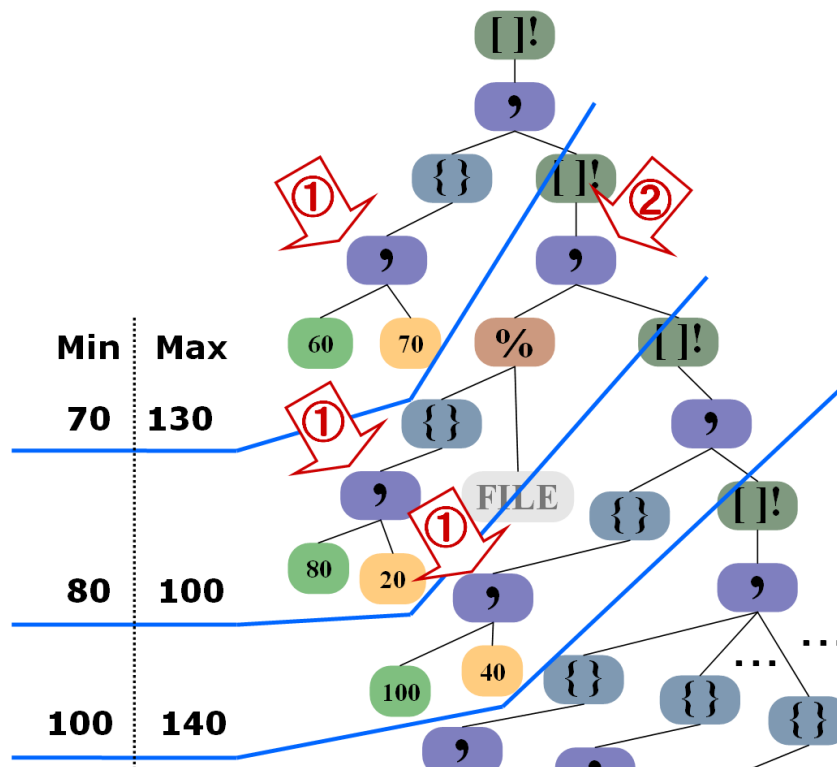


図 5.19: 少ない変換適用を優先する例

⁵min と max は各ネストで生成されるネスト別のレイアウトの最小幅と最大幅の値である。

5.4.5 中括弧内の変換の統一変換の優先

中括弧内に指定されている属性は、他の属性との関係より意味的に強い関連性を持っていると仮定している。そこで、構造の変換を行う場合にあっては同時に変換を行うことでより統一的なレイアウトの生成を目指す。同時変換によって処理コストの削減も可能になる。たとえば、同一ネスト内、あるいは同一中括弧内（図 5.20 の①）に文字列定数と検索結果を表す属性の連結が複数現れた場合（図 5.20 の②）には、同時変換を行う。同時変換が行われたレイアウトと行われなかったものを比較してみると、同時に変換が行われた場合の結果レイアウトの方（図 5.21）が、同時に変換が行われなかった場合（図 5.22）に比べて統一性を維持していることが分かる。

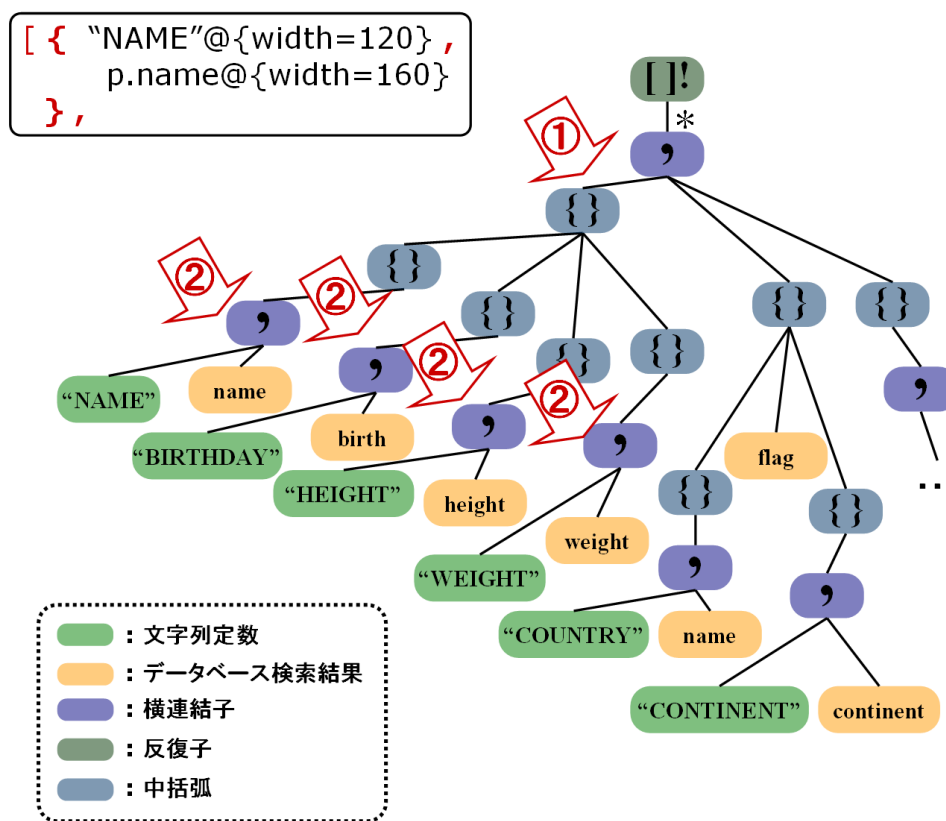


図 5.20: 同一中括弧内の文字列定数を同時に変換する例

ブラウザのアドレスバー: www.db.ics.keio.ac.jp/actiview/ac

NAME	BIRTHDAY	HEIGHT	WEIGHT	COUN
Yasuhito ENDO	28/01/1980	177	75	
NAME	BIRTHDAY	HEIGHT	WEIGHT	COUN
Hidetoshi NAKATA	22/01/1977	175	72	
NAME	BIRTHDAY	HEIGHT	WEIGHT	COUN
Mitsuo OGASAWARA	05/04/1979	173	72	

図 5.21: 同時に変換が行われた場合の結果レイアウト

ブラウザのアドレスバー: www.db.ics.keio.ac.jp/actiview/ac

NAME	BIRTHDAY	28/01/1980	HEIGHT	WEIGHT	75
Yasuhito ENDO			177		
NAME	BIRTHDAY	22/01/1977	HEIGHT	WEIGHT	72
Hidetoshi NAKATA			175		
NAME	BIRTHDAY	05/04/1979	HEIGHT	WEIGHT	72
Mitsuo OGASAWARA			173		

図 5.22: 同時に変換が行われなかった場合の結果レイアウト

5.4.6 リンク変換の優先順位の変換

レイアウト変換によって幅制約を満たしたそれぞれの候補レイアウトが長さ目標を満たしていない場合は、再びリンク変換を適用する。この場合のリンク変換は、5.2 節で述べたリンク変換の優先順位と異なる戦略を定める。幅制約を満たした候補レイアウトに対しては次の優先順位に従いリンク変換を行う。

- (1) 長い結果テーブルをもつネストからの変換を優先する。
- (2) 深いレベルにあるネストからの変換を優先する。
- (3) 多くの子要素をもつネストからの変換を優先する。

5.4.7 メインアルゴリズム

本章で述べたレイアウト変換戦略の実際の適用について、アルゴリズム 5.3 を用いて詳細を示す。

ACTIVIEW は、まずユーザのブラウザサイズ（ユーザ表示画面のサイズ）を受け取る（1 行目）。5.2.6 項で述べたように、レイアウトの変換戦略を適用する際、効率的な処理を行うために変換されたレイアウトが制約と目標を満たしているかいないかを判断するとき必要な情報だけを、元の ACTIVIEW の問合せ文から抽出し、リストを生成する（2 行目）。元の ACTIVIEW の問合せ文（p.22 の図 2.12）から取り出した *structure_info_list* は次のような構造情報リストである。

$$structure_info_list = [\{ 60 , 70 \}, [\{ 80 , 20 \}, [\dots]!]!]!$$

このリストはレイアウトの変換による長さや幅の変換を計算するために、元の問題文から構造情報と属性の幅情報だけを取り出して生成した構造情報リストである。

図 2.12 の問合せ文から生成される *connect_with_nest_list* は次のようになる。

$$connect_with_nest_list = [[9 , " , " , level1 , 2 , subnest_info_list], [16 , " , " , level2 , 2 , , subnest_info_list], \dots]$$

connect_with_brace_list は、中括弧との連結子の *structure_info_list* リスト内の位置情報と連結子、中括弧が含まれているネストの情報（*subbrace_info_list*）、同じネスト内で中括弧同士の包含関係があればこれらの中括弧間の深さ情報⁶を *connect_with_nest_list* と同様のリストでもっている。

⁶上位ノードを優先する戦略に使える。

アルゴリズム 5.3 メインアルゴリズム

```
1: user_info ← browser size of user
2: parsing(original ACTIVIEW query) することによって
   structure_info_list,
   connect_with_nest_list,
   connect_with_brace_list,
   connect_in_brace_list_each_nest,
   calculator_list を生成
3: IC_list ← query(original ACTIVIEW query)
4: layout_width = width_calculate(structure_info_list)
5: layout_length = predict_length(structure_info_list, IC_list)
6: length_goal ← cal_length(layout_length, static_num, IC_list)
7: width_goal ← user_info
8: if layout_length > length_goal then
9:   link(connect_with_nest_list, IC_list) 処理によって
     substructure_list = < s1, s2, ..., sn > を生成
10:  for substructure_list の順に, si をチェック do
11:    candidacy_info_list = layout_change(si)
12:  end for
13: else
14:  if layout_width > width_goal then
15:    substructure_list = layout_change(structure_info_list)
16:    for substructure_list の順に, si をチェック do
17:      if layout_length(si) > length_goal then
18:        candidacy_info_list = link(si)
19:      else
20:        candidacy_info_list = si
21:      end if
22:    end for
23:  else
24:    candidacy_info_list = structure_info_list
25:  end if
26: end if
27: return result_layout = select(candidacy_info_list)
```

```
connect_with_brace_list = [ 10 , “ , ” , nest_level1 , subbrace_info_list , 0 ] , ...
                        [ 91 , “ , ” , nest_level4 , subbrace_info_list , 1 ] , ...
```

connect_in_brace_list_each_nest は同一中括弧内の連結子の情報，包含関係であるネストや中括弧の情報をもっているリストであり，同時変換を行う場合に使われる．

calculator_list はこれ以外の連結子，すなわち属性間の関連付けがない連結子の情報を格納する．戦略に従って変換が行った後でも，制約を満たさなかった場合は *calculator_list* 内の連結子を対象に全数探索を行うのでその場合に使われる．

そのほかに，各属性の IC_i の値のリスト (*IC_list*) と，レイアウトの長さ (*layout_length*) や広さ (*layout_width*) を計算した結果値と，長さ目標値 (*length_goal*) とユーザ表示画面の幅値 (*width_goal*) を用意する (3 行目から 7 行目) ．

まず，初期レイアウトの長さが，長さ目標を満たしていない場合 (8 行目から 12 行目) と元々満たしているレイアウトである場合 (13 行目から 27 行目) に分けて処理が行う．長さ目標を満たしていない場合は 5.2.1 項で述べたように，リンク変換 (5.2.8 項) を先に行うことで変換対象になる連結子の個数を減らす戦略に従う (9 行目) ．リンク変換の適用によって生成された複数のレイアウト (s_i) は，それぞれ制約と目標を満たすようにレイアウト変換を行う (11 行目) ．アルゴリズム 5.3 の 11 行目のレイアウト変換処理 (*layout_change()*) から生成された結果レイアウトが，候補レイアウト (*candicacy_info_list*) になる ．

初期レイアウトが長さ目標を満たしている場合は，幅目標を満たすための処理が行われる (14 行目から 23 行目) ．初期レイアウトが長さ目標も幅目標も満たしている場合，そのまま候補レイアウトになる⁷ (24 行目) ．

初期レイアウトが長さ目標を満たしているレイアウトでも，幅目標に対する変換を適用した段階で，長さ目標を満たさないレイアウトを生成してしまう可能性が高い ．そのため，長さ目標に対して再び変換を行わなければならない場合が生じる (17 行目から 19 行目) ．結果レイアウトが長さ目標を満たしているとそのまま候補レイアウトになる (20 行目) ．

複数の候補レイアウトが生成される場合は，候補レイアウト間の比較を行って最終結果レイアウトを選択する (27 行目) ．

5.4.8 レイアウト変換戦略アルゴリズム

レイアウト変換はアルゴリズム 5.4 のように，同一ネスト内にある中括弧でくくられている属性間の連結子 (2 行目，*connect_in_brace_list_each_nest*) を同時に変換する場

⁷この場合は実際，初期レイアウトがそのまま最終結果レイアウトになる場合，すなわちレイアウト変換が行われない場合である ．

合 (2 行目から 10 行目) と変換しない場合 (11 行目から 15 行目) の 2 つのパターンに対して, それぞれの戦略を適用する. 5.3 節で述べたように統一変換によって構造的に一貫性のあるレイアウトを生成できるとした仮定を基にしている.

アルゴリズム 5.4 レイアウト変換戦略アルゴリズム

```
1: layout_change(substructure_list, connect_in_brace_list_each_nest) 処理によって
   substructure_list =  $\langle l_1, l_2, \dots, t_n \rangle$  を生成
2: for substructure_list の順に,  $l_i$  をチェック do
3:   if layout_width( $l_i$ )  $\langle$  width_goal then
4:     candidacy_info_list =  $l_i$ 
5:   else
6:     temp_list  $\leftarrow$  layout_change( $l_i$ , connect_with_nest_list)
7:     temp_list  $\leftarrow$  layout_change( $l_i$ , connect_with_brace_list)
8:     candidacy_info_list = layout_change(temp_list, calculator_list)
9:   end if
10: end for
11: for substructure_list の順に,  $l_i$  のレイアウト変換を行う do
12:   temp_list  $\leftarrow$  layout_change( $l_i$ , connect_with_nest_list)
13:   temp_list  $\leftarrow$  layout_change( $l_i$ , connect_with_brace_list)
14:   candidacy_info_list = layout_change(temp_list, calculator_list)
15: end for
16: return candidacy_info_list
```

同一ネスト内の中括弧でくくられている属性間の連結子を一括変換した結果, 幅目標を満たした場合は, この結果レイアウトがそのまま候補レイアウトになる. 各候補レイアウトは, 自分のレイアウト変換に用いられた戦略の優先順位の情報を含むレイアウト構造情報リストをもつ. 5.4 節で述べた戦略の通りにネスト (*connect_with_nest_list*) や中括弧との連結子 (*connect_with_brace_list*) の変換を親ノードから優先に行う (6 行目から 7 行目, 12 行目から 13 行目). 提案している戦略の対象外の連結子への全数探索は 8 行目と 14 行目で行う (*calculator_list* の属性が対象になる.). すなわち, *calculator_list* は属性間の関連性を特定できない連結子の情報リストである.

5.4.9 結果レイアウトの選択

複数生成される候補レイアウトから，最もユーザ表示画面に適応したレイアウトを提供するためには最終結果レイアウトを判断する指標が必要となる．

5.4.7, 5.2.8, 5.4.8 項で述べたアルゴリズムに従って生成された複数の候補レイアウト，アルゴリズム 5.3 の *candidacy_info_list* の 27 行目の *select(candidacy_info_list)* によって最終結果レイアウトが選択される．結果レイアウトの選択アルゴリズムをアルゴリズム 5.5 に示す．

アルゴリズム 5.5 結果レイアウトの選択

```
1: candidacy_info_list =  $\langle l_1, l_2, \dots, l_n \rangle$  を生成
2: while  $l_i \neq \emptyset$  do
3:    $point_{tw} \leftarrow l_i$  の TW (幅占有率) を計算する
4:    $point_{tf} \leftarrow l_i$  の TF (充填率) を計算する
5:    $point_{num} \leftarrow l_i$  で行った連結子の変換回数
6:    $point_{order} \leftarrow l_i$  で適用された変換の優先順位
7: end while
8: while 各 point  $\neq \emptyset$  do
9:   candidacy_info_list  $\leftarrow$  各 point を総合的にランキングする
10: end while
11: return candidacy_info_list の中でランク 1 位の候補レイアウト
```

ACTIVIEW では，各候補レイアウトに対する目標の結果値を計算した値（3 行目，3.3 節の幅目標と 4 行目，3.4 節の充填率目標），連結子の変換を行った回数（5 行目），レイアウト変換に適用された変換優先順位（6 行目，5.4 節）を基に総合的にランキングを行う（9 行目）．この結果順位が一番高いレイアウト，すなわち長所が最も多い候補レイアウトが最終結果レイアウトとして選択する．

第 6 章では，今まで述べた提案戦略とこの戦略に従ったアルゴリズムによって実際に生成される結果レイアウトと，従来の手法によって生成される結果レイアウトを，様々な実験によって比較し，本論文でのレイアウト変換戦略が有効であることを示す．

第 6 章

実験および評価

本章では提案手法により得られる結果レイアウトの妥当性と、処理時間の測定のために行った実験およびその結果について示す。

6.1 実験方法

ACTIVIEW は、従来の研究と比べ、データの内容ではなく構造そのものを対象にしているため、まずレイアウト変換手法によって生成されるレイアウトとの差を直感的に理解しやすく定義した図 2.12 の ACTIVIEW の問合せ文 (p.22) を示す。その後、予備実験で行った様々なレイアウト型に対して従来の手法との結果の比較を行う。

ACTIVIEW の問合せ文を書き直すことによって実際生成される結果レイアウトを示しながら説明を進める。まず、図 2.12 の ACTIVIEW の問合せ文によって生成される実際のレイアウトの例を示すために、World Cup のデータ [30] を用意した。FIFA ホームページからの実際のデータをベースに図 6.1 のようなデータテーブルを作成した。

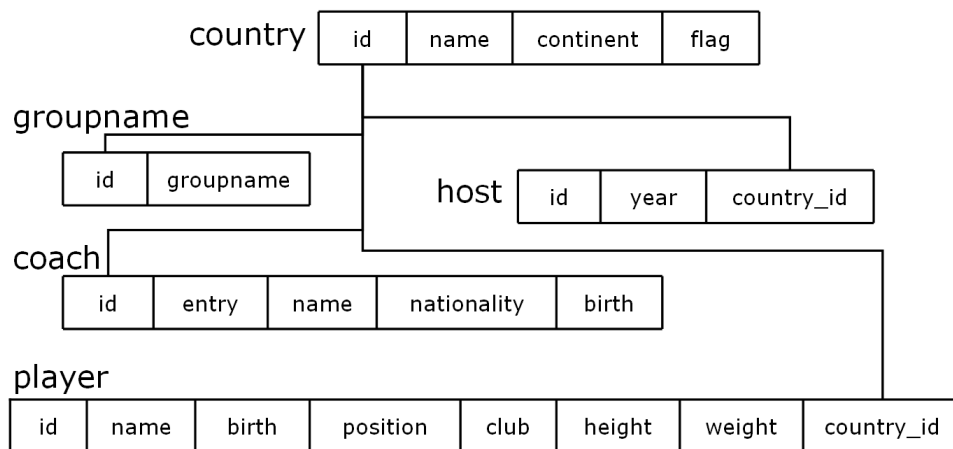


図 6.1: データベースのテーブルスキーマ

本実験では 2006 年のデータを利用している。World Cup は 4 年ごとに開かれ、2006 年には 32 チームがエントリーされた。2006 年は各チーム 23 人の選手がエントリーされたので、736 レコードの選手データ (player) と 32 チーム各国のデータ (country) 32 レコード、各チームのコーチのデータ (coach) が 32 レコードある。各チームは A から H までの 8 のトーナメント組に分かれ、組ごとにリーグ戦をしているので各組の情報 (groupname) の 8 レコードを格納して実験を行った。実際の Web ビューまで生成は関係データベースからの検索結果の量への依存が高いため、処理時間の比較は ACTIVIEW を利用して元の問合せ文を書き換えて、ユーザ表示画面のサイズに合わせた問合せ文

に変換した時点まで，すなわち書き換えた問合せ文を SuperSQL システムに渡す前の時間までの処理時間だけを比較する．

図 2.12 の問合せ文から，変換を適用する前に生成されるオリジナルレイアウト（p.24 の図 2.13）の幅は 3,030pixel である．図 2.12 の問合せ文から生成される結果レイアウト（図 2.13）は，“2006 年”で大きくネストされた各グループにあるポジション別の選手とコーチの情報を表すテーブル（Web ビュー）である．すなわち，選手の情報が，まず対戦グループ別，そしてポジション別の国別にネストされた深さ 5 のテーブル（図 6.2 参照）で表される．

6.2 実験結果・評価

本節では提案手法が一貫性のある構造のレイアウトを生成することを示し，従来の手法によって生成されるレイアウト結果と比較を行う．

第 5 章に提案した戦略に従って，元の問合せ文の再構成によって生成されるレイアウト結果の実現例を観察しながら論じる．

図 6.2 は図 2.12 の問合せ文の一部を省略したものでこの問合せ文のネスト指定に従う初期レイアウトのネスト構造を示す．この初期レイアウトから本論文で提案する変換戦略に従って，レイアウト変換を行った実際の結果レイアウトを用いて，各戦略が妥当であることを示す．

本提案手法によって生成される最終結果レイアウトを生成するまでの中間結果を示しながら説明を続ける．長さ 700pixel のユーザ表示画面に対して，5.2.1 節で述べたリンク変換戦略に従ってリンク変換を行う（アルゴリズム 5.2）．

図 6.3 はリンク変換の戦略のうち，

- 条件
 - (1) 反復子との連結子に適用することを前提にする．
- 優先順位
 - (1) 多くの子要素をもつネストからの変換を優先する．
 - (2) 長い結果テーブルをもつネストからの変換を優先する．

に従って変換を行った結果レイアウトである．

ACTIVIEWの問合せ文

```
[ { "YEAR", h.year } ,
  [ { "GROUP", t.group } %
    [ { "COUNTRY", n.name } ,
      [ { "POSITION", p.position } ,
        [ { "NAME", p.name } ,
          { "BIRTHDAY", p.birth } , ...
        ]! ]! ]! ]! ]!
```

YEAR	COUNTRY	POSITION	NAME	BIRTHDAY
2006	Portugal (POR)	FW	LUIS FIGO	04/11/1972
			PAULETA	28/04/1973
			SIMAO SABROSA	31/10/1979
			PAULO SANTOS	11/12/1972
		GK	QUIM	13/11/1975
			RICARDO	11/02/1976
			COSTINHA	01/12/1974
			DECO	27/08/1977
		MF	HUGO VIANA	15/01/1983
			MANICHE	11/11/1977
			PETIT	25/09/1976
			TIAGO	02/05/1981
		DF	David ROZEHNAL	05/07/1980
			Marek JANKULOVSKI	09/05/1977
			Martin JIRANEK	25/05/1979
			Pavel MARES	18/01/1976
			Tomas UJFALUSI	24/03/1978
			Zdenek GRYGERA	14/05/1980
			Jan KOLLER	30/03/1973

図 6.2: 問合せ文でのネスト指定と結果レイアウトの例

The figure consists of two screenshots of a web browser window. The top screenshot shows a table with the following data:

GROUP	YEAR	COUNTRY	POSITION	GK
D	2006	Mexico (MEX)	POSITION	GK
			POSITION	MF
			POSITION	DF
			POSITION	FW
			POSITION	GK
		Portugal (POR)	POSITION	MF
			POSITION	DF
			POSITION	FW
			POSITION	GK
			POSITION	MF
E	2006	Czech Republic (CZE)	POSITION	DF
			POSITION	GK
			POSITION	MF
			POSITION	DF
			POSITION	GK
E	2006	Ghana (GHA)	POSITION	DF
			POSITION	FW
			POSITION	GK
			POSITION	MF
			POSITION	DF

The bottom screenshot shows a table with the following data:

NAME	BIRTHDAY	HEIGHT	WEIGHT	COUNTRY	CONTINENT
Hidetoshi NAKATA	22/01/1977	175	72	Japan (JPN)	Asia
Junichi INAMOTO	18/09/1979	180	76	Japan (JPN)	Asia
Mitsuo OGASAWARA	05/04/1979	173	72	Japan (JPN)	Asia
Shinji ONO	27/09/1979	175	74	Japan (JPN)	Asia
Shunsuke NAKAMURA	24/06/1978	178	69	Japan (JPN)	Asia
Takashi FUKUNISHI	01/09/1976	181	74	Japan (JPN)	Asia
Yasuhito ENDO	28/01/1980	177	75	Japan (JPN)	Asia

A red arrow labeled "Hyperlink" points from the "POSITION" cell in the top table to the "CONTINENT" cell in the bottom table.

図 6.3: リンク戦略に従って生成されたレイアウトの例

ネストごとに分けられた(アルゴリズム 5.2 の 2 行目¹)要素の中で, 図 6.2 の LEVEL:4 の部分が最も多くの子要素を持つため, このネストの連結子が選ばれて変換される. 図 2.12 の問合せ文では多くの子要素をもっているネストが最も長いコンテンツネストである(アルゴリズム 5.2 の 3 行目から 5 行目).

リンク変換を優先に行った場合, レイアウトの幅のみを基準にした図 6.4 のように分かれたレイアウトを生成するほうが望ましい結果である. 図 6.4 は比較実験であり, 本論文の提案戦略に従ったものではない. しかし, 図 6.4 のように分けるとリンク元である国旗とリンク先の情報の関連性が崩れたレイアウトが生成されてしまう.

すなわち, 図 6.4 のようなレイアウトだとリンク先のレイアウトの情報のうち, 大陸とコーチの情報がどんな関係であるか判別できない.

他に, 図 6.4 のように分かれたレイアウトを生成するためには“ n.flag@{width=30} ”の次の横連結子“ , ”にリンク変換が行われなければならない(図 6.5). しかし, 図 6.5 の横連結子(,)がリンク連結子(%)に変換されると図 6.6 のように, 次の“ { “ CONTINENT ”@{width=90} , n.con@{width=300} } ”部分のみがリンク先のページに分かれる.

図 6.4 のように分かれたレイアウトを生成するためには自動的に中括弧を入れて同じ

¹リンク戦略 1 に従って変換対象になる連結子がネストとの連結子(*connect_with_nest_list*)だけを対象にしている.

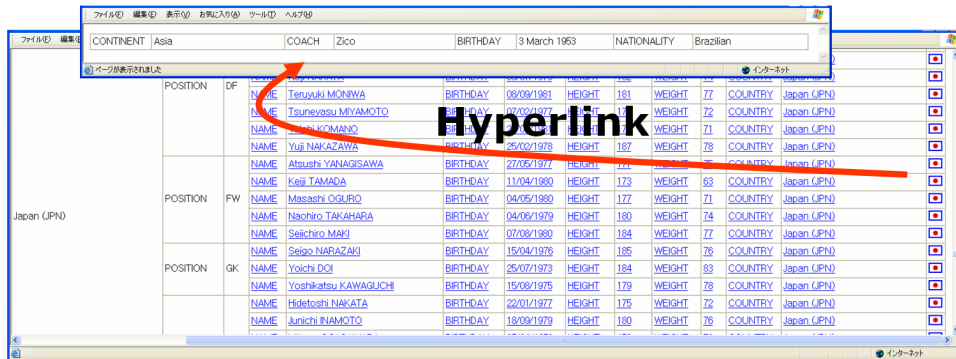


図 6.4: 幅条件だけを基準にして分割したレイアウトの例

ACTIVIEWの問合せ文

```

...
{ "COUNTRY"@{width=80} , n.name@{width=300} } ,
  n.flag@{width=30} } ,
{ "CONTINENT"@{width=90} , n.con@{width=300} } ,
{ { "COACH" @ {width=110} , c.name@{width=260} } } ,
...

```

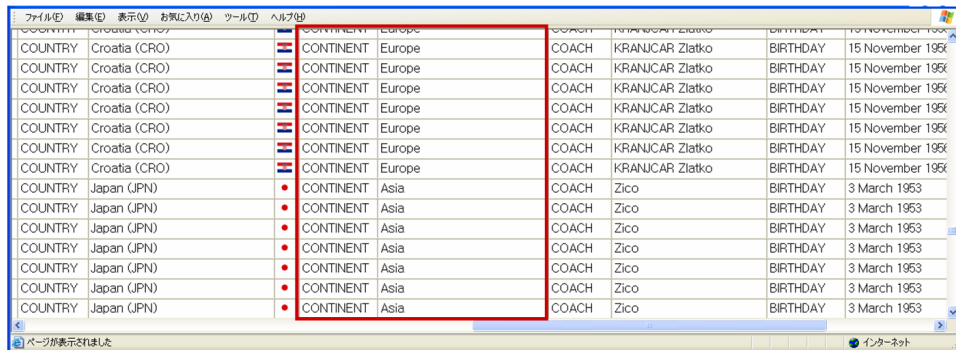


図 6.5: リンク変換を行う連結子

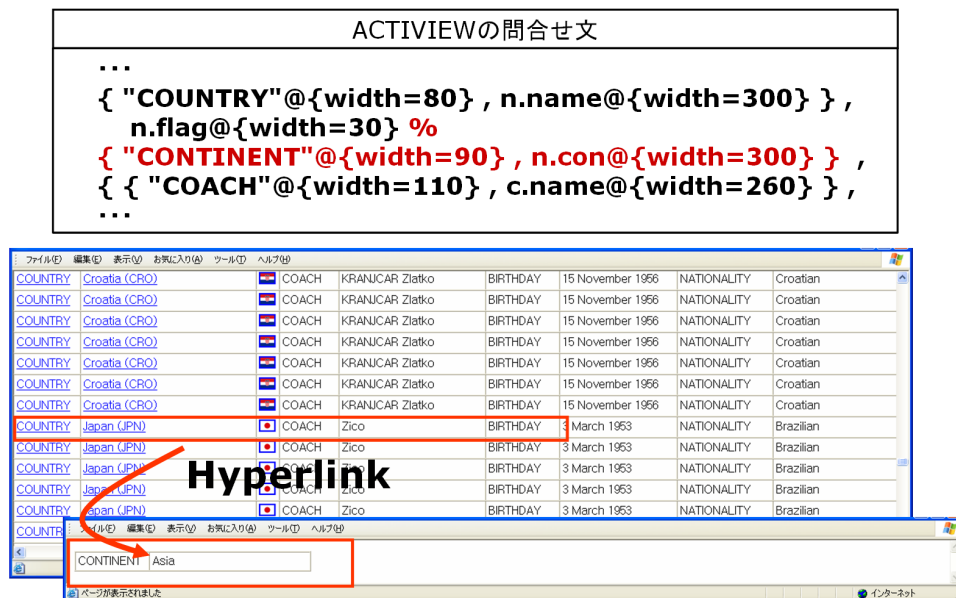


図 6.6: リンク先のレイアウトになる範囲の例

グループにしなければならない (図 6.7) . しかしながら , 自動的に中括弧をつけてグループを生成することには注意しなければならない (3.2.4 項) .

レイアウト結果の妥当性は , 開発者の意味反映が維持されているか , そうではなくて崩れてしまったかによって決定される . これを自動的に判断することは難しい . そこで , リンク変換では ,

- (1) 反復子との連結子に適用することを前提にする .

変換戦略 (5.2.1 項) の条件を用いている .

リンク戦略に従って変換を行うと図 6.3 のようにリンク元の幅は十分狭くなるので , リンク先の幅広いレイアウトが幅制約を満たすまでリンク元のレイアウトの変換は行なわれない (アルゴリズム 5.3 の 24 行目) . 幅が急に狭くなり , 幅目標の評価値の低いレイアウトが生成されてしまうが , 意味的な関連性に基づいた一貫性のあるレイアウトが生成されるので優先にすることは妥当であると考える .

ユーザの表示画面の幅が狭くなったときにはリンク先のレイアウトに対して , 5.4.7 , 5.4.8 項のアルゴリズムに従ったレイアウト変換を行う (5.4 節の変換戦略) .

5.4 節の一貫性のあるレイアウトを生成するための変換戦略に従ったレイアウト変換を変換順に述べる .

まず , 幅制約を満たすために行う連結子の変換を複数の領域 (複数の中括弧内) で行った場合 (図 6.8) と , 同一領域内 (同一ネスト内の中括弧同士) の連結に適用する場



図 6.7: 中括弧の適用による範囲指定

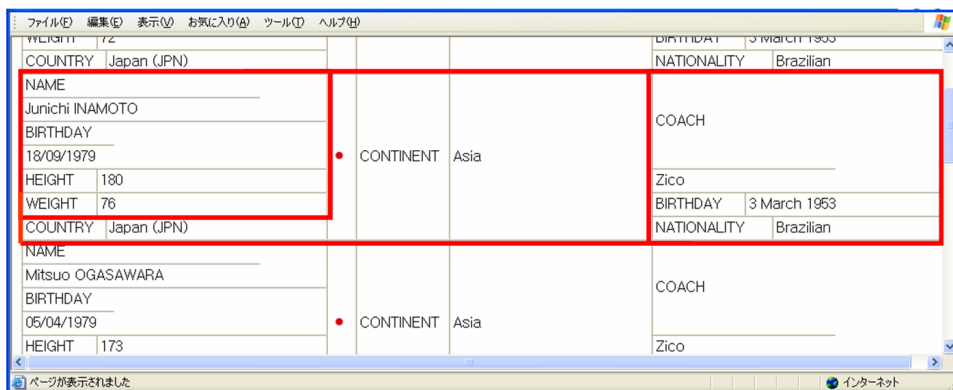


図 6.8: 複数の中括弧内の連結子変換による結果レイアウトの例

合)で行った場合(図 6.9)の結果レイアウトを比較してみる。

図 6.8 と図 6.9 は同じく幅制約を満たしたレイアウトであるが、生成された結果の構造は大きく異なっている。

図 6.8 は複数の中括弧内領域で連結子の変換を行ったために、構造的に混ざっているレイアウトが生成されてしまう。この実験結果のようにレイアウトの幅だけを基準にすると、全数探索により幅に最適化されたレイアウトを生成してもユーザには望ましくないレイアウトになってしまう恐れが高い。

図 6.8 に比べて、図 6.9 は同一ネスト内の中括弧同士の変換に適用した場合である。図 6.8 の結果よりは、構造的に統一されたレイアウト結果であることが分かる。

同一連結子が同時に変換されるとレイアウトの構成は統一されたものになり、直感

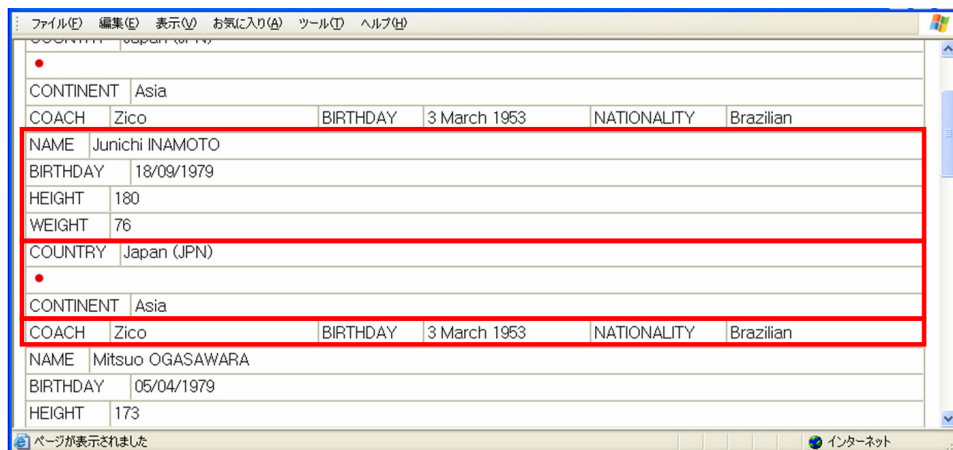


図 6.9: 同一ネスト内の連結子変換による結果レイアウトの例

的に優れたレイアウトになると判断できる。

これは、5.4 節に述べた一貫性のあるレイアウト変換戦略、

- (1) 複数のネスト内で行う変換より同一ネスト内での変換を優先する。

の妥当性を示す一例である。詳しい変換戦略は 5.4.1 項で述べた。しかしこの例において、図 6.8 と図 6.9 の両方とも低い充填率のレイアウトを生成していることが分かる。この問題を解決するために、一貫性を維持しながらより効率的にユーザ表示画面の領域を利用できる戦略が必要である。

5.3 節に述べたように統一変換可能な連結子は同時に変換する手法によって、処理コストの削減と一貫的なレイアウトの生成が可能になる。

5.4.8 項の変換アルゴリズムでも文字列定数などの中括弧内の連結子を同時に変換する場合（アルゴリズム 5.4 の 2 行目から 10 行目）と、同一ネスト内の連結子を変換する場合（アルゴリズム 5.4 の 11 行目から 15 行目）の両方で候補レイアウトを生成している。

図 6.10 は 5.3 節の戦略に従い同一ネスト内の文字列定数との連結子を同時に変換した結果レイアウトである。これは、5.4 節に述べた一貫性のあるレイアウト変換戦略、

- (5) 中括弧内の連結子は統一変換を優先する。

に従った結果レイアウトの優先順位をもつ。詳しい変換戦略は 5.4.5 項で述べた。

図 6.11 は意味的な区別が自動的に判断可能な範囲の属性を中括弧を利用してくり、グルーピングしたうえ、これらの連結子に変換を行った結果レイアウトである。

NAME	BIRTHDAY	HEIGHT	WEIGHT	COUNTRY	CONTINENT	COACH	BIRTHDAY	NATIONALITY
Hidetoshi NAKATA	22/01/1977	175	72	Japan (JPN)	Asia	Zico	3 March 1953	Brazilian
NAME	BIRTHDAY	HEIGHT	WEIGHT	COUNTRY	CONTINENT	COACH	BIRTHDAY	NATIONALITY
Jurichi INAMOTO	18/09/1979	180	76	Japan (JPN)	Asia	Zico	3 March 1953	Brazilian
NAME	BIRTHDAY	HEIGHT	WEIGHT	COUNTRY	CONTINENT	COACH	BIRTHDAY	NATIONALITY
Mitsuo OGASAWARA	05/04/1979	173	72	Japan (JPN)	Asia	Zico	3 March 1953	Brazilian
NAME	BIRTHDAY	HEIGHT	WEIGHT	COUNTRY	CONTINENT	COACH	BIRTHDAY	NATIONALITY
Shinji ONO	27/09/1979	175	74	Japan (JPN)	Asia	Zico	3 March 1953	Brazilian
NAME	BIRTHDAY	HEIGHT	WEIGHT	COUNTRY	CONTINENT	COACH	BIRTHDAY	NATIONALITY
Shunsuke NAKAMURA	24/06/1978	178	69	Japan (JPN)	Asia	Zico	3 March 1953	Brazilian
NAME	BIRTHDAY	HEIGHT	WEIGHT	COUNTRY	CONTINENT	COACH	BIRTHDAY	NATIONALITY
Takashi FUKUNISHI	01/09/1976	181	74	Japan (JPN)	Asia	Zico	3 March 1953	Brazilian
NAME	BIRTHDAY	HEIGHT	WEIGHT	COUNTRY	CONTINENT	COACH	BIRTHDAY	NATIONALITY

図 6.10: 文字列定数に同時に変換を行った結果レイアウトの例

COUNTRY	Japan (JPN)	CONTINENT	Asia				
COACH	Zico	BIRTHDAY	3 March 1953	NATIONALITY	Brazilian		
NAME	Mitsuo OGASAWARA	BIRTHDAY	05/04/1979	HEIGHT	173	WEIGHT	72
COUNTRY	Japan (JPN)	CONTINENT	Asia				
COACH	Zico	BIRTHDAY	3 March 1953	NATIONALITY	Brazilian		
NAME	Shinji ONO	BIRTHDAY	27/09/1979	HEIGHT	175	WEIGHT	74
COUNTRY	Japan (JPN)	CONTINENT	Asia				
COACH	Zico	BIRTHDAY	3 March 1953	NATIONALITY	Brazilian		
NAME	Shunsuke NAKAMURA	BIRTHDAY	24/06/1978	HEIGHT	178	WEIGHT	69
COUNTRY	Japan (JPN)	CONTINENT	Asia				
COACH	Zico	BIRTHDAY	3 March 1953	NATIONALITY	Brazilian		
NAME	Takashi FUKUNISHI	BIRTHDAY	01/09/1976	HEIGHT	181	WEIGHT	74
COUNTRY	Japan (JPN)	CONTINENT	Asia				
COACH	Zico	BIRTHDAY	3 March 1953	NATIONALITY	Brazilian		

図 6.11: システムが指定した中括弧間の連結子に変換を行った結果レイアウトの例

複数の属性を中括弧でくくると、この中括弧との連結子が上位の連結子になる（図 5.4 の B_i にあたる）。この連結子に対して変換を行うと、3 つに分かれたグループ間の 2 カ所に変換を行うことにより、図 6.10 のように 9 カ所に変換を行った結果レイアウトと同じ程度の幅をもつレイアウト結果が生成できる。図 6.11 の結果レイアウトは、充填率の観点からも図 6.10 の結果レイアウトと比べて良い結果レイアウトを生成している。

5.4 節に述べた一貫性のあるレイアウトの変換戦略、

- (3) 同一ネスト内では親ノードの方を優先する²。
- (4) 変換の回数が少ないレイアウトを優先する。

に従うと、図 6.11 の候補レイアウトが図 6.10 の候補レイアウトより高い優先順位をもっている。詳しい変換戦略は 5.4.9 項で述べた。

ユーザ表示画面がもっと狭くなる場合は、図 6.10 と図 6.11 の候補レイアウトからさらにレイアウト変換を行っている。レイアウト変換を続ける際、関連性に基づいたレイアウト変換、すなわち本論文で提案した変換戦略に従って変換を行う対象になる連結子がなくなった場合は全数探索を行って候補レイアウトを生成する。これらの候補レイアウトから、最終結果レイアウトは評価関数によって選ぶ（5.4.9 項）。

図 6.12 は 5.4 節の、

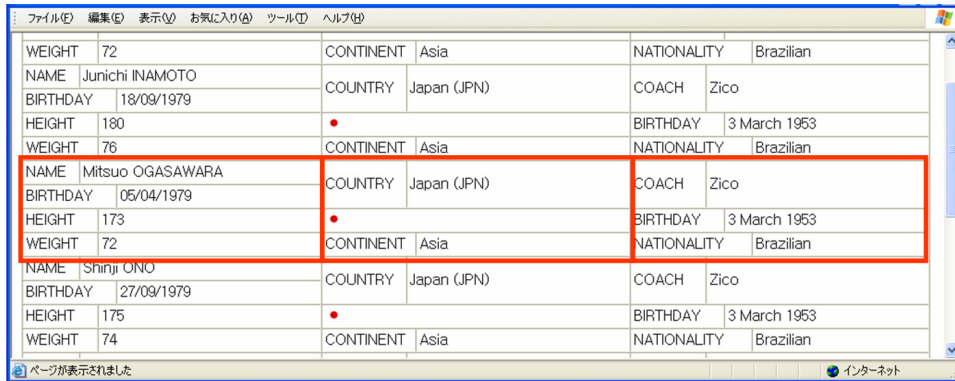
- (2) 各属性間の変換より入れ子や中括弧との連結子への変換を優先する。
- (3) 同一ネスト内では親ノードの方を優先する。

戦略に従った、アルゴリズム `alg:strategy-algorithm` によって生成された結果レイアウトである。図 6.12 から分かるように、文字列定数と属性を含む中括弧をさらに含む中括弧の間、すなわち上位ノード優先に変換が適用された。

続いて上位ノード優先に変換を行った結果レイアウトは、図 6.13 のような結果レイアウトを生成する。しかし、図 6.12 と図 6.13 の結果レイアウトの幅差が大きいのでユーザ表示画面の幅がこの 2 つの結果幅の間の値をもっている場合、全数探索を行った候補レイアウトが生成される。

予備実験で行ったレイアウト変換の結果でも図 6.14 の結果レイアウトのように、5.4.2 項に述べた戦略に従って統一的構造のレイアウトを生成した。図 6.14 は上位のネストとこれに含まれる下位ネストの間の連結子が横から縦に変換された結果レイアウトを示す。5.4.2 項で述べたように、開発者が望んだ意味的なネスト関係の守られたレイアウトの生成が可能になっていることが分かる。

²中括弧を導入することによって、この中括弧は既存の連結同士を集めた上位のノードになる。



WEIGHT	72	CONTINENT	Asia	NATIONALITY	Brazilian
NAME	Junichi INAMOTO	COUNTRY	Japan (JPN)	COACH	Zico
BIRTHDAY	18/09/1979			BIRTHDAY	3 March 1953
HEIGHT	180				
WEIGHT	76	CONTINENT	Asia	NATIONALITY	Brazilian
NAME	Mitsuo OGASAWARA	COUNTRY	Japan (JPN)	COACH	Zico
BIRTHDAY	05/04/1979			BIRTHDAY	3 March 1953
HEIGHT	173				
WEIGHT	72	CONTINENT	Asia	NATIONALITY	Brazilian
NAME	Shinji ONO	COUNTRY	Japan (JPN)	COACH	Zico
BIRTHDAY	27/09/1979			BIRTHDAY	3 March 1953
HEIGHT	175				
WEIGHT	74	CONTINENT	Asia	NATIONALITY	Brazilian

図 6.12: 上位中括弧間の連結子に変換を行った結果レイアウトの例



CONTINENT	Asia
COACH	Zico
BIRTHDAY	3 March 1953
NATIONALITY	Brazilian
NAME	Mitsuo OGASAWARA
BIRTHDAY	05/04/1979
HEIGHT	173
WEIGHT	72
COUNTRY	Japan (JPN)
CONTINENT	Asia
COACH	Zico
BIRTHDAY	3 March 1953
NATIONALITY	Brazilian
NAME	Shinji ONO
BIRTHDAY	27/09/1979
HEIGHT	175
WEIGHT	74
COUNTRY	Japan (JPN)
CONTINENT	Asia
COACH	Zico

図 6.13: 上位ノードに変換を行った結果レイアウトの例

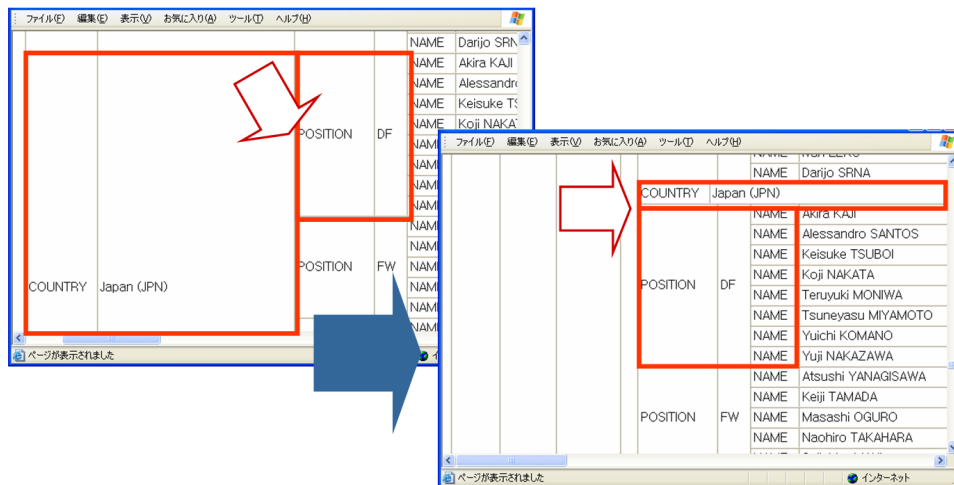


図 6.14: 反復子, グルーピングとの結合を優先したレイアウトの例

GROUP	A
A	NAME Jervis DRUMMOND Date of Birth 08/09/
	COUNTRY Costa Rica (CRC) CONTIN
	COACH GUIMARAES Alexandre Date of Birth
	NAME Jorge GUAGUA Date of Birth 28/09/
B	COUNTRY Ecuador (ECU) CONTIN
	NAME Alvaro MESEN HEIGHT 18 WEIGHT 80
	Date of Birth 24/12/1972
	NAME Artur BORUC HEIGHT 192 WEIGHT
	Date of Birth

図 6.15: 親ノードの方の変換を優先した結果レイアウトの例

同じく, 5.4.3 項の

- (3) 同一ネスト内では親ノードの方を優先する .

戦略に対して, 上位ノードに変換を適用する (図 6.15 の A) 方が, 下位ノード (図 6.15 の B) に変換を適用するより一貫性のあるレイアウトを生成している .

変換アルゴリズムに従って, ユーザ表示画面の幅が 400pixel である場合の幅制約を満たしたリンク元とリンク先の最終結果レイアウトを図 6.16 に示す .

さらに幅狭いユーザ表示画面に対しては, 図 6.3 のリンク元のレイアウトに対してもレイアウト変換が行われなければならない . 図 6.3 のリンク元にレイアウトが適用され

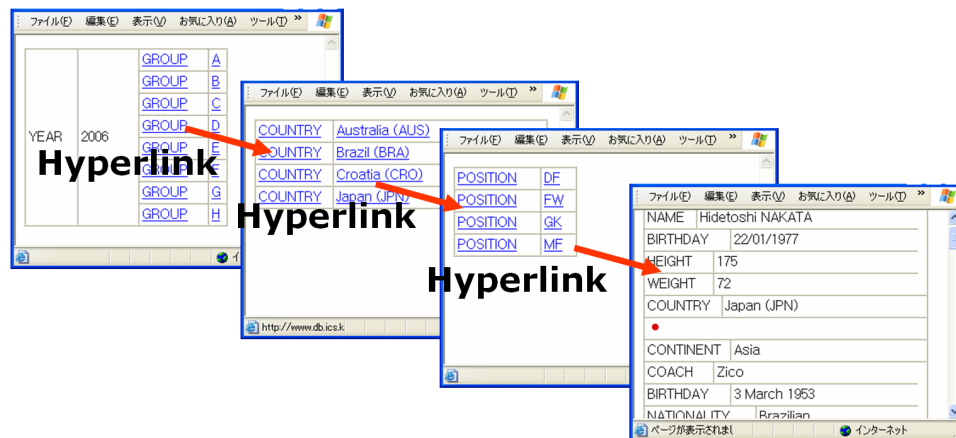


図 6.16: 幅制約を満たしたリンク先の結果レイアウト

る場合はネスト構造をもっているため、リンク変換戦略に従ってレイアウト変換を行うことになる。

- (4) 深いレベルにあるネストからの変換を優先する。

の戦略に従った（アルゴリズム 5.2 の 8 行目）処理によって、リンク変換を行い幅が縮まる）。

コンテンツネストのレイアウトの構成もすべて縦連結になり、可能な範囲で最も狭い幅のレイアウトを生成していることが分かる。

図 2.12 の問合せ文からレイアウト変換戦略により生成されるレイアウトの幅の種類と充填率を図 6.17 に示す。図 6.17 は 3,030pixel から 300pixel までの間で提案手法と単純順序探索によって生成可能なレイアウトの幅の全種類とその充填率を比較する。図 6.17 から見るように単純順序探索と比べ、提案手法の方が生成されるレイアウトに対してより高い充填率を示している。

たとえば、ユーザ表示画面の幅が 2,700pixel である場合は、単純順序探索と提案手法とともに、2,660pixel の幅をもつレイアウトを結果レイアウトとして生成する。しかし、提案手法の方が高い充填率をもつレイアウトを提供している（図 6.17 の A）。

図 6.17 に示すように単純順序探索によって生成されるレイアウトは、幅が縮められるほど充填率が極端に悪くなっていく。

特に、幅 2600pixel 以下では単純順序探索による結果レイアウトは非常に低い充填率を示しているが、これに比べて提案手法によって再構成されるレイアウトは全般的に高い充填率を示している。

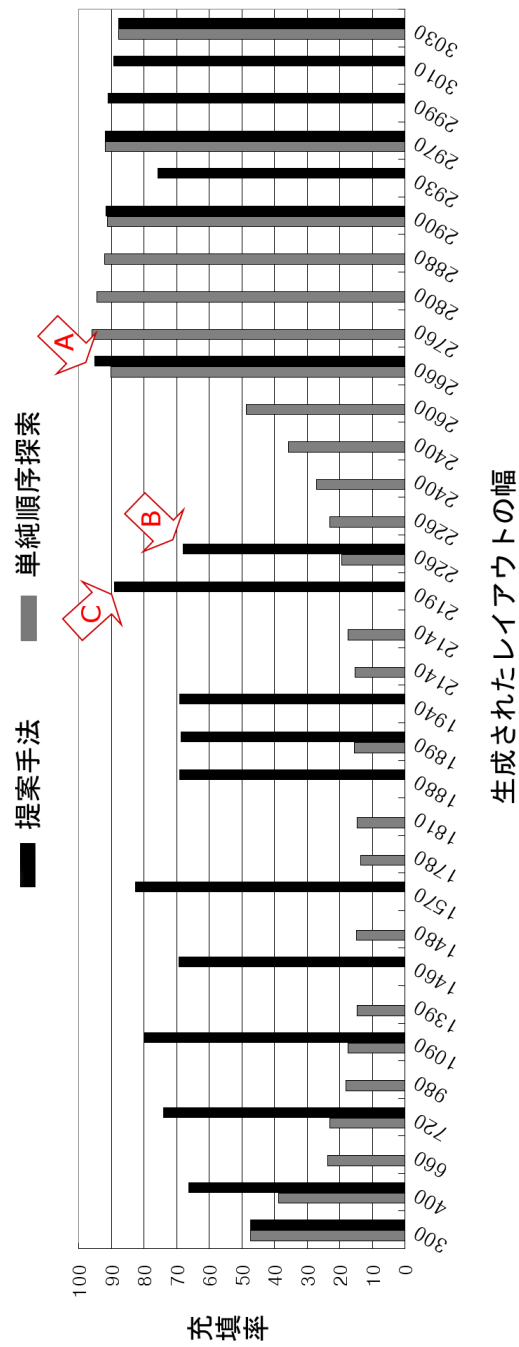


図 6.17: 変換されたレイアウトの充填率

YEAR				
2006				
GROUP				
A				
POSITION				
DF				
NAME				
Jervis DRUMMOND				
Date of Birth	WEIGHT	66	COUNTRY	Costa Rica (CRC)
08/09/1976				
HEIGHT	172			

図 6.18: 単純順序探索の結果レイアウト







NAME	Marcell JANSEN	Date of Birth	04/11/1985	HEIGH
COUNTRY	Germany (GER)		CONTINENT	Europe
NAME	Mariusz JOP	Date of Birth	03/08/1978	HEIGH
COUNTRY	Poland (POL)		CONTINENT	Europe
NAME	Alvaro MESEN	Date of Birth	24/12/1972	HEIGH
COUNTRY	Costa Rica (CRC)		CONTINENT	North a
NAME	Artur BORUC	Date of Birth	20/02/1980	HEIGH
COUNTRY	Poland (POL)		CONTINENT	Europe
NAME	Edwin VILLAFUERTE	Date of Birth	12/03/1979	HEIGH
COUNTRY	Ecuador (ECU)		CONTINENT	South ,
NAME	Jens LEHMANN	Date of Birth	10/11/1969	HEIGH
COUNTRY	Germany (GER)		CONTINENT	Europe
NAME	Gary NEVILLE	Date of Birth	18/02/1975	HEIGH

図 6.19: 提案手法による結果レイアウト

単純順序探索 (図 6.18) と提案手法 (図 6.19) による結果レイアウトの充填率の差が非常に大きな 2,260pixel の幅をもつレイアウトの生成結果を比較する。

単純順序探索によるレイアウト変換手法は統一性がない変換を行うため、変換を行えば行うほど 図 6.18 の結果レイアウトのように、それと連結される構造に大きい影響を与え、充填率が低くなる。これに比べて提案手法によって生成されるレイアウトは (図 6.19)、ほぼ無駄なスペースがない構造を生成している。

図 6.20 は各結果レイアウトを生成するために適用された変換の回数を示す。提案手法によるレイアウト変換では結合子の変換数が多い場合 (図 6.20 の B) であっても、統一変換を行うために計算コストは小さい。そのうえ、属性間の関連性に基づいた変換を優先しているので (5.4 節)、単純順序探索に比べて少ない数の結合子の変換 (図 6.20

の A) で制約を満たしたレイアウトの生成が可能になっている。

たとえば，ユーザの表示画面の幅が 2,300pixel である場合，提案手法では最適の充填率をもつ 2,190pixel (図 6.17 の C) の幅のレイアウトと，最適の幅占有率をもつ 2,260pixel の幅をもつレイアウト (図 6.17 の A) が候補レイアウトとしてあげられる．5.4.9 項で述べたように，ユーザに提供される結果レイアウトは，この 2 つの候補レイアウトから表 6.1 のように各目標の結果値の比較と連結子の変換を行った回数 (図 6.20)，変換優先順位 (5.4 節) の比較によって選択される．この結果，2,260pixel の幅をもつレイアウトが結果レイアウトとして選ばれる．2,190pixel 幅のレイアウトを生成するために 12 カ所の連結子の変換されたが，統一変換が適用されたので比較する際の変換数は 2 回 (統一変換が行われたのが 1 回，それ以外に 1 回) となる。

表 6.1: 結果レイアウトの選択

基準	2190	2260
幅占有率	95%	98%
充填率	90%	68%
連結子変換の回数	12 → 2	1
戦略優先順位	2	2

提案手法によるレイアウト変換コスト削減の結果で処理を高速化することが可能になる．図 4.4 (p.4.4.1) の全数探索の実行時間によると対象になる連結子の個数が 18 個で実行時間が 10 秒に近くなっている．実験は Intel Core2 Duo CPU 2.83GHz，2GB RAM の環境で行った．図 2.12 の ACTIVIEW の問合せ文にある連結子は 26 個で，この 26 個の連結子に対して全数探索を行うと非常に時間がかかってしまい，ユーザのアクセスに動的な対応ができなくなる．しかし，本提案手法によると，13 個の連結子がレイアウト変換戦略の対象になり，これ以外の 13 個の連結子が全数探索の対象になる．この 13 個の連結子に対しても，本論文で再設計した ACTIVIEW システムでは，連結子が現れるネスト別，中括弧別に行うため，全数探索によるレイアウト変換コストを削減することが可能だった．図 4.4 の全数探索にかかる実行時間から考えると全数探索の対象になる連結子を 15 個以下に減らすとユーザアクセスに動的な対応が可能になると判断できる。

そして，Web ビューの生成時間は関係データベースに格納されているターゲットデータの検索結果の個数 (レコード数) に大きく依存する．そこで，本論文での実行時間は最終結果レイアウトを SuperSQL システムに渡す前の ACTIVIEW によるレイアウト変換時間，すなわち元の問合せ文を書き直す時間のみ対象にすることで関係データ

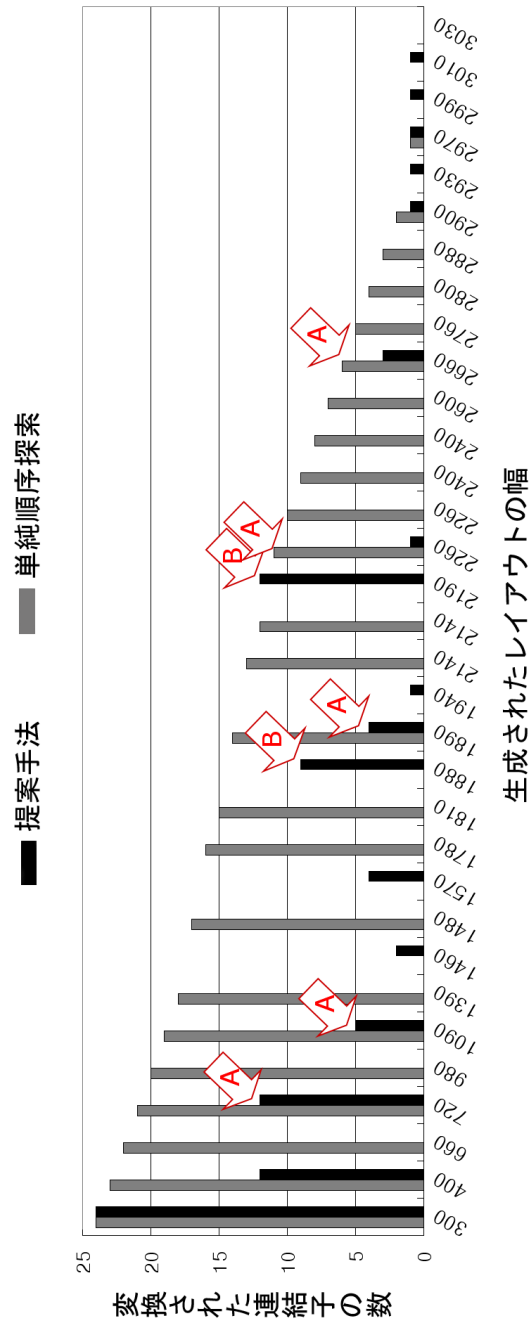


図 6.20: 変換された結合子の数

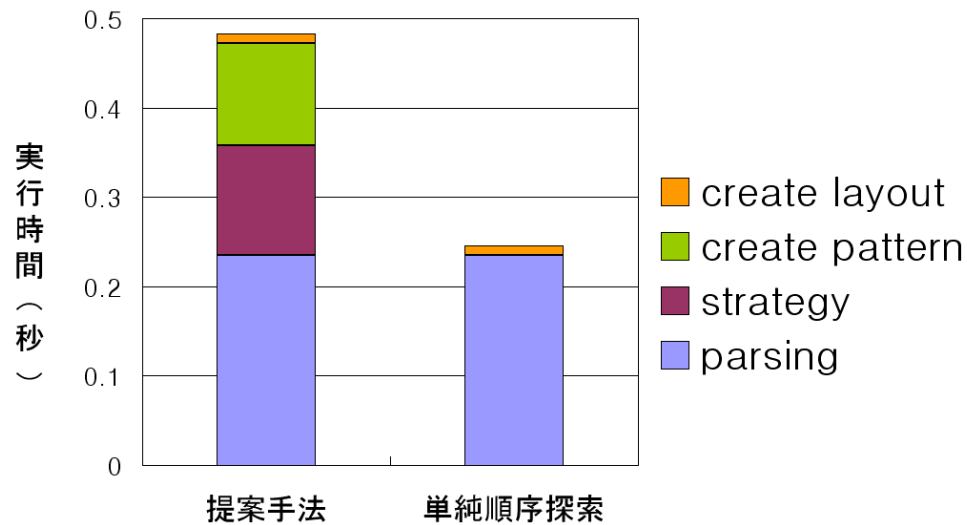


図 6.21: 実行時間

ベースの検索結果の量には影響を受けないようにした。

図 6.21 は図 6.17 と図 6.20 に現れる最終結果レイアウト 1 つを生成する時間である。単純順序探索は候補レイアウトを生成するための、レイアウトパターンの生成時間 (create pattern, 全数探索が必要な部分) と本論文で提案するレイアウト変換戦略を行う時間 (strategy) がないので問合せ文の分析 (parsing) 後、レイアウト生成に必要な時間 (create layout)³だけ必要とする。

単純順序探索によるレイアウト変換に比べては処理時間が遅くなるが、全数探索と比べると非常に速くなったことが分かる。

全数探索のように生成可能なレイアウトに対してすべての比較を行っていないが、提案する戦略に従うとユーザに対して良いレイアウトの提供が可能になる。

図 6.22 に実際の端末機を利用し ACTIVIEW をアクセスした結果を示す。図 6.22 から端末機の画面表示幅に合わせたレイアウトが生成されていることが確認できる。同じ端末機であっても図 6.23 のように表示画面を横に変更できる端末機に対しては、それに合わせてレイアウトの変換が行われている。

³レイアウト生成に必要な時間は Web ビューを生成する時間ではなく、元の問合せ文を書き直す処理を行う時間を意味する。

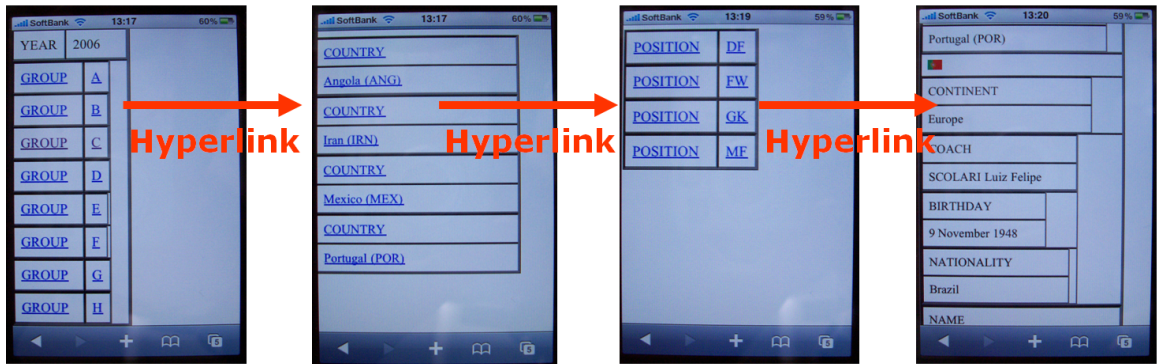


図 6.22: 実際の端末機からのアクセス結果



図 6.23: 同一端末機であっても異なる表示画面に対するレイアウト結果

6.3 実際の Web ページへの適用

ACTIVIEW の問合せ文によって構造化された Web ビューに対するレイアウトの変換について、実際の Web ページの例から適用可能性を述べる。実際にインターネット上でサービスされている Web ページは通常の HTML によって作成されているものであり、今まで述べた ACTIVIEW での直接的な構造化への適用は困難である。そのうえ、商用 Web ページは複雑な構造を持っているため、ACTIVIEW の問合せ文による構造定義だけではすべてのページを作成することは困難である。しかしながら、商用 Web ページの中で ACTIVIEW の問合せ文によって構造化可能なページを取り出して、このページが ACTIVIEW の問合せ文によって構造化されているとの仮定した後、レイアウトの変換について議論することは可能だと考える。

実際の Web ページが ACTIVIEW の問合せ文によって構造化されていると仮定したうえで、本論文で提案したレイアウト変換手法によって再構成されるレイアウトの実際の例を取り上げる。実際の Web ページへの応用例を従来の研究と比較して本論文の提案手法の実用性と妥当性を示す。実験対象にする Web ページは Alexa 社 [31] の分析による日本のアクセス数上位 40 位のサイト⁴を中心にする。日本のアクセス数上位 40 位の Web ページを分析した後、カテゴリ化可能なページへの適用、データ集約的な Web ページへの適用、統計データあるいは複雑なテーブル型ページへの適用の 3 つに分類した比較結果を、代表的な Web ページを取り上げて述べる。

図 6.24 (日本のアクセス数上位 15 位) のようにリンク先の最後のページが巨大なデータをもつページであるために、携帯端末や PDA のように幅狭い端末機からアクセスするには見にくいページである Web ページや、モバイル端末機からアクセスするためにはデータの縮約による構造の再設計が必要なページであると判断した Web ページに対しては今回の実験では対象外にする。

そのほかに、対象にした Web サイトのうち、図 6.25 の mixi ページ (日本のアクセス数上位 8 位) と Ameba ページ (日本のアクセス数上位 20 位) のような会員登録が必要な閉ざされた Web ページは実験の対象にならない。

複雑な構造をもつ Web サイトと比べて Google サイト (日本のアクセス数上位 2 位と 9 位) のように極端に単純化された構造をもつ Web ページへの適用は非常に簡単にできる。

元のページが単純な構造のページであるため、ACTIVIEW の問合せ文でこの構造を定義するだけでユーザ環境に適用されたレイアウトへの変換が可能になる。実際の応用例を図 6.26 に示す。しかし、Google サイトのように構造化の意味がないくらい単純な構造の Web ページに対しても実験の対象外にする。

⁴2009 年 12 月 31 日基準

表 6.2: 日本のアクセス数上位 40 位の Web サイト

順位	サイト URL	順位	サイト URL
1	www.yahoo.co.jp	21	www.infoseek.co.jp
2	www.google.co.jp	22	www.geocities.jp
3	www.fc2.com	23	www.ocn.ne.jp
4	www.youtube.com	24	www.live.com
5	www.rakuten.co.jp	25	www.seesaa.net
6	www.ameblo.com	26	www.kakaku.com
7	www.livedoor.com	27	www.yourfilehost.com
8	www.mixi.jp	28	www.blogger.com
9	www.google.com	29	www.livedoor.biz
10	www.wikipedia.org	30	www.dmm.co.jp
11	www.goo.ne.jp	31	www.yahoo.com
12	www.amazon.co.jp	32	www.atwiki.jp
13	www.nicovideo.jp	33	www.twitter.com
14	www.msn.co.jp	34	www.jugem.jp
15	www.2ch.net	35	www.exblog.jp
16	www.nifty.com	36	www.cocolog-nifty.com
17	www.biglobe.ne.jp	37	www.so-net.ne.jp
18	www.hatena.ne.jp	38	www.excite.co.jp
19	www.sakura.ne.jp	39	www.facebook.com
20	www.ameba.jp	40	www.microsoft.com

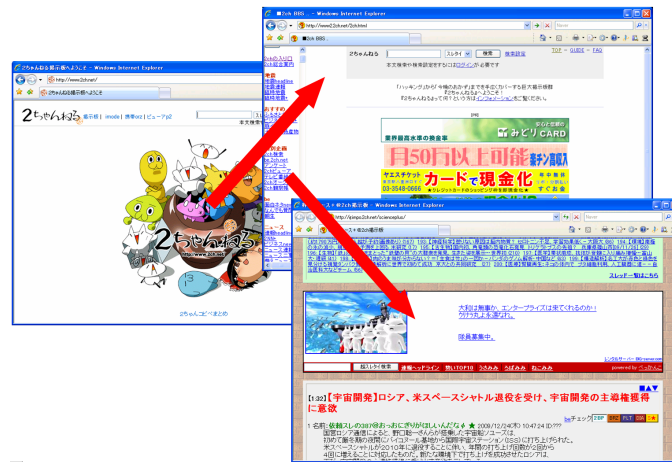


図 6.24: データの縮約による構造の再設計が必要なページの例

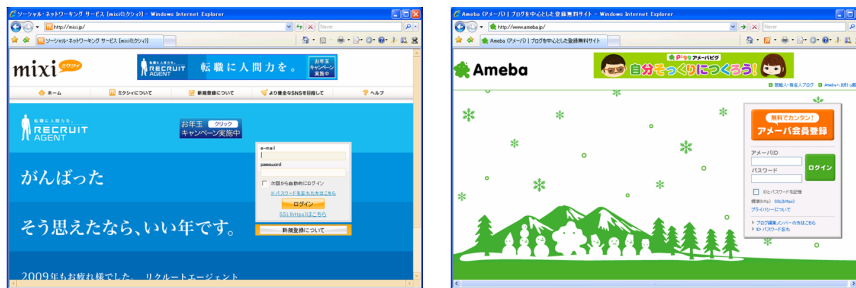
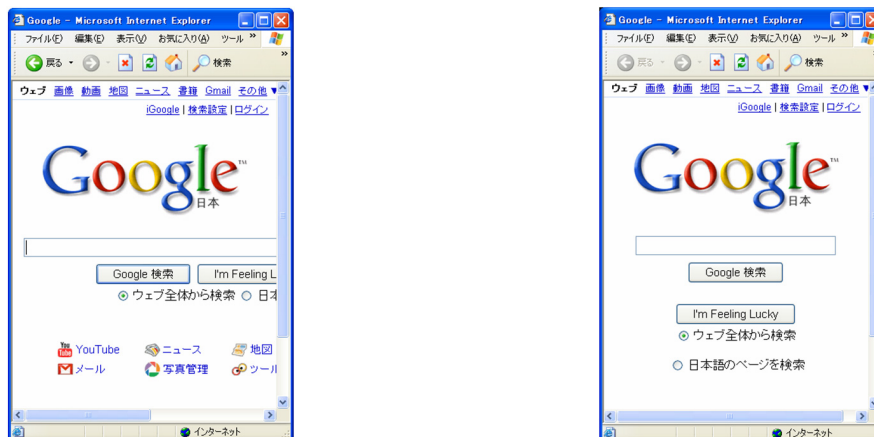


図 6.25: 閉ざされた Web ページの例



(a) オリジナル Google サイト (b) 再構成された Google メインページ

図 6.26: Google ページへの応用例



図 6.27: Yahoo Japan サイトのメインページ

6.3.1 カテゴリ化可能なページへの適用結果と比較

日本のアクセス数上位 1 位は Yahoo Japan サイト [32] であった (図 6.27)。Yahoo Japan のメインページはかなり複雑な構造を持っているため、ACTIVIEW の問合せ文だけでこのページを生成することは困難である。しかしながら、パソコン以外の端末へのサービスを支援するために構造を単純化したと仮定して、このメインページを ACTIVIEW の問合せ文に定義することは可能であると考えられる。

Yahoo Japan サイトのメインページを見ると、図 6.27 の①はメインカテゴリであり、図 6.27 の②は人気カテゴリを、このカテゴリに含まれる記事ページは図 6.27 の③に表示される。それと、天気情報などの独立 (図 6.27 の④) に表示される情報をもつ構造になっている。

ACTIVIEW の問合せ文でメインカテゴリとこのカテゴリに含まれるサブカテゴリ、このサブカテゴリ別の記事のみ表示するように単純化した構造を定義すると、

```
GENERATE ACTIVIEW [ maincategory.title,
                    [ subcategory.title, [ article.content ]! ]! ]!
```

のように ACTIVIEW の問合せ文でレイアウトの定義が可能になる。しかしながら、ACTIVIEW の問合せ文で Yahoo Japan サイトのメインページのように複雑な Web



図 6.28: カテゴリのみを取り出して生成した Web ページの例

ページを全般的に定義することは難しい。AJAX による Web ページ作成に対して小林 [33] が研究したものがあがるが細かい Web ページの実現はまだできない。しかし、この ACTIVIEW の問合せ文のように Yahoo Japan サイトのメインページからカテゴリの情報のみ取り出すと、ACTIVIEW によってネスト構造化を行った Hyperlink ページが図 6.28 のように生成できる。このように、単純化されたデータのみを表示する Web ページを作成すると ACTIVIEW への応用が可能になる。

実際の Web サイトのメインページに対する ACTIVIEW の適用は、カテゴリ別に単純化した、すなわち ACTIVIEW の問合せ文の定義によって再構成される Web ビューを対象にする。実際の Web サイトの大部分のメインページは図 6.29 のように、ACTIVIEW による再構成の対象になるカテゴリが限られている。すなわち、メインページに現れる Web サイトのカテゴリが ACTIVIEW の問合せ文の属性になる。

図 6.30 は Yahoo Japan サイトの Web ページにあるコンテンツページにたどりつくまでのカテゴリ連結の一例である。商品のコンテンツページまでたどるのに 6 段階のカテゴリが繋がれている。

Yahoo Japan サイトにあるカテゴリを基に ACTIVIEW を用いて構造を定義するために設計したデータベーススキーマの一部を図 6.31 に示す。

図 6.31 のスキーマを基に ACTIVIEW を用いて、図 6.32 のようなレイアウトの Web ビューを生成する。

従来のレイアウト変換手法によるレイアウト変換と提案手法によるレイアウトの変換の結果の比較を、図 6.32 を基に行う。

図 6.32 の初期レイアウトとこのレイアウトを定義した ACTIVIEW の問合せ文（図



図 6.29: 実際の Web サイトのメインページの例

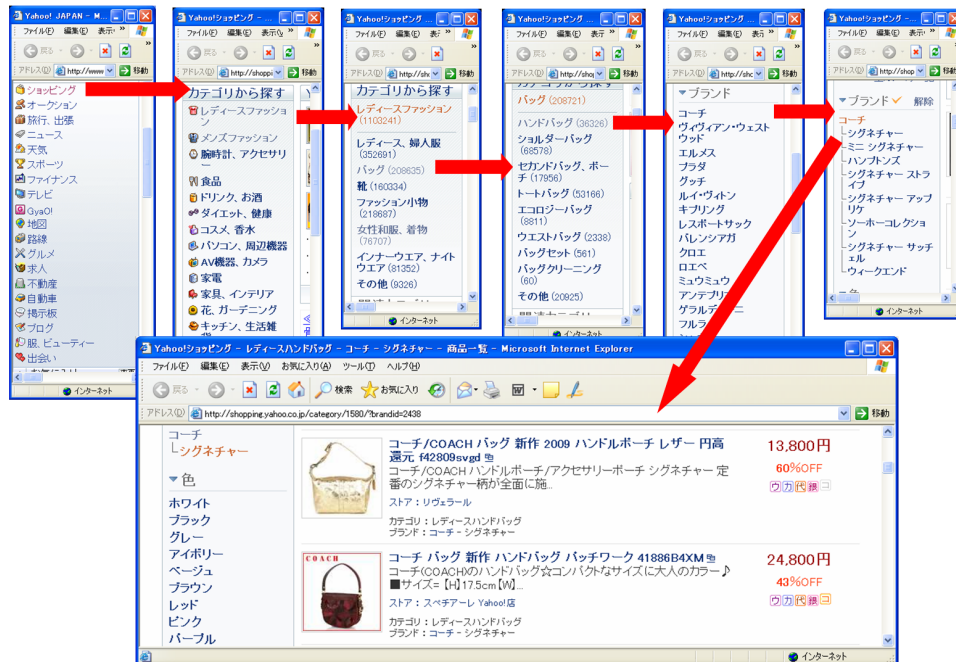


図 6.30: Yahoo Japan サイトのカテゴリ例

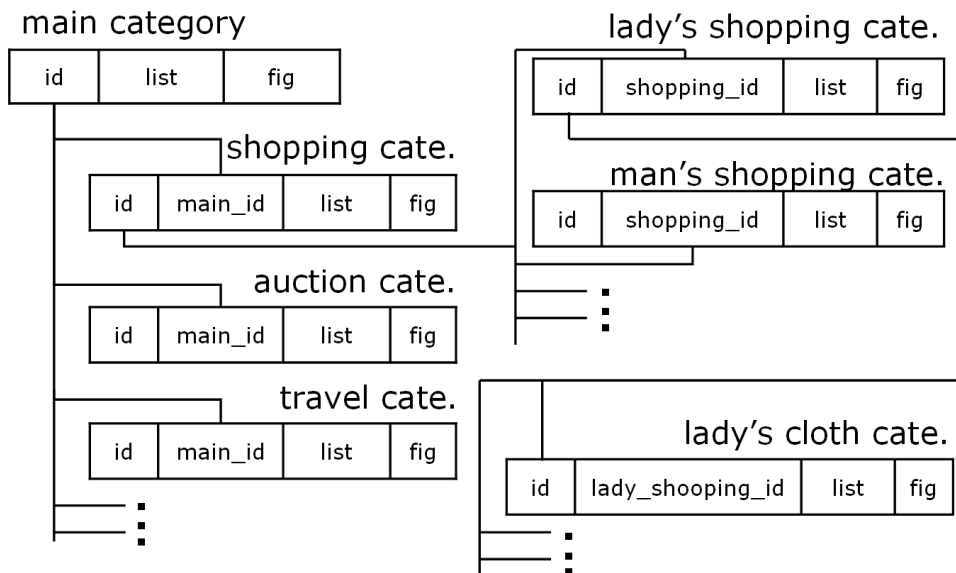


図 6.31: Yahoo ページを再構成するためのスキーマ

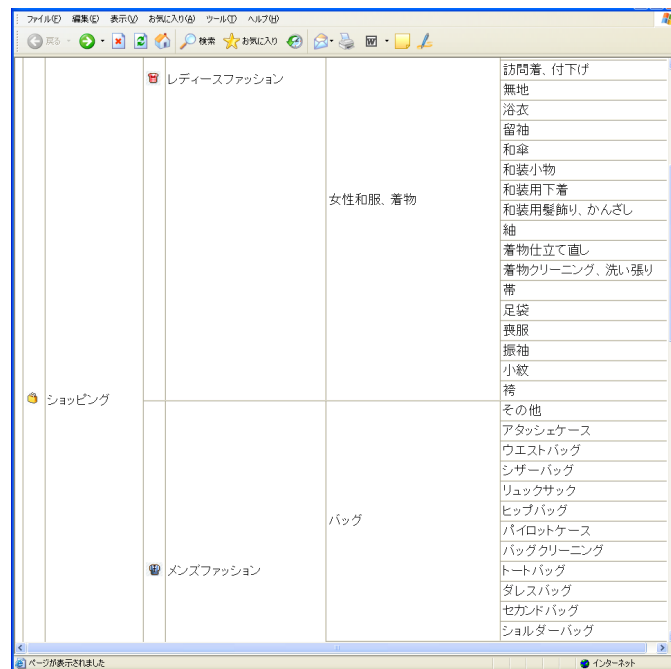


図 6.32: ACTIVIEW を用いて再構成した Yahoo ページ

6.33⁵⁾ から分かるように，同一ネスト内に存在する属性の数がほぼ 1 個だけであるので，本論文で提案したレイアウト生成にかかる変換コスト削減などの長所を生かしたレイアウト戦略を組む余地がない。

図 6.32 の初期レイアウトは 1,440pixel の幅をもつレイアウトを生成する．単純順序探索を行うと連結子の個数である 8ヶ所でのレイアウト変換を行った候補レイアウトを生成する．これに比べ，提案手法によるレイアウト変換は，

5.3 節の統一変換可能な連結の変換を優先する．

戦略に従い，中括弧を同時に変換した場合としなかった場合の 2通りと，一貫性のあるレイアウトを生成するための戦略（5.4 節）のうち，

- (6) これ以外は，全数探索によって生成されるレイアウトのうち，高い目標の結果値をもつレイアウトを優先する．

戦略に従って，中括弧内での連結子を抜いた 6ヶ所の連結子に対する全数探索を行った結果を候補レイアウトして生成する．

図 6.33 の ACTIVIEW の問合せ文を基に，単純順序探索と，提案手法に従ったレイアウト変換によって生成される結果レイアウトの幅の種類と充填率を図 6.34 に示す．

⁵⁾属性のネスト定義を分かりやすくするために，例文として簡単に書き直した問合せ文である．

```

GENERATE ACTIVIEW
  [ { maincate.fig@{width=30}, maincate.list@{width=140} },
    [ { 1stsubcate.fig@{width=30}, 1stsubcate.list@{width=230} },
      [ 2ndsubcate.list@{width=250},
        [ 3rdsubcate.list@{width=230},
          [ 4thsubcate.list@{width=220},
            [ 5thsubcate.list@{width=210},
              [ item.pic@{width=100} ]!
            ]! ]! ]! ]! ]! ]!
FROM ...

```

図 6.33: 再構成される Yahoo ページを生成する ACTIVIEW の問合せ文

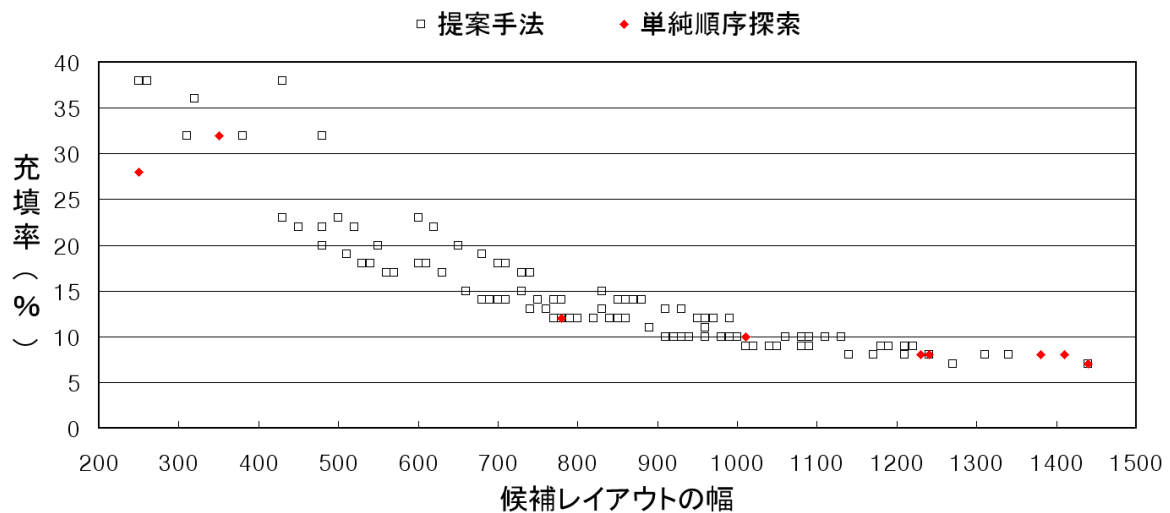


図 6.34: ACTIVIEW の問合せ文 (図 6.33) によって生成される候補レイアウトの幅の種類と充填率

図 6.34 の結果レイアウト 1 つを生成するための実行時間を次に示す .

- 単純順序探索 : 0.265 秒
- 提案手法 : 0.483 秒

図 6.34 に示すように , 提案手法の方が単純順序探索と比べて遅いが , はるかに多い種類の候補レイアウトを生成している . 多様な幅をもつ候補レイアウトが生成できることは様々なユーザ表示画面サイズに対応できることを意味する .

図 6.32 はネスト構造によって反復される長いレイアウトを生成する構造 (検索結果の個数が多い) をもっている . ユーザの表示画面の長さが 400pixel であると仮定したう⁶え , 画面サイズより 3 倍以内の長さをもつレイアウトを生成するように設定した場合 , 単純順序探索と提案手法の両方ともすべてのカテゴリに対してリンク変換を行ったレイアウトを生成した (図 6.35) .

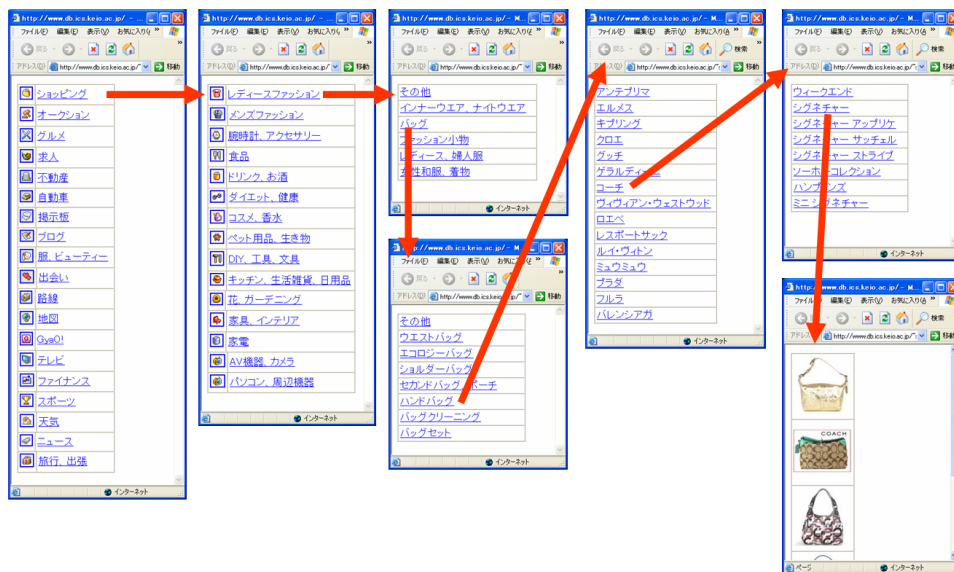


図 6.35: ACTIVIEW によるリンク変換

長さ目標が設定されていない場合は , 元のレイアウトの構造的な制約 , すなわちすべての属性間の関係がネスト関係になっているため , 横連結されるとサブネストが反復的に現れてしまうので効率的な空間活用ができない . このため , 単純順序探索と提案手法の両方とも低い充填率のレイアウトを生成してしまう (図 6.36) . 深いネストをもつレイアウトであるほど縦横連結の変換だけでは , 反復して現れるサブネストが多いので充填率は大きく落ちてしまう .

⁶携帯端末や PDA の画面サイズに基づく .

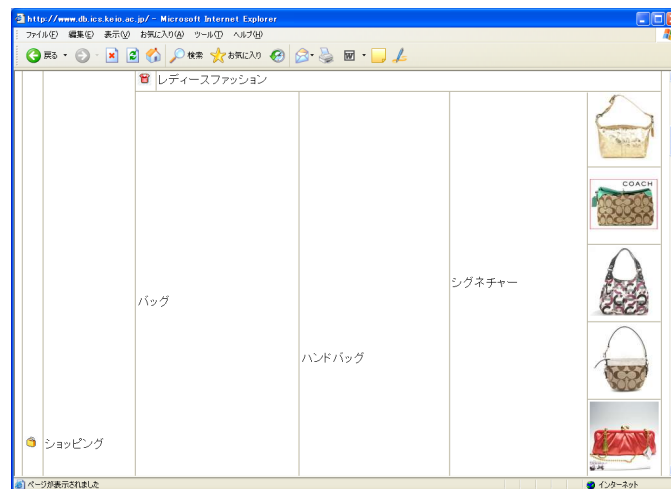


図 6.36: 低い充填率の結果レイアウト

図 6.37 に実験で使ったカテゴリ化されている Web ページの他の例を示す。

図 6.37 の①のように実際の Web ページは異なる長さのサブページをもつ場合が多い。この場合は、5.2.4 項で述べたようにサブページの長さの平均値を基準として変換を行う。そのため、実際の結果レイアウトと予測した長さが異なってしまう問題がある。今後、統計値を利用するなど、解決手法を考えるべきである。

図 6.38 は実験で利用したレイアウトの構造を分かりやすくするため、元の ACTIVIEW の問合せ文を簡単に記述したものである。図 6.38 は 5 段階の深さのネスト構造をもつレイアウトを定義している。ニュースの記事 (4thsubcate.news) などの長い文章から成るコンテンツネストをもっているが、ACTIVIEW の問合せ文では通常の携帯端末の幅である 300pixel をセル幅の値として指定した。

図 6.38 の ACTIVIEW 問合せ文から生成される初期レイアウトは図 6.39 のようになる。

図 6.38 の ACTIVIEW 問合せ文を基に、単純順序探索と、提案手法に従ったレイアウト変換によって生成される結果レイアウトの幅の種類を図 6.40 に示す。

図 6.34 の結果レイアウト 1 つを生成するための実行時間を次に示す。

- 単純順序探索： 0.265 秒
- 提案手法： 0.487 秒

実験結果から見るように、単純順序探索と比べて、提案手法に従ったレイアウト変換の方が多種類の結果レイアウトを生成することができた。



図 6.37: カテゴリ化されている Web ページの例

GENERATE ACTIVIEW

```
[ maincate.list@{width=110},
  [ { 1stsubcate.list@{width=140}, 1stsubcate.explain@{width=140} },
    [ 2ndsubcate.list@{width=120},
      [ 3rdsubcate.list@{width=300},
        [ 4thsubcate.news@{width=300} ]!
      ]! ]! ]! ]!
```

FROM ...

図 6.38: カテゴリ化されている Web ページを生成する ACTIVIEW の問合せ文

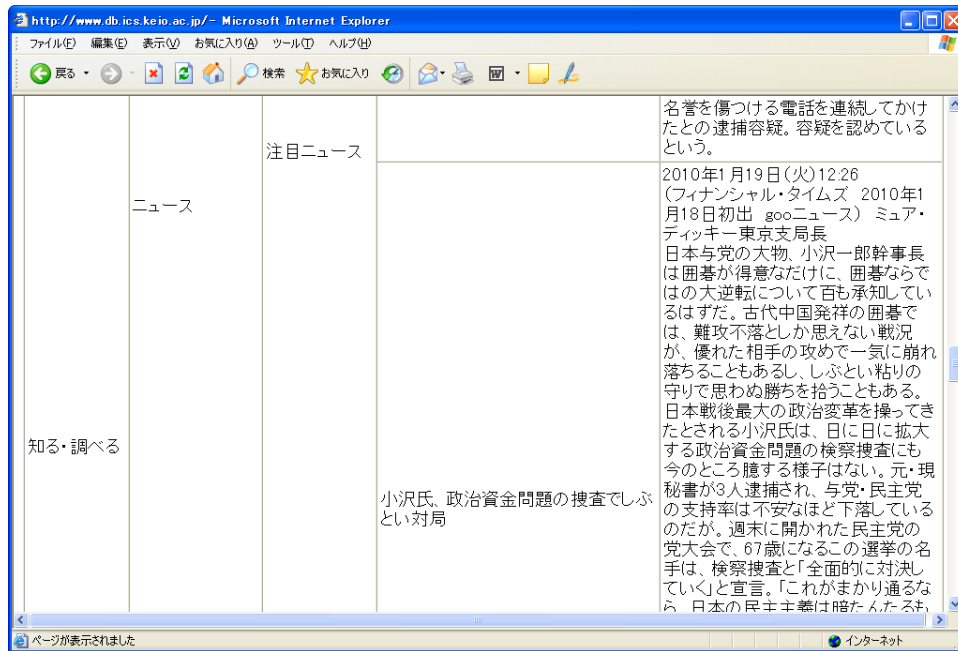


図 6.39: 図 6.38 の ACTIVIEW 問合せ文から生成される初期レイアウト

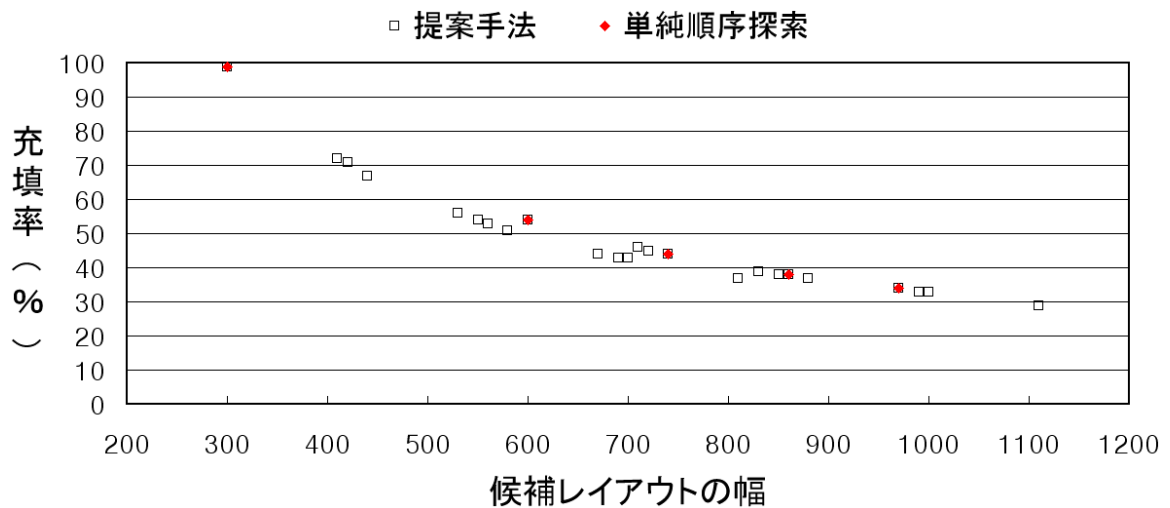


図 6.40: ACTIVIEW の問合せ文 (図 6.38) によって生成される候補レイアウトの幅の種類と充填率



図 6.41: データ集約的な Web ページ

カテゴリ化されている様々な Web ページを対象に ACTIVIEW によって再構成を行い、単純順序探索との比較を行った結果、単純順序探索より良い結果レイアウトを獲得した。

カテゴリ化可能な様々なページを対象に、ACTIVIEW を用いて再構成した Web ビューを生成した後、この再構成した Web ビューに対してレイアウト変換を行った。この結果、単純順序探索と比べて多様な候補レイアウトを生成することができ、より良い結果レイアウトを獲得することが可能になった。しかしこの実験では、ACTIVIEW の問合せ文によって定義された初期レイアウトの各サブネストがほぼ 1 個の属性によるカテゴリ化という構造的な制約があったため、ほとんど全数探索を行ってしまい、提案手法の戦略の長所を生かすことができなかった。

6.3.2 データ集約的な Web ページへの適用結果の比較

本項では Web サイト内のテーブル型のページ、あるいはテーブル型に変換可能なページを対象にした、提案レイアウト変換戦略による実験結果について述べる。図 6.41 は Web ページのうち、データ集約的な Web ページの一例を示す。

データ集約的な Web ページに関しても 6.3 節と同じく、まず元の Web ページに対し

路線・最寄駅 所在地	徒歩 バス	面積 間取り	築年 数	賃料 管理費	敷金・保証 金 礼金 償却・敷引	外観写真	間取り図	詳細	マイ 物件	選択
東急東横線 日吉駅 神奈川県横浜市北区日吉本町3丁目	10分 -	18.63m ² 1K	20年	6.3万円 -	- 1ヶ月			詳細表示	☆	<input type="checkbox"/>
東急東横線 日吉駅 神奈川県横浜市北区日吉2丁目	8分 -	21m ² 1K	20年	6.0万円 -	1ヶ月/ 1ヶ月			詳細表示	☆	<input type="checkbox"/>
東急東横線 日吉駅 神奈川県川崎市幸区南加瀬4丁目	20分 -	49.14m ² 3DK	17年	10.0万円 5000円	1ヶ月/ 1ヶ月			詳細表示	☆	<input type="checkbox"/>
東急東横線 日吉駅 神奈川県横浜市北区鶴屋敷3丁目	7分 -	26.87m ² 2K	18年	7.9万円 4000円	1ヶ月/ 1ヶ月			詳細表示	☆	<input type="checkbox"/>

図 6.42: 代表的なテーブル型の Web ページの例

GENERATE ACTIVIEW

[路線・最寄駅・所在地, 徒歩・バス, 面積, 築年数,
賃料・管理費, 保証金など, 外観写真, 間取り図, 詳細]!

FROM ...

図 6.43: テーブル型の Web ページを生成する ACTIVIEW の問合せ文

て ACTIVIEW を用いた再構成を行う。この再構成した Web ビューに対するレイアウト変換結果について述べる。

図 6.42 のようなテーブル型の Web ページを生成する, ACTIVIEW の問合せ文を図 6.43 のように定義する。この ACTIVIEW の問合せ文は生成するレイアウトの構造や内容を分かりやすいように表現した問合せ文である。

図 6.43 のように 1 つのテーブルを生成する問合せ文は開発者による制約指定がないと, 問合せ文を構成する属性間の関連性をとることが難しくなるので全数探索を行ってしまう。

全数探索を行うと幅目標に対しては最適化されたレイアウトを生成するが, 意味的に一貫性のあるレイアウトであるとは限らない。しかし, 図 6.44 のように各属性間に関連性 (中括弧による) が付けられると, 意味的に分かりやすい一貫性のあるレイアウトが生成できる。

Yahoo Japan サイトにあるコンテンツページのうち, 天気情報や地図情報のように行政区域によって区別されている情報 (図 6.45) は, 情報別にグルーピングすることが可能である。このような情報をもつデータページは特に ACTIVIEW への応用がし

GENERATE ACTIVIEW

[{ 路線・最寄駅・所在地, 徒歩・バス }, 面積, 築年数],
 { 賃料・管理費, 保証金など }, { 外観写真, 間取り図 }, 詳細]!

FROM ...

図 6.44: 関連性を付けた ACTIVIEW の問合せ文



図 6.45: Yahoo Japan サイトのうち、地図情報と天気情報の Web ページ

やすいページである。

Yahoo Japan サイトの天気情報ページからデータを取り出して ACTIVIEW によるテーブル構造を作成し、レイアウト変換を行ってみる。

まず、図 6.46 のように簡単にスキーマを定義してデータベースを作成する。

このスキーマに基づく ACTIVIEW の問合せ文を図 6.47 のように定義した。

この問合せ文は、レイアウトの構造を分かりやすくするために、実際の ACTIVIEW の問合せ文を分かりやすいように簡単に示したものである。

元の ACTIVIEW の問合せ文によって生成される初期のレイアウトを図 6.48 に示す。各県にある代表的な都市の天気情報は、県単位の地域要素にまとめられ、またこの情報はもっと大きい地域単位の要素でまとめられるテーブル型の Web ページを生成している。各都市の天気情報はある日の天気（晴れ、曇時々晴、...）などを示している。オリジナル Web ページ（図 6.49）に比べて単純化されているが、必要な情報はすべて表示している。そのうえ、ACTIVIEW で定義すると元の Web ページでは不可能なレイアウトの動的再構成が可能になる。

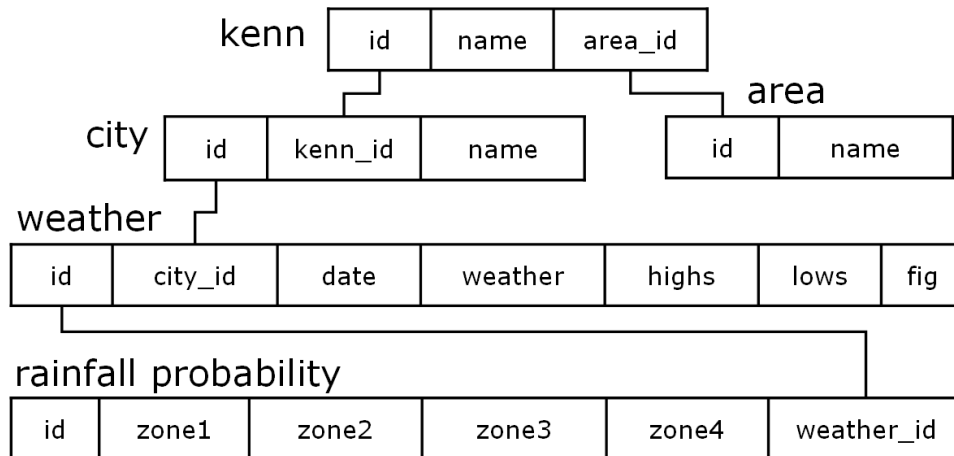


図 6.46: 天気情報データベースのテーブルスキーマ

```

GENERATE ACTIVIEW
[ area.name,
  [ kenn.name,
    [ c.name, c.date, c.weather, c.fig, c.highs, c.lows,
      [ rp.zone1, rp.zone2, rp.zone3, rp.zone4 ]!
    ]! ]! ]!
FROM city c, Rainfall-Probability rp
  
```

図 6.47: 天気情報 Web ビューを生成する ACTIVIEW の問合せ文

都道府県	市町村	日付	天気	天気アイコン	最高気温	最低気温	降水確率	時間帯	風	波
東京都	大島	20091222	晴れ		12[+1]	5[0]	10	0	0	0
	東京	20091222	晴れ		12[+3]	3[-1]	0	0	0	0
	八丈島	20091222	晴時々曇		13[+3]	8[-1]	10	10	10	0
長野県	父島	20091222	晴時々曇		20[-2]	17[-1]	20	20	10	20
	松本	20091222	晴時々曇		7[+4]	-3[0]	10	10	10	10
千葉県	長野	20091222	曇時々晴		4[+3]	-3[-1]	20	30	30	20
	飯田	20091222	晴時々曇		5[+1]	-4[-2]	10	10	10	10
神奈川県	館山	20091222	晴れ		12[+2]	1[-2]	10	10	0	10
	千葉	20091222	晴れ		11[+1]	2[-2]	0	0	0	0
	狹子	20091222	晴れ		12[+3]	2[-1]	0	0	0	0
新潟県	横浜	20091222	晴れ		11[+1]	3[-1]	10	10	0	0
	小田原	20091222	晴れ		11[+2]	1[+1]	10	10	0	0
山梨県	高田	20091222	曇後雨		9[+7]	0[-1]	30	30	50	30
	新潟	20091222	曇後雨		7[+5]	2[+2]	40	70	90	80
	相川	20091222	曇時々雨		9[+6]	4[+3]	40	80	80	40
山梨県	長岡	20091222	雪時々曇		6[+4]	1[+2]	60	50	70	50
	河口湖	20091222	晴れ		6[+3]	-6[-1]	0	0	0	0
山梨県	甲府	20091222	晴れ		11[+3]	-3[0]	0	0	0	0

図 6.48: 天気情報ページの初期レイアウト

Yahoo!天気情報トップ > 関東・信越 > 神奈川県 > 東部(横浜)

[PR] [「Visual Studio 2010」で大幅に時間短縮 今すぐベ...](#)

北日本は猛ふぶき・大雪に警戒を (チーム森田の“天気で斬る!”) - 週

今日の天気 2009年12月22日 5時00分発表

12月22日(火) 最高気温(°C)[前日差] 11 [+1] 最低気温(°C)[前日差] 3 [-1]

曇後晴

降水確率(%)	—	10	0	0
時間帯(時)	0-6	6-12	12-18	18-24

風: 北の風後北西の風 波: 1メートル

図 6.49: Yahoo の天気情報ページ

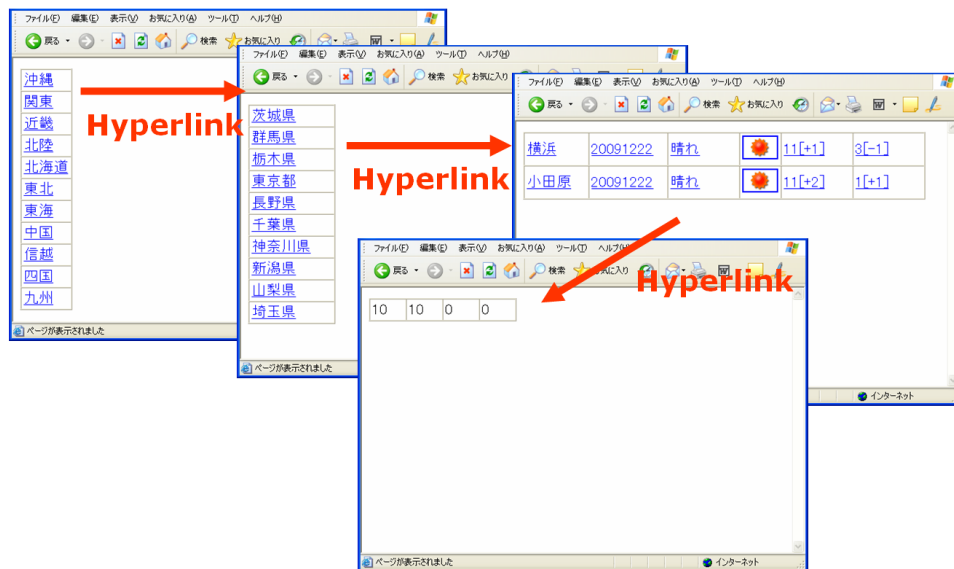


図 6.50: レイアウト変換戦略による再構成

図 6.48 の初期レイアウトを、ACTIVIEW のレイアウト変換戦略によって変換したレイアウトの結果は図 6.50 のようになる。幅目標の結果値は低く見えるが意味的な範囲をもとに変換を行った結果、一貫性のあるレイアウトの生成が可能になる。同じ表示画面の幅をもつユーザ環境であっても長さ目標の変数が異なることによって異なる結果レイアウトを生成する場合もある(図 6.51)。長さ目標の変数に大きい数を指定することによって長いレイアウトのページが生成されて縦スクロールの手間は大きくなるが、幅目標の結果値の良い結果レイアウトを生成している。長さ目標の変数値は開発者によって指定される。開発者は端末の種類やユーザの好み、元の問合せ文の構造定義などの様々な要素を考えた上に環境に従い異なる値を指定する。

6.3.1 項のメインページへ適用したレイアウトに対する変換結果より、提案手法の長所を生かした結果を抽出できると判断したが、データ集約的な Web ページであってもネスト型ではない場合は、提案手法の長所を十分生かすことができなかった。

データ集約的な Web ページへのレイアウト変換結果を比べてみると、図 6.47 の天気情報 Web ビューを生成する ACTIVIEW の問合せ文のように、関係データベーステーブル別に正しくネスト構造を定義した問合せ文、すなわち開発者制約に対する依存性が高かった。したがって、ACTIVIEW の問合せ文を定義する開発者による属性間の関連性の定義によって、適用されるレイアウト変換戦略が変わるので、この定義に従って生成する結果レイアウトへの影響も最も高い。

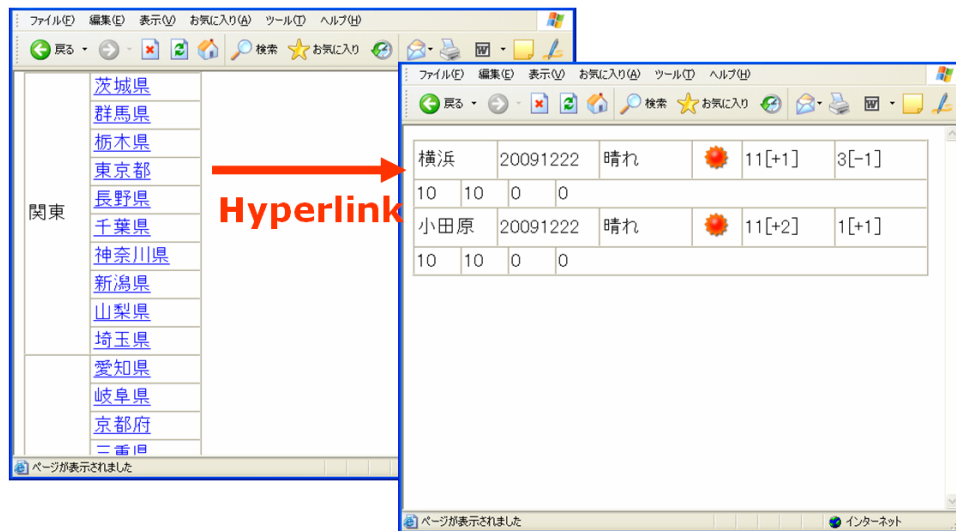


図 6.51: 異なる長さ目標値に従った結果レイアウト

6.3.3 統計データあるいは複雑なテーブル型ページへの適用結果の比較

本研究で提案しているレイアウト変換戦略の長所を一番生かす適用分野は、統計データなどの複雑なテーブル型をもつ情報ページ（図 6.52）、あるいは 1 つのテーブルの中にサブカテゴリをもつテーブル型への変換適用である。

図 6.53 のようにカテゴリ化されたテーブル型の情報中心の Web ページ [34] が、ACTIVIEW の対象にふさわしいページである。すなわち、元々テーブル型でカテゴリ化されている Web ページなので ACTIVIEW への構造化定義に適用しやすいと言える。

ACTIVIEW によるレイアウト変換の実験を行うために、まず図 6.54 のようにスキーマを定義してデータベースを生成する。実験のために Web ページにある一部のデータを収集して ACTIVIEW でのレイアウト変換を行う。

図 6.53 に各ノートパソコン会社別に分けられた商品情報と、リンク先のページの情報を表す。これに対する ACTIVIEW の問合せ文を図 6.55 のように定義する。

この ACTIVIEW の問合せ文によって生成される初期レイアウトは図 6.56 のようになる。図 6.53 の価格.com のオリジナルテーブルとほぼ同じレイアウトの結果を生成していることが分かる。

実際、図 6.53 のオリジナルテーブルは 1 つのテーブルでありながら図 6.57 に表すようにいくつかの内容で分けられる。

ACTIVIEW によってレイアウトを再構成する際、これらの内容、すなわち要素間の関連性を参考にしたレイアウトの変換を行わないと図 6.58 のように意味的に崩れてし

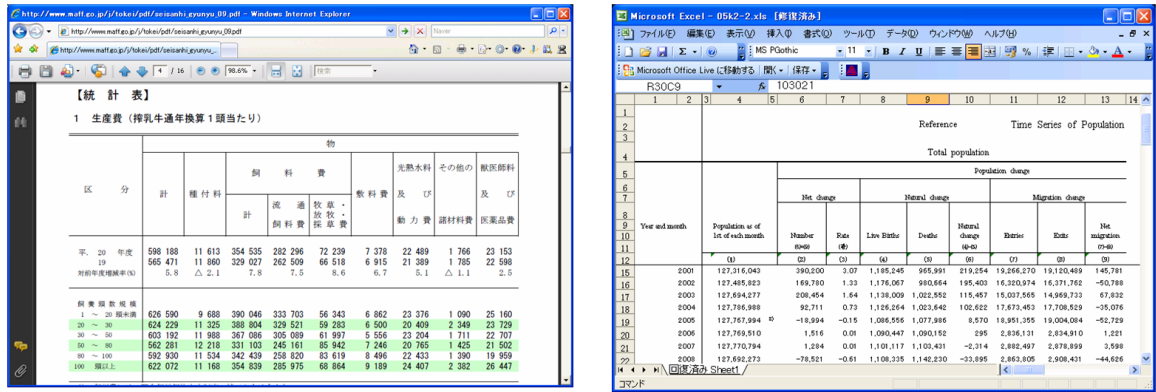


図 6.52: 統計データの例

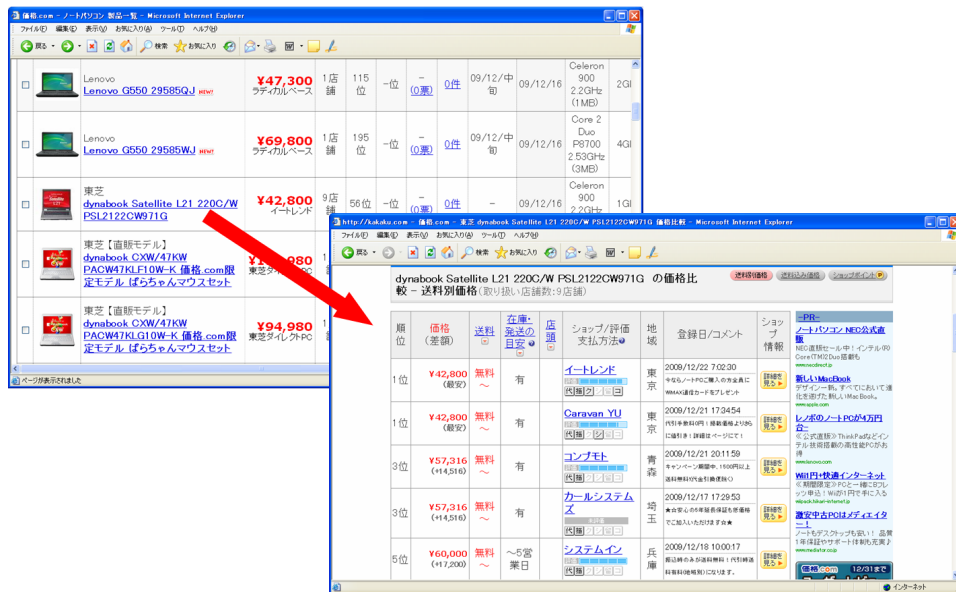


図 6.53: 複雑なテーブル型の Web ページ

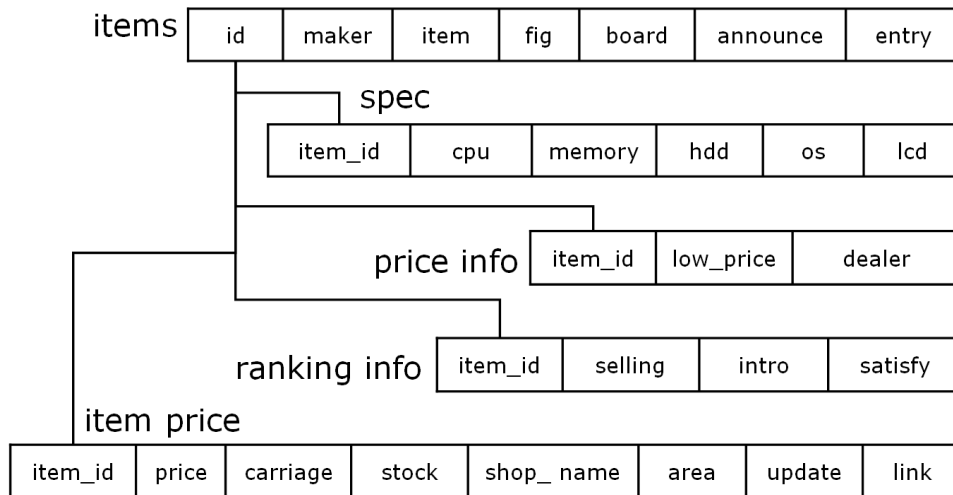


図 6.54: Kakaku データベースのテーブルスキーマ

GENERATE ACTIVIEW

```

[ items.maker,
  [ items.fig, items.item%
    [ itemprice.price, itemprice.carriage, itemprice.stock,
      itemprice.shopname, itemprice.area, itemprice.update,
      itemprice.link
    ]!,
    priceinfo.price, priceinfo.dealer,
    rankinginfo.selling, rankinginfo.intro,
    rankinginfo.satisfy,
    items.board, items.announce, items.entry,
    spec.cpu, spec.memory,
    spec.hdd, spec.os, spec.lcd
  ]! ]!
FROM ...

```

図 6.55: 図 6.53 のようなテーブルを作成するための問合せ文

メーカー	製品名	型番	価格	在庫	送料	保証	発売日	登録日	CPU	メモリ	HDD	OS	画面	
ACER	Aspire 5739	AS5739G-MD24	111890	23	184	179	0	0	09/12/4	09/11/27	Core 2 Duo P8800 2.66GHz (3MB)	4GB 500GB	Windows 7 Home Premium	15.6インチ
	Aspire 8940G	AS8940G-BR101	184899	18	65	54	5	1	09/12/4	09/11/27	Core i7 720QM 1.6GHz	4GB 1000GB	Windows 7 Home Premium	18.4インチ
	Aspire Timeline 4810T	AS4810TG-A23	45200	6	480	480	0	0	09/12/4	09/11/27	Core i7 720QM 1.6GHz	4GB 1000GB	Windows 7 Home Premium	18.4インチ
ASUS	Eee PC	1101HA-WP	79800	17	4	3	4.73	0	09/12/19	09/12/16	Dual-Core SL2300 1.2GHz (1MB)	2GB 320GB	Windows 7 Home Premium	12.1インチ
	VAIO	Aシリーズ VGN-AW53FB	199704	28	39	35	0	10	09/10/22	09/10/08	Core 2 Duo T9600 2.8GHz	4GB 1000GB	Windows 7 Home Premium	18.4インチ

図 6.56: ACTIVIEW の問合せ文 (図 6.55) によるレイアウト生成

製品名・型番	価格情報	ランキング情報	掲載情報	登録日	スペック情報								
	最安価格	取扱い	売れ筋	注目	満足度	掲載数	発売時期	登録日	CPU種類	メモリー容量	HDD容量	OS	液晶
SONY VAIO Aシリーズ VGN-AW53FB	¥210,483 EGファントム	25店舗	39位	35位	-	10件	09/10/22	09/10/08	Core 2 Duo T9600 2.8GHz(6MB)	4GB	1000GB	Windows 7 Home Premium	18.4インチ
SONY VAIO Aシリーズ VGN-AW73FB	¥214,984 大阪ボンバー	25店舗	102位	81位	-	47件	09/10/22	09/10/08	Core 2 Duo T9600 2.8GHz(6MB)	4GB	1000GB	Windows 7 Home Premium	18.4インチ
SONY【直販モデル】VAIO Aシリーズ VGN-AW83FS	¥189,800 SonyStyle	1店舗	546位	398位	-	0件	09/10/22	09/10/08	Core 2 Duo P8700 2.53GHz(3MB)	2GB	640GB	Windows 7 Home Premium	18.4インチ
SONY【直販モデル】VAIO Aシリーズ VGN-AW93FS	¥244,800 SonyStyle	1店舗	666位	398位	-	0件	09/10/22	09/10/08	Core 2 Duo P8700 2.53GHz(3MB)	2GB	640GB	Windows 7 Home Premium	18.4インチ
SONY VAIO Cシリーズ VPCCW18FJ/P	¥99,437 EGファントム	20店舗	115位	84位	5.00 (1票)	20件	09/10/22	09/10/08	Core 2 Duo P8700 2.53GHz(3MB)	4GB	500GB	Windows 7 Home Premium	14インチ
SONY VAIO Cシリーズ VPCCW18FJ/R	¥101,787 A-price	19店舗	120位	111位	5.00 (1票)	20件	09/10/22	09/10/08	Core 2 Duo P8700 2.53GHz(3MB)	4GB	500GB	Windows 7 Home Premium	14インチ

図 6.57: テーブルの分析

Model	Price	Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5	Attribute 6	Attribute 7	Attribute 8	Attribute 9
Aspire Timeline 4810T AS4810TG-A23	86100	21	79	90	0	09/11/27	09/11/27	2GB	320GB	Windows 7 Home Premium 14インチ
NI0Jb NI0JB-HV001U	45200	8	203	267	90	09/7/18	09/06/30	1GB	160GB	Windows XP Home 10.2インチ
Eee PC 1101HA-WP	79800	5	223	91	9	09/11/14	09/11/10	2GB	250GB	Windows 7 Home Premium 11.6インチ

図 6.58: 中括弧が指定されていない場合の結果レイアウトの例

まった結果レイアウトを生成してしまう。レイアウト変換の際も、属性間の関連性を特定できなくなり、変換コストの削減も困難である。これらの問題を解決するために、意味的に関連性が高い要素同士を中括弧でくくる処理を行う（3.2.4 項）。

初期の ACTIVIEW の問合せ文に中括弧を指定して要素間の関連付けを行った問合せ文を図 6.59 に示す。中括弧を導入することによって問合せ文に対し、提案したレイアウト変換戦略を適用することが可能になり、戦略の長所を生かすことが可能になった。

図 6.60 が中括弧の指定における提案戦略に従い一貫性のある結果レイアウトを生成した例である。図 6.58 と比べて、各属性集合の構造が統一的になっていることが分かる。この結果は、結果レイアウトの構造は開発者に定義に大きく依存することを示す。

図 6.59 の ACTIVIEW の問合せ文を基に、単純順序探索と、提案手法に従ったレイアウト変換によって生成される結果レイアウトの幅の種類と充填率を図 6.61 に示す。図 6.61 に示すように、6.3.1, 6.3.3 項の応用例の結果より候補レイアウトの数が多く、高い充填率を見せる良い結果を示している。

実際のサービスしている Web ページに対し、ACTIVIEW がどれくらい応用できるかの実験を行った結果を、表 6.3 にまとめた。従来の研究と比べて一貫性のあるレイアウトの生成が可能になり、様々な Web ページ型にも応用可能であることが分かる。実際の Web ページを対象とした実験による検討によると、複雑な構造をもつ Web ページに対して、ACTIVIEW の問合せ文による構造の再指定を行うことでレイアウト変換へ

```
GENERATE ACTIVIEW
[ items maker,
  [ items fig, items item%
    [ itemprice price, itemprice carriage, itemprice stock,
      itemprice shopname, itemprice area, itemprice update,
      itemprice link
    ]!,
    { priceinfo price, priceinfo dealer } ,
    { rankinginfo selling, rankinginfo intro,
      rankinginfo satisfy } ,
    { items board, items announce, items entry } ,
    { spec cpu, spec memory,
      spec hdd, spec os, spec lcd }
  ]! ]!
FROM ...
```

図 6.59: 中括弧の指定による要素間の関連付けを行った問合せ文

の応用が可能になっている。

特に ACTIVIEW によるレイアウト変換はテーブル構造によるデータを表現する Web ページに対して良い結果を示すことが分かる。さらに、開発者の意図的な意味を反映したレイアウト変換戦略の提案によって、ユーザに一貫性のあるレイアウトを提供することが可能になった。



図 6.60: 中括弧の指定における結果レイアウトの例

表 6.3: Web ページの構造化結果の比較

	関連研究	従来手法	提案手法
単純な構造をもつ Web ページ	△	△	○
複雑な構造をもつ Web ページ	△	△	△
データ中心のテーブルをもつ Web ページ	△	△	○
一貫性のあるレイアウト生成	×		

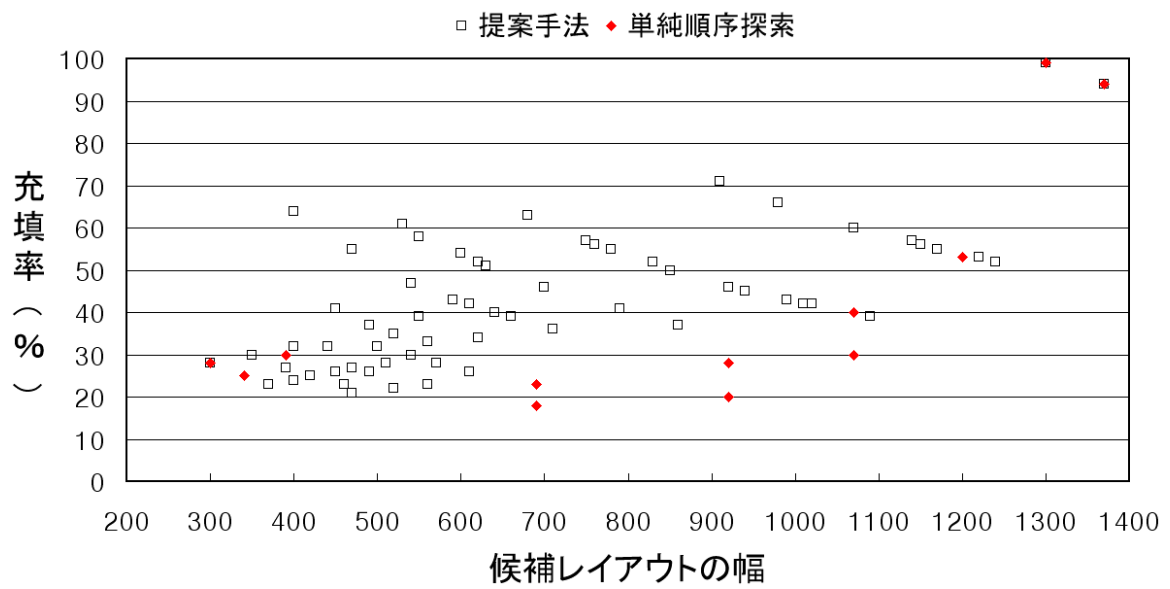


図 6.61: ACTIVIEW の問合せ文 (図 6.59) によって生成される候補レイアウトの幅の種類と充填率

第 7 章

関連研究

本章では関連研究と ACTIVIEW との相違点を調べて、今後の ACTIVIEW の開発方式について述べる。

本論文では、データベースからの検索結果を限られたサイズのユーザ表示画面のサイズに収まるように、オリジナルのレイアウト指定に近い形式、すなわち開発者の意図が崩れないレイアウトとユーザに見やすいレイアウトに自動変換するレイアウト変換戦略を提案した。ACTIVIEW は従来の研究とは異なり、レイアウト構造の定義という観点から研究を始めているため、他の研究と直接関係ある点は少ない。しかし、Web ページのコンテンツを様々な端末機に表示する様々な研究の特徴を検討することで将来の ACTIVIEW の機能拡張のために、参考になると考えられる。このため関連研究として、制限された表示領域における適切な情報提示手法を紹介する。

限られたユーザ表示画面のサイズに適したレイアウトへの変換については様々な手法が研究されている。特に、Web ページを対象にしているものでは、Web コンテンツの抽出、省略、要約によって小さい画面の端末機へ提供する手法と、特殊な Web ブラウザを提供する手法などが研究されている。他に、Baral らは SQL を拡張して、マルチメディアドキュメントをマルチメディアデータベースから生成する SQL + D という検索言語を提案している [35]。さらにこれによる動的プレゼンテーションの効率的な生成手法を提案している。まず、Web コンテンツを制限された表示領域で提示する手法として最も一般的な、Web コンテンツの抽出、省略、要約のアプローチについての代表的な研究を 2 つ述べる。

7.1 Web コンテンツの抽出，省略，要約

デスクトップコンピュータ用に生成された Web コンテンツをクライアント（ユーザ）の表示画面に合わせたコンテンツに変換するためには、もとの Web コンテンツに対しコンテンツの抽出、省略、要約をする必要がある。代表的なシステムとして、“Digestor システム”と“Power Browser システム”がある。他には、Web コンテンツを個人端末に表示する手法に関して、Bartlett[36]、Cessler ら [37]、Voelker ら [38]、Watson[39] たちが研究を行った。

7.1.1 Digestor システム

Bickmore らはデスクトップ用に作成された Web ベースのコンテンツを Palm-PC や PDA、携帯電話などの小さい表示画面を持つモバイル装置に合うフォーマットに自動変換するシステムである Digestor を提案した [18, 19]。Digestor システムは、与えら

れた表示画面に対して最適化された文書を提供するため、与えられた Web ページと表示画面のペアに対する最適な組合せの選択を可能とする技術として 6 つの自動的な re-authoring 手法を提案している。これはユーザに提供する Web ページのデザインを再構成する手段で、様々な実験による発見的計画アルゴリズムによる Web ページ変換手法である。

Outlining transforms：テクニカルペーパーやレポートのように各項目を持つ文書は各項目の内容をリンク先に移動して、項目だけ残すことによって表示画面サイズに合わせる。

First sentence elision transform：各テキストブロックは最初のページだけ残して、残りのページはリンク先のページに変換する。これにより表示に必要な画面の範囲が減少できる。

Indexed segment transform：まず Web ページが内容を基にソーティングされているページであるかソーティングされていないページであるか、内容的にあるいは構造的に連続しているページであるか（Previous や Next リンクを持っている）、ページ 1 つずつの段落になっているページ（この文からリンク先がない）であるかテーブルであるかなど、論理的に分割できるページの要素¹を探してユーザ表示画面のサイズが一杯になるまで適切に入れる。残る要素は“ Next ”と“ Previous ”ナビゲーションリンク先の文書に分ける。

Table transform：テーブルはそのままユーザに提供せず、各セルずつ上から下に、左から右の順に分割してページに変換する。

Image reduction and elision transforms：Digester システムで指定されているスケーリングファクター（25%、50%、75%）に基づき、画像の縮小を行う。

Image map transform：ユーザが縮小した画像が表示できないくらい小さい画面に対しては、画像は表示しない。しかし、画像がリンクを持っている場合は HTML の“ alt ”タグを利用し、リンク先のページをなくさないようにリンク情報を維持する。

Digester システムは、このように発見的手法によって、求められた優先順位に基づいて Outlining 以外にも、ユーザ画面に合わせてフォントの縮小処理を行うため、オリジナルのイメージを壊すことなく Web コンテンツをユーザに提供できる。ACTIVIEW

¹ページに含まれている段落など。ページを構成する HTML の分析によって指定される。たとえば、table や div タグなどによって内容が分けられると、これが 1 つの構成要素の基準になる。

と比べて、PDA や携帯電話など限られている対象に最適化されている。特に“ Indexed segment transform ”手法は SuperSQL システムで行う“ ORDER BY ”によるソーティングと似ている。ACTIVIEW でもリンク変換によってページ 1 つずつ生成しているが、“ previous や Next リンク ”の支援はまだしていない欠点がある。

“ Table transform ”処理によってテーブルを分割して、ユーザに Web の内容を提供しているが、上から下に、左から右の順に分けるだけでは複雑な型のテーブルに対応するのは困難である。これに比べ、ACTIVIEW は構造的な定義を基にしているのでテーブル型に依存しない。

7.1.2 Power Browser システム

Buyukkokten らは、小さい端末表示画面に Web ページの内容を提供するために、Web ページにある単語の重要度を用いて文章の要約を行いテキストのみでユーザに提示する Power Browser システムを提案した [20, 21, 22]。Power Browser システム (図 7.1) では、Stanford 大学の Digital Library プロジェクトの一部分で、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 という手法を用いている [23]。

Accordion Summarization では 3 つの手法を使っている。

Page Summarization : 元の Web ページを長方形の“ semantic textual units (STUs)”と呼ぶ単位に分ける。STUs は HTML タグ (br, table など) を基準にして分ける。各部分を 1 行に要約し、それらを木構造に表示する。1 行にした後の STUs の残りの部分はリンク先のページになる。図 7.1 の⑦は最初の 5 個の単語に要約されている (図 7.1 のⒶ)。

Keyword-Driven Summarization : PDA などの小さい画面から必要な内容を探すことは手間がかかる。そこで KWIC インデックス [40] を利用し、木構造のすべてのノードにキーワードを自動的に貼り付ける。この手法によって、ユーザが PDA を通じて内容を検索する際、入力した単語に対する検索時間を減らすことが出来る。検索された単語はハイライトされる (図 7.1 のⒷ)。

Automated View Transitions : 小さい画面でリンク先に動いたりすると元に戻りにくくなる。そこで、ユーザが小さい画面に適応できるように、急に画面を変えず、アップダウン動作をする“ Smooth scrolling ”とユーザにとって見やすいオートスクロール機能の“ Automated page scrolling ”、発見的な手法によるコン

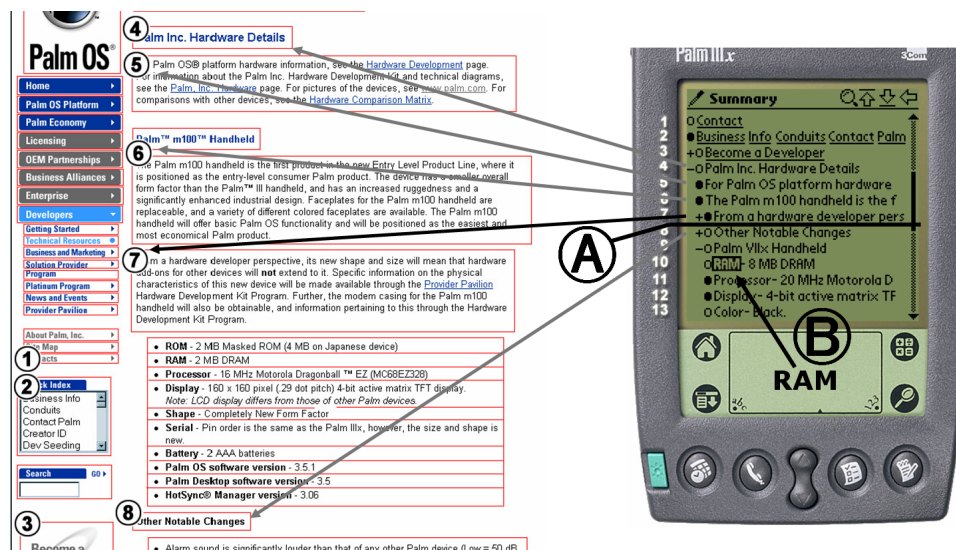


図 7.1: Power Browser システム

テンツの省略機能の“ Automated single-line reading ”の3つの手法を提案している。

Power Browser システムは自動的な Web ページの分析を行って結果を木構造に作成した後、PDA や携帯電話に提供する手法をもっているため、ACTIVIEW とは対象になる元のデータが異なる。特に、Digital Library プロジェクトの一部で行って研究であるので検索機能を重要な特徴としている。

“ Page Summarization ”手法は長い記事やテキストコンテンツなどを表示する際に ACTIVIEW でも利用すべきであると考えられる。今後、Digester システムの“ Indexed segment transform ”と“ Page Summarization ”手法を ACTIVIEW に適用する手法に関して研究を続けたいと考える。

7.2 特殊 Web ブラウザ

Web ページのコンテンツを特定の端末機に表示するために、特別なブラウザと、このブラウザに合わせて加工したコンテンツを提供するシステムもある。ACTIVIEW はブラウザ技術ではないが、これらの研究で用いている特別なブラウザに情報を提供する手法は参考になる。

Web ページを PDA などの小さい画面に表示するためのブラウザを提供する手法として、Björk らの WEST[24] と、Bruijn らの RSVP Browser[25, 26, 27, 28] などがある。

7.2.1 WEST

WEST (**W**EB browser for **S**mall **T**erminals) は , WAP (Wireless Application Protocol) [42] で利用する WML (Wireless markup Language , 7.3.2 項で詳しく述べる .) で作成されたページを表示するブラウザである . WAP は携帯電話などのデバイスでインターネット閲覧などのサービスが行えるようにする為の技術仕様である .

WML で作成されたページは“ decks ”と呼ばれて , decks は“ cards ”の集合で構成される . cards は他の cards やコンテンツへのリンクをもつオブジェクトである . PDA などの端末機から WML で作成されたページをアクセスすると , アクセス先のすべての cards が WAP サーバーからダウンロードされる . したがって , WML で作られたページを提供するためには , サービス (WML ページ) を提供するサーバー側と , この WML ページを表すブラウザが必要である .

WEST はサーバー側として“ proxy server ”とブラウザ側の“ client application ”の 2 つの部分で構成されている . Proxy server では , 次の 3 つの処理が行われる .

A chunking stage : 元の Web ページにあるテキストを WML に変換して decks ページとこのページに集められる cards に分ける .

A test reduction stage : テキストから繰り返し現れる単語を抽出してキーワードを生成する . このキーワードだけを表すことをコンテンツ省略と呼ぶ .

A link extraction stage : 各 cards に分けたページに対するすべてのリンクを抽出して生成されたキーワードに元のコンテンツへのリンクを付ける .

元の Web ページからサーバー側で生成した結果 , cards はクライアントに渡される際 , キーワードとリンクを保持している . クライアントアプリケーションは ,

Thumbnail view : ここでは , cards の focus+context 視覚化手法 [41] による (*flip zooming*) 構成のミニチュアビューが提供される . focus+context はユーザ端末機に表示される cards の中で , ユーザに選択した card を大きく見せる手法である .

Keyword view : cards のキーワードが表示される .

Link view : cards のキーワードからリンクされている内容が表示される .

の 3 つのディスプレイモードを提供する (図 7.2) ことで Web ページの内容を PDA に表示する .

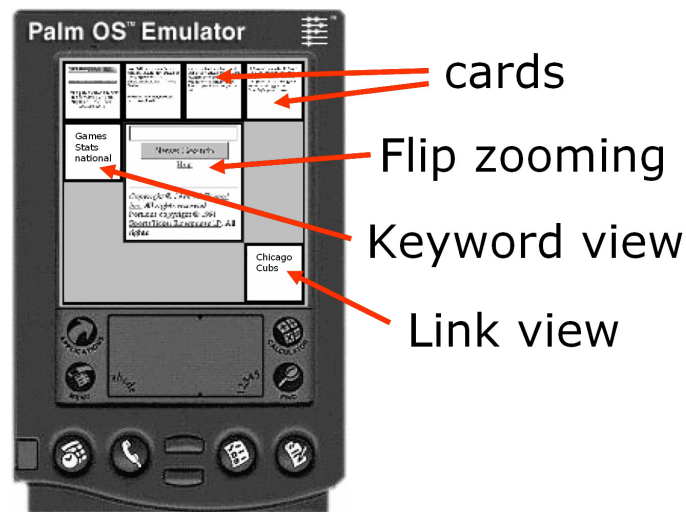


図 7.2: WEST ブラウザ

SuperSQL は現在 WML 文書を支援していない。しかし、WML 文書の支援を追加する際に ACTIVIEW を用いてデータベースからの検索結果をネスト別に cards に分ける手法で応用できると考える。この際の問題は、現在の ACTIVIEW がコンテンツの省略やキーワード抽出を行っていないため、データベースに格納されている情報からのキーワード抽出手法について研究するべきである。さらに、AJAX などを応用することで ACTIVIEW によって分けられたレイアウトをスライドで表示する手法を研究する価値があると思う。

7.2.2 RSVP

RSVP (Rapid Serial Visual Presentation) は、元の Web ページのコンテンツを省略して、小型化されたイメージと簡略化されたテキストでページを作成する手法と、この作成されたページを小さな画面の端末に提供する技術である。

RSVP は小さな画面の端末に Web コンテンツを提供するために、ブラウザ画面を 4 つに分けて (図 7.3) もとの Web コンテンツを表示する技術を提案している。

viewing area : Web ページのリンクとコンテンツをグラフィカルな表現手法で省略し表示する (図 7.3 の②)。

history bar : 図 7.3 の①。“backward” と “forward” ボタンを持つ (図 7.3 の⑤ と⑥) ものでユーザが見たページの情報を記録する。通常のブラウザの “back”

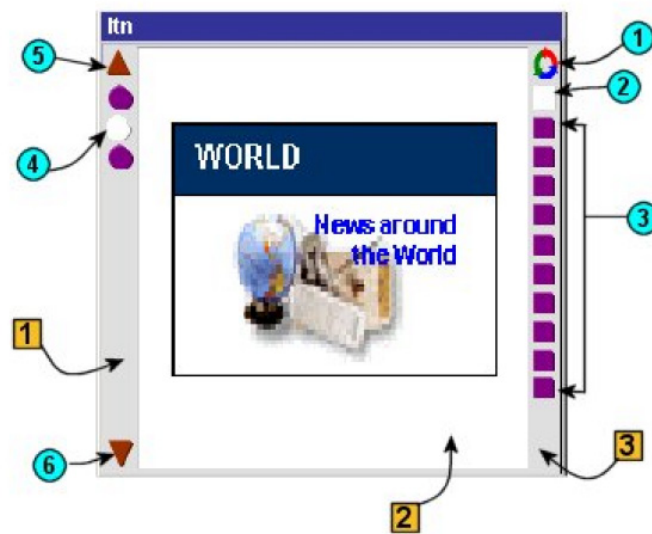


図 7.3: RSVP ブラウザ

と“ forward ”ボタンと同じ処理をする．小さい画面に表示できるコンテンツの量は限られているため，ユーザに提供したページの情報維持する必要がある．

RSVP bar：図 7.3 の③．現在のページからリンクされたページを自動的に連続して表示する機能を提供している．自動的に表示されるリンク先のページは内容が省略されたプレビューイメージで提供されるようになっている．このプレビューイメージの連続表示を始めたり止めたりするボタンと図 7.3 の①，プレビューイメージからページの内容を表示するボタン図 7.3 の②，表示されたページからのリンクされたページを表示するボタン図 7.3 の③を持つ．

title bar：Web ページのタイトルを表示する．

深いネストをもつレイアウトが ACTIVIEW によってリンク変換されて，リンクでつながっている多くのページを生成する場合，深いページまでたどったユーザは Web ページ内の現在の自分の位置を見失ってしまう恐れがある．この場合，RSVP ブラウザのように現在のページからリンクされているページのキーワード情報や全般的な Web ビューの構造に対する情報も同時に提供すると，ユーザにとっては全体的な Web ビューの構造が直感的理解できる．このように関係データベースからの検索結果から生成されるレイアウトだけではなく，レイアウトが変換される場合，このレイアウトとの関連情報，すなわち連結されているページが何の情報を持っているかに対するサマリー情報を HTML の“ alt ”タグを利用して提供するなどの手法に対する研究も将来の ACTIVIEW には必要であるう．

7.3 Markup Language

Markup Language とは、文書の構造や書体の変更、デザインなどの文書を表示するための情報をテキストファイルに盛り込むために使われる言語を意味する。文書はタグと呼ばれる文字列によって表現される。現在 Web ページを作成する言語の標準である HTML がマークアップ言語としては最も有名である。

開発者によって ACTIVIEW の問合せ文に定義される情報の数が多ければ多いほど、ACTIVIEW によってユーザ環境や元のデータの関連性を生かしたレイアウトを生成する可能性は高まる。このように構造的な定義以外に、元のデータの情報、たとえば検索結果の統計値、幅が狭くなる場合にはなくしても良い属性を指定できることで、ユーザ環境への最適化の可能性が高まると考える。さらに、携帯端末にレイアウトを提供する際はレイアウト結果を WML のような特定の環境に最適化された言語に変換して出力した場合がより良い結果を生成する可能性もありうる。それで、ACTIVIEW で適用可能であると考えた Markup Language について簡単に述べる。

7.3.1 XHTML

XHTML Basic[43] が、幅広い端末に対応したマークアップページを提供する目的で W3C によって勧告されて以来、端末へのアダプテーションに利用可能という観点から注目を集めている。

XHTML1.0(*Extensible Hyper Text Markup Language*)²[44] は構造記述言語とし発展してきた HTML に、XML の持つ柔軟性、拡張性を取り入れたものである。XHTML Family の次期バージョンとして XHTML2.0[45] が策定されていたが、W3C は 2009 年 07 月 03 日に策定の打ち切りを決定し、今後は HTML5 にリソースを注ぐものとした。理由として、XHTML2 市場は HTML 5 に比べて小さいことがあげられている。

HTML は、規格が拡張され、携帯端末などにとってはフルサポートするのが困難な規模となった。このため利用環境に応じて必要な要素のみが選択できるモジュール化や、他の XML 文書との融合などが行えるようにすることが、XHTML の重要な目的となった。

XHTML1.1 で、文書の構成要素を小さな単位に分けて定義し、これらを組み合わせることで多様なデバイスや環境に合わせた独自のタグセットを設計できるモジュール化という考え方を取り入れた。XHTML1.1 は、コンピューターなど大きな画面と高い処理能力を持つ端末向けで、小さな画面しか持たず処理能力の低い携帯端末では、フ

²2000 年 1 月 26 日に W3C の勧告となり、2002 年 8 月 1 日に改訂版である Second Edition が勧告された。

ルセットの HTML を処理するのは負担が大きい．このため XHTML Basic という別の標準が提案された．

W3C は、モジュール化された XHTML を用いて、携帯端末やテレビから通常の PC まで幅広い環境で利用できる文書の共通仕様を作成することを目指した XHTML Basic を 2000 年 12 月 19 日に勧告した．XHTML Basic は、XHTML モジュール仕様に基づき、定義済の共通モジュールを組み合わせで設計されている．特徴として、“Basic Tables”モジュールでは、テーブルを入れ子にすることはできないことや、イベント関連モジュールは、端末によって対応すべきイベントが異なるために含まれていないなどが挙げられる．すなわち、XHTML Basic は必要最小限のモジュールだけで構成されている．コンテンツ提供者は、XHTML Basic に従ったマークアップページを作成することで、幅広い種類の端末にひとつの記述で対応できることになる．

限られたユーザ環境に XHTML を利用することで、XHTML の機能を生かしたより多様なレイアウト型を提供できると考える．

7.3.2 WML (Wireless Markup Language)

WML[46] は WAP(Wireless Application Protocol)[42] 対応の無線機器を対象に、Web サイトのコンテンツが表示できるようにするために開発された HTML に似たマークアップ言語である．WML も、Web ブラウザを搭載した携帯電話などの小型携帯無線端末に Web サイトのコンテンツを提供することを可能とする言語である．

WML はインターネット上の情報を素早く検索、表示させるために、

- 携帯電話網とインターネット網の間にゲートウェイを置く形で実現される．
- 独自の通信プロトコルを使用しているため、HTTP、TCP などのインターネット標準のプロトコルは使用していない．
- HTML との互換性、画像の表示はサポートしていない．

と言う特徴がある．

簡単に言えば、WML は PC の Web ブラウザ上に表示される Web ページと同じコンテンツを配信するものではなく、HTML ページに含まれるテキスト以外の情報（画像やアニメーションなど）を除いたコンテンツを配信するものである．しかし、HTML ベースの Web サイトに WAP 対応の無線携帯端末からアクセスできるようにするには、各ページを WML で記述しなおした別ページを追加する必要がある．したがって、標準の HTML に書かれたページを WML で書かれたページに変換するアプリケーションの研究が行われている．

ACTIVIEW でも携帯端末のような極端に狭い幅をもつ端末機に対しては, ACTIVIEW によって生成された HTML 文書を WML 形式に変換することでより良い結果を得られる場合もありうる.

7.4 従来の研究との比較

7.1, 7.2, 7.3 節で紹介した研究では, Web コンテンツを縮小や要約したり, リンクを用いたりなど, 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, もとになる Web ページのレイアウト構造の変換は対象にしていない. 多様な端末にエンベッドされたアプリケーションからの Web アクセスも, 一般性のない特定の端末に限られている.

これらの研究と比べて ACTIVIEW は次のような特長がある.

1. 関係データベースの検索結果の表示を対象にする.
2. 元の ACTIVIEW の問合せ文によって定義される初期レイアウトからユーザの表示画面サイズに適したレイアウトへの変換を動的に行うことで, 特定の端末環境に限られず, ユーザ環境に合わせた多様なレイアウトの生成が可能である.

SuperSQL 処理系を利用するため, 本研究では一般的な Web コンテンツではなく, データベースからの検索結果を対象にする. これは ACTIVIEW が SuperSQL に従属することによる制約であるが, その代わり, SuperSQL の機能により, PDF などの HTML 以外の多様なメディアに出力することが可能になる.

今までの研究は, 携帯端末や PDA などの限定されたユーザ環境を対象にしている. しかし, ユーザの環境は変化するもので, この変化するユーザ環境に動的に適応できる技術を必要とする. 特に, 関係データベースの検索結果を多様な要求に従って Web ページを生成し, 提供することについては従来の研究ではなされてこなかった.

パソコンからある Web ページを見る際も, 人によってモニタの使い方が異なる場合も考えられる. たとえば, モニタの全画面で Web ページを見る人がいれば, 半分は Web ページを開き, もう半分は他の作業をする人もいる. また, 作業する途中で, Web ブラウザの幅を変換する場合もありうる.

ACTIVIEW は, レイアウトを生成する SuperSQL の問合せ文を書き換え, ユーザ環境に合わせたレイアウトを再構成することで, 変化するユーザ環境への動的な適応を可能とする.

亀岡ら [10] は SuperSQL におけるレイアウトの自動修正手法を提案している. この研究では生成対象を PDF として, 固定サイズの紙面に対して, レイアウトを自動変換す

る手法を提案している．その基本方法は ACTIVIEW と同様に SuperSQL の演算子の変更によるレイアウトの変換である．幅と長さには制約がある出版物に対応するため，反復子によって反復される出力を折り返す装飾子やラベル付けアルゴリズムを用いて長さ調整を行うことを提案している．しかし，様々なユーザ環境への対応や Web 上での動的な対応，PDF より縦方向の制約がゆるい HTML を対象とすることなど，ACTIVIEW は [10] とはその前提条件が大きく異なっている．

本章では関連研究およびブラウザ処理の中で ACTIVIEW に応用可能な技術に関して述べた．今後の ACTIVIEW の拡張に際してこれらの研究は適用可能であると考えられる．

第 8 章

結論

本論文では、データベースからの検索結果を Web ビューとしてユーザに提供する ACTIVIEW において、ユーザの端末環境に動的な適応を実現するレイアウト変換戦略を提案した。

ACTIVIEW は、関係データベースからの平坦な検索結果を構造化する SuperSQL の機能を用いて、Web 開発者が SuperSQL で定義した元の問合せ文から生成される初期レイアウトを、多様なユーザの端末の表示画面のサイズに合わせたレイアウトに変換する技術である。本論文ではより効率的なレイアウト変換を行うため、従来の ACTIVIEW システムを再設計し、また Lisp で開発されたシステムを Java ベースで再実装した。

従来の ACTIVIEW におけるレイアウト変換手法である単純順序探索と全数探索では、それぞれ候補レイアウトの生成の少なさと処理時間の長さの問題があった。これらを解決するため、レイアウトの評価基準を定め、より良いレイアウトへの変換の実現を実現するために制約と目標指標を設定した。具体的には、必ず満たすべき“幅制約”、“開発制約”の 2 つの制約と、生成されたレイアウトを評価する“幅占有率目標”、“充填率目標”、“長さ目標”の 3 つの評価関数である。

幅制約は、ACTIVIEW におけるレイアウト変換を行った結果レイアウトがユーザ表示画面の幅に収まるレイアウトであることを判定する。開発者制約は、開発者が元の問合せ文に定義する基本構造化情報である。開発者が定義した問合せ文に現れる属性の順序、入れ子構造、中括弧の指定をレイアウトの変換を行う際も維持することによって、開発者の意図的な意味を反映したレイアウト変換を行うようにした。具体的に、開発者の意図に基づく変換手法として 6 種類のレイアウト変換戦略を提案した。

本研究では統一性のあるレイアウトがユーザにとって見やすいと考えているため、6 種類のレイアウト変換戦略のうち、一貫性の高いレイアウトを提供することに高い優先順位を与えた。

候補レイアウトの選択基準として、幅占有率目標、充填率目標、長さ目標の 3 種類の評価関数を導入した。幅占有率目標はユーザ表示画面の幅の利用率を評価する。充填率目標は面積の利用率、長さ目標は画面に対するレイアウトの長さを評価する。さらに、候補レイアウトを生成するために行う連結子の変換回数、適用された変換戦略の優先順位を総合的に評価することで、開発者の意図的を保つ最終結果レイアウトの提供を可能にした。

最後に、本論文で提案したレイアウト変換手法と、単純順序探索によるレイアウト生成手法とのレイアウト生成時間、変換される結合子の個数、幅占有率、充填率の結果について評価実験によって比較し、本提案手法の有効性を示した。単純順序探索では充填率が 20%以下になるケースであるが、提案手法ではほぼ 60%以上を保つことができた。さらに、単純順序探索では結合子の変換とともに 26 個まで増加した変換回数

が、提案した変換戦略に従う統一変換を行うことによって結合子の変換回数をほぼ 5 回以下に収めることができた。

また、実際の商用 Web サイトをカテゴリ型、データ集中型、統計情報あるいは複雑なテーブルを持つ型の 3 つに分類し、ACTIVIEW の問合せ文を用いてこれらを擬似的に生成する応用の実験の例を示した。この結果、特に統計情報あるいは複雑なテーブルをもつサイトの再構造化に対しては良い結果を示すことが分かった。具体的には、7 段階のカテゴリをもつ Yahoo サイトを生成する実験を行った結果、単純順序探索が 9 種類の候補レイアウトを生成したのに比べて、本論文の提案手法は 128 種類の候補レイアウトを生成しており、両方とも充填率は低かったが、単純順序探索がほぼ 15% 以下となるのに比べて 15% 以上の充填率を提供する候補レイアウトも数多く提供することができた。複雑なテーブルを持つサイトへの応用例では、40% を下回る充填率を示した単純順序探索に比べ、提案手法で生成された 56% 以上の候補レイアウトが 40% 以上の充填率を示した。最終結果レイアウトを生成する時間も単純順序探索の約 0.26 秒に比べて、提案手法の約 0.48 秒は数値的には約 2 倍であるが、全数探索が 15 個以上の変換対象の場合で 10 秒以上を要するのと比べて、ユーザに十分動的な対応が可能になったと主張できる。

本論文で再設計を行った ACTIVIEW システムとそのレイアウト変換手法は、現在のモバイル端末機の発展による表示画面の急速な変化や、ノートパソコン画面サイズの多様化に対応する技術として応用が期待される。また、ACTIVIEW による Web ビューによって既存の情報提供中心の Web サイトにおいて、追加の開発コストをかけずに多様なユーザ環境からの効率的な閲覧が実現できる。

今後、構造化情報が定義されていない問合せ文、すなわち通常の SQL で書かれた問合せ文に対して、適切なレイアウト構造を自動的に提供する手法と、ACTIVIEW が提供するレイアウトの認知的な理解度を数値的に評価できる評価関数に関して研究を続けたいと考えている。

謝 辞

本研究の機会を与えて下さり，また研究を進めるにあたり御多忙にも関わらず，終始，有益なご助言と御指導を頂いた慶應義塾大学理工学部准教授 遠山元道 先生に心より感謝致します．レイアウト変換戦略に対する遠山先生のアドバイスがなければ，また，探索手法に対する色々な御指導がなければ，本論文の完成には至らなかったことと思います．本当に，長い間，温かく見守って頂き，様々な経験を通して，研究することの楽しさと厳しさを教えて頂きました．

慶應義塾大学理工学部教授 山本喜一 先生には，ご健康が万全でない状態の御身体をおして，本論文の審査を御快諾頂き，また論文に関して多くの貴重な御意見を頂きました．さらに博士論文の手続きを進める上で，的確なご助言まで頂き，大変お世話になりました．慶應義塾大学理工学部教授 天野英晴 先生には，お忙しい中，本論文の審査をお引き受け頂き，大変有益なコメント，論文の修正，御指導を頂きました．感謝致します．慶應義塾大学理工学部教授 櫻井彰人 先生には，本論文の審査を御快諾頂き，研究内容のみならず，研究の進め方や論文の作成について，機会あるごとにご助言を頂き，参考にさせて頂きました．大変感謝しております．

そして，本研究の元になる，ACTIVIEW を初めて提案した前田葉子さんと，韓国から留学に来て，研究生としての在籍時から遠山研究室にお世話になり，はじめは戸惑うことも多かった私を様々な面で支えてくれた先輩方，同期，後輩の皆様に感謝致します．また，ご多忙な中，何度も論文の修正を頂いた有園チエさん，様々な面で支えて下さった山田浩之氏に感謝致します．

日本での長い留学生活にあたり，常に励ましながら支えて下さった李泰金氏，孫用順さんご夫婦，李賢吉氏，岩崎節雄氏，岩崎里子さんご夫婦，南部陽介氏，山田研氏，川島麻子さん，坂田明美さん，下平淳吾氏，前田歩美さんに感謝します．

最後に，長きにわたり勉強と研究の機会を与えてくれるとともに，健康面，精神面において，常に支えてくれた韓国の両家の両親及び祖母，妹と弟に心から感謝致します．そして長い留学生活，離れて生活する苦労を共にし，励ましてくれた妻の申京任に深く感謝し，この論文を捧げます．

2010年2月
慎 祥 揆

参考文献

- [1] Adomaviciu, G. and Tuzhilin, A. Personalization technologies: A process-oriented perspective, *Comm. ACM*, Vol.48, No.10, pp.83-90 (2005).
- [2] Abiteboul, S., Cluet, S. and Milo, T. Active Views for Electronic Commerce, *Proc. VLDB*, pp.138-149 (1999).
- [3] Ceri, S. and Franterali, P. Data-Driven, One-To-One Web Site Generation for Data-Intensive Applications, *25th VLDB*, pp.615-626 (1999).
- [4] 清光英成, 竹内淳記, 田中克己. ActiveWeb: アクティブルールのXML表現に基づく Web コンテンツの個別化とアクセス管理, *情報処理学会論文誌*, Vol.42, No.SIG8 (TOD10), pp.139-147 (2001).
- [5] 平林真美, 大月一弘, 清光英成, 森下淳也, 北村新三, 絹川達也. リンクに基づいた明示的 Web ページ評価法, *情報処理学会論文誌: データベース*, Vol.43, No.SIG12 (TOD16), pp.92-102 (2002).
- [6] 竹内 準, 清光英成, 田中克己. アクセス履歴の集約とメタルールに基づく Web コンテンツのアクセス管理機構, *情報処学会データベースシステム研究会報告*, Vol.2000, No.96, pp.315-322 (2000).
- [7] Toyama, M. SuperSQL: An Extended SQL for Database Publishing and Presentation, *Proc. ACM SIGMOD*, pp.584-586 (1998).
- [8] Shin, S.G., Arisawa, T. and Toyama, M. The Integration of Media Generators in SuperSQL Query Processor, *3rd International Conference on ELPIIT2003*, pp72-76, KIPS&KMMS (2003).
- [9] Seto, T., Nagafuji, T. and Toyama, M. Generating HTML Sources with TFE Enhanced SQL, *ACM Symposium on Applied Computing*, pp.96-100 (1997).
- [10] 亀岡慎平, 遠山元道. SuperSQL を用いた組版におけるレイアウト自動修正, *情報処理学会論文誌: データベース*, Vol.46, No.SIG13 (TOD27), pp.78-93 (2005).

- [11] Maeda, Y. and Toyama, M. ACTIVIEW: Adaptive data presentation using SuperSQL, *27th VLDB*, pp.695-696 (2001).
- [12] 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, 5B-7 (2001).
- [13] 前田葉子, 遠山元道. ACTIVIEW における適応型表提示の最適化, 電子情報通信学会データ工学ワークショップ, A2-6 (2002).
- [14] 前田葉子. SuperSQL による適応型表示ビューの生成, 学部卒業論文, 慶應義塾大学 (2001).
- [15] 前田葉子. ACTIVIEW における適応型表提示の最適化, 修士学位論文, 慶應義塾大学 (2003).
- [16] Shin, S.G., Maeda, Y. and Toyama, M. ACTIVIEW: Implementation of Adaptive Web View Using SuperSQL, *The 6th International Conference on Informatics and Systems*, Cairo, IEEE (2008).
- [17] EBNF. ISO/IEC 14977: 1996(E) The international standard for Syntactic Meta-language - Extended BNF.
- [18] Bickmore, T. and Schilit, W. Digestor: Device-independent Access to the World Wide Web, *Proc. Computer networks and ISDN Systems*, 29(8-13), pp.1075-1082 (1997).
- [19] 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).
- [20] Buyukkokten, O., Garcia-Molina, H., Paepcke, A. and Winograd, T. Efficient Web Browsing for PDAs, *Proc. CHI'2000*, pp.430-437 (2000).
- [21] Buyukkokten, O., Garcia-Molina H., and Paepcke, A. A Focused Web Searching with PDAs, *Proc. WWW9*, pp.213-230 (2000).
- [22] 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).

- [23] 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).
- [24] Björk, S., Holquist, L-R., Redstrom, J. Bretan, I., Danielsson, R., Karlgren, J. and Franzen, K. WEST: A Web Browser for Small Terminals, *ACM Symposium on UIST'99*, pp.187-196 (1999).
- [25] Bruijn, D.O. and Spence, R. Rapid serial visual presentation: A space-time trade-off in information presentation, *Proc. AVI2000*, pp.189-192 (2000).
- [26] Bruijn, D.O. and Spence, R. Movement in the Web, *Extended Abstracts CHI2001*, pp.209-210 (2001).
- [27] Spence, R. and Bruijn, D.O. The Usefulness and Usability of Space-time Trade-offs for information Navigation on Small Display, EPSRC grant GR/N20881.
- [28] Bruijn, D.O., Spence, R. and Chong, M.Y. RSVP Browser: Web Browsing on Small Screen Devices, *Personal and Ubiquitous Computing*, Vol.6, Issue 4, pp.245-252 (2002).
- [29] Silberschatz, A., Korth, H. F., and Sudarshan, S. *Database system concepts*, 4th edition, MC Graw Hill, 2001.
- [30] FIFA World Cup. <http://www.fifa.com/worldcup/index.html>.
- [31] Alexa. <http://www.alexa.com>.
- [32] Yahoo Japan. <http://www.yahoo.co.jp>.
- [33] 小林武彦, 遠山元道. SuperSQL を用いた RIA 作成支援の提案, 電子情報通信学会 データ工学ワークショップ, pp.303-308 (2007).
- [34] 価格.com. <http://kakaku.com>.
- [35] Baral, C., Gonzalez, G. and Nandigam, A. SQL+D: Extended Display Capabilities for Multimedia Database Queries, *Proc. 6th ACM International Conference on Multimedia 1998*, pp.109-114 (1998).
- [36] Bartlett, J. Experience with a Wireless World Wide Web Client, Technical Note TN-46, Digital Western Research Laboratory (1995).

-
- [37] Gessler, S. and Kotulla, A. PDAs as mobile WWW browsers, *Comput. Networks ISDN Syst.*, Vol.28, Issues 1-2, pp.53-59 (1995).
- [38] Voelker, G. and Bershad, B. Mobisaic: an information system for a mobile wireless computing environment, *Workshop on Mobile computing Systems and Applications*, pp.185-190, IEEE Computer Society Press (1994).
- [39] Watson, T. Application design for wireless computing, *Workshop on Mobile Computing Systems and Applications*, pp.91-94, IEEE Computer Society Press (1994).
- [40] Luhn, H.P. Keyword-In-Context Index for Technical Literature (KWIC Index), *American Documentation XI*, 4, pp.288-295, (1960).
- [41] Björk, S. and Holquist, L.E. Formative Evaluation of a Focus+Context Visualization Technique, *Proc. HCI'98*(poster presentation), The British HCI Society (1998).
- [42] Wireless Application Protocol. <http://www.wapforum.org>.
- [43] XHTML Basic 1.1. <http://www.w3.org/TR/xhtml-basic>.
- [44] XHTML1.0 The Extensible HyperText Markup Language. <http://www.w3.org/TR/xhtml1>.
- [45] XHTML 2.0. <http://www.w3.org/TR/xhtml2>.
- [46] Website Meta Language. <http://thewml.org>.
-