SuperSQL におけるクロス表生成機能の実装

田畑　篤智†　　五嶋　研人†　　遠山元道††

† 慶應義塾大学理工学部情報工学科　〒 223–8522 神奈川県横浜市港北区日吉 3–14–1
E-mail: †{tabata,goto}@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし　統計表や製品の諸元表など, 多くの出版物で広く用いられているクロス表は一種のデータモデルとみなすことができる. クロス表とは表頭および表側に属性があり, その属性が一致するところに値が入る, という構造になっている表形式のことである. クロス表は関係形式の表とは異なり二つの値に対して合致する値が表の中に値として入るのである. 本論文ではクロス表の本質的な構造を考えその構造を形式的に表現し, 関係データモデルからクロス表への変換の一般的な定義を行った. そしてその定義を用いてデータベースから取得したデータを短いクエリで様々な種類の構造化文書にすることができる SuperSQL の一つの機能として組み込んだ. しかし SuperSQL での出力の仕様の関係もありこのクロス表の形でデータを出力するためには, 構造の把握が容易ではない長いクエリを書かないといけなくなってしまう. そこで本論文では, SuperSQL 上でクロス集計表を作成できる関数を実装したことを紹介する. このことにより SuperSQL で簡単に作成できる表の形式が増え表現力が向上された.

キーワード　SuperSQL, Cross table, SQL

1. はじめに

統計表や製品の諸元表など, 多くの出版物で広く用いられているクロス表は一種のデータモデルとみなすことができる. これは関係形式の表現とは異なり表頭及び表側属性という 2 つの属性に対して合致する値を表の値として持つデータモデルである. 実際に表として表すと図 1 のようになっている. また図 1 のような場合から図 2 のようにデータベースの値の平均値や合計を集約して表現する複雑な場合がある. 4.1 節で後述するが本論文ではクロス表の構造を考えそれを形式的に表現した. またその形式を利用して関係データモデルで定義されているデータをクロス表の構造に変換する方法を考え実装した. その実装したソフトウェアが SuperSQL である.

SuperSQL とは, データベースの値を様々な構造で出力することができる SQL の拡張言語である. 通常の SQL では, 最もシンプルでフラットな表しか作成することができないが.SuperSQL を用いれば様々な構造の表を簡単に作成することができる. また, HTML, PDF, PHP など様々なメディアにも対応しておりそれぞれの言語で書くよりもはるかに少ないコード量で生成できるのも特長のひとつである.

HTML などでそのまま実現させる方法も考えられるが表側の属性値と実際の値を同時に入力することになる, 多段階の集約になるとセルの結合なども考えないといけない, というような問題点が考えられるので SuperSQL 上で実現するのが最も良い手段であると考えられる. しかしながら現在の SuperSQL でクロス表の構造を宣言しようとするとクエリが複雑になり尚且つ直感的に構造を宣言することができないため生成が容易ではない.

そこで本論文では, クロス表の構造を考え SuperSQL にクロス表を作成できる機能の実装することを目的としている. 実装したクロス表を作成する機能は表頭, 表側, 値それぞれに様々なレイアウトを実現する TFE 文を用いることが可能なのでユーザーが望む複雑な構造を持ったクロス表も容易に実現することが可能となっている. また表頭や表側, 値には平均値や合計も入れることができるのでクロス集計表としての機能もある.

	男性	女性
購入する	30人	43人
購入しない	21人	8人

図 1　クロス表

		月曜日		火曜日	
		数学	英語	物理	化学
		68点	55点	80点	48点
A組	安西	66点	49点	89点	65点
	飯山	54点	61点	75点	46点
B組	宇野	91点	89点	90点	59点
	木村	53点	48点	68点	40点
	武内	87点	50点	80点	46点
C組	中瀬	66点	56点	66点	33点
	西村	59点	32点	92点	47点

図 2　集約などを用いた複雑なクロス表

以下本論文の構成を示す. 第 2 章では関連研究関連技術について述べ, 第 3 章では SuperSQL についての説明, 第 3 章ではクロス表生成機能について説明する. 第 4 章で実験評価を述べ, 第 5 章で結論を述べる.

2. 関連研究, 関連技術

Microsoft SQL server [4] という SQL server には PIVOT 句がある. この句は行を列に変換することができ, クロス表の実現することが可能となる. しかし値として用いることのできる集約関数が一つしか使えない, 複雑な構造を指定することができないという欠点がある.

また Microsoft Office Excel [5] に於いてもピボットテーブルという機能がある.SQL クエリで書くのではなく直感的な操作でクロス集計表を作成することが可能となる. クエリを書くわけではないので SQL を知らないユーザでも作りやすいが, 複雑

な構造をあらわすことができない.

Gray らの研究 [6] ではクロス表を実現することのできる cross–tabulation という多次元での集約方法を考えそれを応用することによって CUBE, ROLL–UP というオペレーターを考案している.

Johnson らの研究 [7] では病院データを多次元集約するために本来行として出力されるテーブルに格納されているデータを列に変換する pivoting という考えを用いて SQL の拡張を提案している.

3. SuperSQL とは

この章では，SuperSQL について簡単に述べる．SuperSQL は関係データベースの出力結果を構造化し，多様なレイアウト表現を可能とする SQL の拡張言語であり，慶應義塾大学遠山研究室で開発されている [1] [2]．そのクエリは SQL の SELECT 句を GENERATE <*media*> <*TFE*> の構文を持つ GENERATE 句で置き換えたものである．ここで <*media*> は出力媒体を示し，HTML，PDF などの指定ができる．また <*TFE*> はターゲットリストの拡張である Target Form Expression を表し，結合子，反復子などのレイアウト指定演算子を持つ一種の式である．

3.1 結 合 子

結合子はデータベースから得られたデータをどの方向 (次元) に結合するかを指定する演算子であり，以下の 3 種類がある．括弧内はクエリ中の演算子を示している．

- 水平結合子 (,)

データを横に結合して出力する．

例：name, place

name	place

- 垂直結合子 (!)

データを縦に結合して出力する．

例：name! place

name
place

3.2 反 復 子

反復子は指定する方向に，データベースの値があるだけ繰り返して表示する．また反復子はただ構造を指定するだけではなく，そのネストの関係によって属性間の関連を指定できる．例えば

[部署名]! , [雇用者名]! , [給料]!

とした場合には各属性間に関係はなく，単に各々の一覧が表示されるだけである．一方，ネストを利用して

[部署名! [雇用者名, 給料]!]!

とした場合には，その部署毎に雇用者名と給料の一覧が表示されるといったように，属性間の関連が指定される．以下，その種類について述べる．

- 水平反復子 ([ ],)

データインスタンスがある限り，その属性のデータを横に繰り返し表示する．

例：[Name],

name1	name2	···	name10

- 垂直反復子 ([ ]!)

データインスタンスがある限り，その属性のデータを縦に繰り返し表示する．

例：[Name]!

name1
name2
···
name10

3.3 順 序 指 定

SuperSQL では関係データベースから取得したデータの表示順序を昇順又は降順に表示するように指定できる. 昇順の場合は昇順にしたい属性に ‘(asc)’ をつけ降順の場合は ‘(desc)’ をつけることになる.

(*asc*)*or*(*desc*)< 属性名 >

3.4 装 飾 子

SuperSQL では関係データベースより抽出された情報に，文字サイズ，文字スタイル，横幅，文字色，背景，高さ，位置などの情報を付加できる．これらは装飾演算子 (@) によって指定する.

< 属性名 >@{< 装飾指定 >}

装飾指定は “装飾子の名称 = その内容” として指定する．複数指定するときは各々を “,” で区切る.

[name@{width=100, color=red}]!

3.5 関 数

SuperSQL では専用の関数が多数存在する. 宣言の形式は一般的な関数と同じで

SuperSQL 関数の宣言形式

関数名 (引数 1, 引数 2, ...)

となっている. 以下で SuperSQL 関数を幾つか紹介する.

3.5.1 image 関数

image 関数を用いると画像を表示することが可能となる．引数には属性名，画像ファイルの存在するディレクトリにパスを指定する.

image(pict, ‘./pic’)

3.5.2 link 関数 (出力メディアが HTML の場合のみ)

link 関数は，FOREACH 句と同時に用いる．これらを用いることで深度結合子と同様にリンクを生成することができる.

link(name, ‘./menu.ssql’, place)

図 3　クロス表生成機能の仕様

4.　クロス表生成機能

この章ではクロス表生成機能の仕様, クロス表生成機能に必要だった機能, 実装の際のアルゴリズム, 用例を説明する.

4.1　クロス表生成機能の仕様

クロス表の構造を考えると表頭, 表側に属性があり表の値としてはその二つの値に合致するものが当てはまるという構造になっていると考えられる. そして SuperSQL には TFE という構造を表現する表現形式がある. この TFE を表頭, 表側, 値に用いることでクロス表を表現が可能となる. 以上のように考えてクロス表生成機能を SuperSQL の関数として実装した際の宣言文は以下のようになった.

クロス表生成関数の宣言

cross_tab(*TFE1*,　*TFE2*,　*TFE3*)@{side-width=*num*, null-value='*str*'}

上記の宣言文の TFE と装飾子と表の対応は図 3 の様になっている. 以下で表の各属性の構造指定, 表の幅指定, 該当する値がない時の処理について説明するが引数と表の値との対応関係は図 3 に示した通りとなる.

4.2　表の属性の構造変換

4.1 節にあるように cross_tab 関数は TFE を三つ引数としてとることによってクロス表を生成する. 本節ではそれぞれの部分について説明をする. 図 3 にもある様に第 1 引数である *TFE1* は表頭属性の構造を指定し第 2 引数である *TFE2* は表側属性の構造を指定する. 第 3 引数である *TFE3* はクロス表に入る値を指定する. この時 *TFE3* でも複数の値を指定することができ, 構造の指定も可能である. 構造変換部分では最初は関数形式になっている TFE をクロス表となる TFE に変換することを行っている.

4.3　属性の順序指定

SuperSQL では属性値の表示順序を指定しないと図 4 のように表頭と表側の属性と一致しないランダムで出力されてしまう. それを防ぐために以下の様にルールを決めた.

（1）　ユーザが指定した順序を最優先

（2）　指定がない場合は昇順で表示する

この様にルールを決めた結果, 図 5 のように表頭および表側属性と値が対応した表が生成された. ユーザが順序を宣言する方法は 3.3 節にあるように属性の前に目的に応じて 'asc' 又は 'desc' をつけることである. 以下でその例およびルールを適用した結果を示す.

		tue	mon	
		Webapp	DM	DBS
takaba		C	B	A
		DBS	DM	Webapp
		mon	mon	tue
kitahara		B	A	A
		DBS	DM	Webapp
		mon	mon	tue
matsuda		B	B	A
		DM	DBS	Webapp
		mon	mon	tue

図 4　順序指定しなかった場合のクロス表

		mon		tue
		DBS	DM	Webapp
tanaka		C	A	A
		DBS	DM	Webapp
		mon	mon	tue
takei		B	B	C
		DBS	DM	Webapp
		mon	mon	tue
tabata		A	B	C
		DBS	DM	Webapp
		mon	mon	tue
suzuki		A	B	C
		DBS	DM	Webapp
		mon	mon	tue

13

図 5　順序指定した場合のクロス表

ユーザが一部を順序指定をした場合のクエリ例

```
cross_tab([c.day![c.name],], ,
[p.performance]! ,
[(desc)s.id]!)
```

この例では s.id という属性のみ desc が指定されているのでこの属性は降順に表示される. その他の属性は何も指定がないので昇順に表示となる.

4.4　該当する値がない時の処理

例えば図 6 の様にある学校の各授業の年齢および性別に対しての A, B, C の人数を表示するクロス表を作成した時該当する人がいない可能性がある. このような場合 4.1 節のクロス表生成関数の宣言にあるように 'null-value' を指定することによって任意の文字列を入れることが可能となる. 図 6 は該当する値がない場合にそのセルには 'N/A' を入れる, とした場合となる.

4.5　セルの幅指定

まず各 *TFE* は装飾子をつけられるので手動で幅指定ができる. しかしそれだとクエリが長くなってしまうので何も指定がなかったらそれぞれのセルの横幅は 100px となる. また 4.1 節のクロス表生成関数の宣言にあるようにクロス表を作る関数では 'side-width' という値を指定することができる. この値はクロス表を作ると必ずできる左上の空白の横幅を指定するもので

		female			male		
		13	14	15	13	14	15
DBS	A	1	1	N/A	4	1	1
	B	2	3	4	N/A	1	N/A
	C	N/A	1	1	1	2	1
DM	A	1	4	1	2	2	1
	B	2	N/A	3	2	N/A	N/A
	C	N/A	1	1	1	2	1
Webapp	A	N/A	N/A	4	1	N/A	1
	B	N/A	3	N/A	1	3	1
	C	3	2	1	3	1	N/A

図 6　該当するセルに値がない場合の例

		female			male		
		13	14	15	13	14	15
DBS	A	1	1	N/A	4	1	1
	B	2	3	4	N/A	1	N/A
	C	N/A	1	1	1	2	1
DM	A	1	4	1	2	2	1
	B	2	N/A	3	2	N/A	N/A
	C	N/A	1	1	1	2	1
Webapp	A	N/A	N/A	4	1	N/A	1
	B	N/A	3	N/A	1	3	1
	C	3	2	1	3	1	N/A

図 7　幅指定をしなかった場合のクロス表

ある. これを指定しないと図 7 の様に左上の空白の横幅は 10px となる. 未指定の場合 10px にした理由としては, 左上に枠を入れられることを忘れている場合にそこに枠が入ることがわかりやすくするするためである.

4.6　実装の際のアルゴリズム

クロス表生成機能の概要を説明したのち, ‘表の構造の変換 (4.6.2 節)’, ‘属性の順序指定 (4.6.3 節)’, ‘該当する値がない時の処理 (4.6.4 節)’ のアルゴリズムについて説明していく.

4.6.1　クロス表生成機能の概要

SuperSQL には SSQL クエリを受け取り SQL クエリと表の構造 (TFEtree) に分ける Parser, SQL クエリによる検索結果と表の構造情報を受け取りデータの整形をする Data Constructor, 整形されたデータを受け取り実際にファイルとして出力する Code Generator の三つの機構がある. 以下の図 8 にあるようにクロス表を生成するために本論文では ‘表の構造の変換’ と ‘属性の順序指定’ をパーザーとコード生成部の間で行い, 以降の章で説明する該当する値がないときの処理をデータコンストラクタで行っている.

4.6.2　表の構造の変換

SuperSQL では表の構造を木構造で保持している. 例えば横結合は横結合のトークンを親に結合する要素を子として持つ木となっている. 変換する前は 4.1 節で説明した宣言文の構造が木として保持されている. それをクロス表となる形に変換する. 以下にクロス表となる構造を提示する.

クロス表になる TFE

””, [TFE1], ! [TFE2 , [null(TFE1) , TFE3]!]!

4.1 節で紹介した宣言文を上記の構造となる様に変化させれば結果的にクロス表が生成できる. この変換のアルゴリズムは疑似

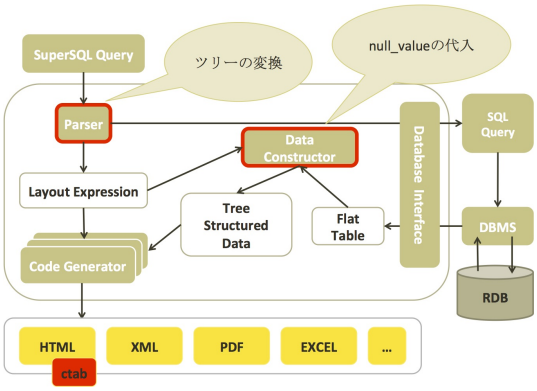

図 8　クロス表生成機能のアーキテクチャ

コードで表すと以下の様になる. 以下で使用している NULL 関数とは表示させたくないがグルーピングなどで使いたい属性につける関数である. 後半に使う TFE1 は表示させる必要がないので NULL 関数に入れる. 10, 11 行目についてだがここで前述

Algorithm 1 TFE の変換

Input: cross_tab 関数の TFE
Output: クロス表の形になった TFE

```
1: cross_tab 関数の引数として TFE1 から TFE3 を抽出
2: while TFE1 の要素がある間 do
3:     if 属性名を示すトークンがあったら then
4:         属性名を保存する
5:     end if
6: end while
7: while TFE1 の属性がある間 do
8:     NULL 関数に入れる
9: end while
10: NULL 関数に入った TFE1 に TFE3 を統合し, それを TFE2 に
    統合する
11: 表示する TFE1 と結合する
12: return TFE
```

のクロス表となる構造を作成している. まず 10 行目の ‘NULL 関数に入った TFE1 に TFE3 を統合’ というのは TFE1 の横結合に TFE3 を入れるということである. また ‘TFE2 に結合’ というのは TFE2 のグルーピング以下に前述のものを追加するということである. ここまでで以下の構造が出来ている.

ここまでで完成した TFE

[TFE2 , [null(TFE1) , TFE3]!]!

11 行目ではクロス表の左上の枠と TFE1 に 10 行目で生成したものを結合している. その結果以下の構造が出来上がる.

最終的に完成した TFE

””, [TFE1], ! [TFE2 , [null(TFE1) , TFE3]!]!

4.6.3 属性の順序指定

4.3 節で説明した様に SuperSQL では値の表示順序を昇順又は降順で指定しないとランダムに表示されてしまう. すると表頭および表側属性と値が一致しなくなってしまうことがある. 4.3 節にあるルールを実現するための処理を行ったアルゴリズムの疑似コードは以下の様になる. 1 行目の tfe_anlysis 関数で

Algorithm 2 属性の順序指定

Input: TFE 文
Output: 順序指定を追加した TFE

```
1: FUNCTION tfe_anlysis(TFE)
2: if 属性を示すトークンを発見 then
3:     if 属性を示すトークンの前に順序指定が無い then
4:         return 順序指定を追加した TFE
5:     else
6:         順序指定の情報を保持
7:         return TFE
8:     end if
9: else
10:    TFE の一つ下の要素を参照し tmp に保存
11:    return tfe_anlysis(tmp)
12: end if
```

受け渡された TFE 文を解析し 2, 3 行目で順序指定されているかを確認している. この部分で 4.3 節のルールを実現していることになる. また SuperSQL では TFE 文は以下の様なリスト構造になっている.

リスト構造の例

[縦結合を示すトークン, [横結合を示すトークン, [[属性を示すトークン], [属性を示すトークン]]]]

これを深く潜っていくことで TFE 文を解析しており, この解析自体は 12 行目で行なわれている.

4.6.4 該当する値がない時の処理

4.4 節で説明した様に集約関数を用いた際該当する値が存在しない場合がある. その時はユーザーが指定した文字列をその値として代入することになっている. 以下にその処理を行ったアルゴリズムの疑似コードを示す. 1 行目にある様にまず表頭および表側の属性値として実際に出てきているものを保持する. 図 9 の様な表の場合保持される属性は表頭属性の場合 [[female, male], [13, 14, 15]] で表側属性の場合 [[DBS, DM, Webapp], [A, B, C]] となる. この処理を 2 行目から 8 行目で行っている. 次に 9 行目にある様に 2 行目から 8 行目で作成したリストを用いてそのありうる組み合わせをすべて作る. 前述の例の場合出来上がるタプル数は 54 個となる. この処理自体は 10 行目にある様に 26 行目以下の combine 関数で行っている. この関数では再帰的に関数を呼び出し結合した結果にさらに次の属性値を結合していくことで全通りを作成している. しかし 11 行目にあるようにクエリを実際に見てみると属性名が一緒のものも存在する. たとえば表頭属性と表側属性どちらもで評定 (A, B, C) を指定していた場合これは検索結果としては常に一致していな

Algorithm 3 該当する値がない時の処理

Input: SQL クエリの検索結果に集約処理を適用したタプル群
Output: 入力タプル群の内の不足分 (値を N/A とした) を追加したタプル群

```
1: //各属性のありうる値を取得する
2: for all 入力タプル群 do
3:     for all 各入力タプルの要素 do
4:         if 属性の保存リストに同じものがなかったら then
5:             保存リストに追加する
6:         end if
7:     end for
8: end for
9: //値の保存リストから全通りの組み合わせをつくる
10: combine(1, 保存リストの 1 番目のリスト, 保存リストの 2 番目のリスト, 保存リストのサイズ, 保存リスト全体)
11: //SQL クエリにおいてタプル内の同じ属性名の値は同じ値にならないといけないので, なっていないものを除去する.
12: for all 全通りの組み合わせ全て do
13:     if 属性名が同じなのに値が違ったら then
14:         全通りの組み合わせリストから除去
15:     end if
16: end for
17: //SQL の検索結果に対して全通りの組み合わせを比較し該当するものがなかったら検索結果に追加する
18: for all 全通りの組み合わせリスト do
19:     for all 検索結果 do
20:         if 一致するものがなかったら then
21:             N/A を追加して検索結果に追加する
22:         end if
23:     end for
24: end for
25: return N/A を追加したタプル群
26: //以下 combine 関数
27: FUNCTION combine(num, リスト 1, リスト 2, 保存リストのサイズ, 保存リスト全体)
28: for all リスト 1 の各要素 do
29:     for all リスト 2 の各要素 do
30:         リスト 1 の要素とリスト 2 の要素を結合し result に格納
31:     end for
32: end for
33: num++
34: if num が保存リストのサイズより小さかったら then
35:     combine(num, result, 保存リストの num 番目のリスト, 保存リストのサイズ, 保存リスト全体)
36: end if
37: return result
```

		female			male		
		13	14	15	13	14	15
DBS	A	1	1	N/A	4	1	1
	B	2	3	4	N/A	1	N/A
	C	N/A	1	1	1	2	1
DM	A	1	4	1	2	2	1
	B	2	N/A	3	2	N/A	N/A
	C	N/A	1	1	1	2	1
Webapp	A	N/A	N/A	4	1	N/A	1
	B	N/A	3	N/A	1	3	1
	C	3	2	1	3	1	N/A

図 9　年齢および性別に対する各授業で A, B, C を取った人数

いといけない. この処理を行うために SSQL クエリから SQL クエリを作成した際のクエリを参照し属性名が一致しているところは同じ値になる様に違う値の場合はその組み合わせを除外している. この処理は 12 行目から 16 行目で行なわれている. ここまででありうる属性値の組み合わせをすべて作成できたので 17 行目のある様に実際にデータベースに問い合わせた結果と比較して抜けているものがあったら 'N/A' などユーザが指定した文字列を組み合わせに追加して検索結果に追加している. 以上の処理で該当する値がない場合の処理が行われている.

4.7　クロス表生成機能の用例

以下に幾つかのクロス表生成機能の用例を挙げる. 本用例で使用するテーブルは

- student(id, name, gender, age, dept)
- class(id, name, day)
- performance(id, s_id, c_id, performance(評定), score(点数))

の三つで student テーブルは生徒情報, class テーブルは授業情報 (授業名と開講する曜日), performance テーブルは student と class の主キーを外部キーとして持ち評定とその科目の点数を持つ.

4.7.1　例 1:生徒全員の各授業に対する評定

クエリは以下の様になる.

生徒全員の各授業に対する評定

```
Generate HTML
        cross_tab([c.day![c.name],], ,
        [s.dept,[s.name]!]! ,
        p.performance)@{side-width=240}
FROM student s, class c, performance p
WHERE s.id = p.s_id AND c.id = p.c_id
```

このクエリにより表頭属性は授業名を開講曜日でグルーピングし縦結合したもので, 表側属性は生徒の情報を学科でグルーピングし名前と横結合したことになる. 実際の値は成績となっている. 実行結果は以下の図 10 となる.

4.7.2　例 2:男女の学科別各授業の A, B, C の生徒人数

クエリは以下の様になる.

		mon		tue
		DBS	DM	Webapp
CS	takei	B	B	C
	tabata	A	B	C
	suzuki	A	B	C
	shu	A	C	B
	seki	A	A	C
	kenji	B	A	C
EE	tamura	A	A	B
	sugiyama	B	C	C
	shishido	A	B	C
	saito	A	C	B
	hakoda	C	A	B
ME	tanaka	C	A	A
	tagawa	C	A	B
	ooki	B	C	C
	kotani	B	A	C
	kajisa	C	A	B
	horaguchi	C	A	B
SD	takaba	C	B	A
	matsuda	B	B	A
	maemura	B	C	B
	kitahara	B	A	A
	kawaguchi	A	C	A
	hunagi	B	A	C
	hibi	B	B	A

図 10　生徒全員の各授業に対する評定

男女の学科別各授業の A, B, C の生徒人数

```
Generate HTML
        cross_tab([s.gender! [s.dept],], ,
        [c.name, [p.performance]!]! ,
        count[s.id])@{side-width=120, null_value='
        該当なし'}
FROM student s, class c, performance p
WHERE s.id = p.s_id AND c.id = p.c_id
```

		female				male			
		CS	EE	ME	SD	CS	EE	ME	SD
DBS	A	1	1	該当なし	該当なし	3	2	該当なし	1
	B	2	該当なし	2	5	該当なし	1	該当なし	該当なし
	C	該当なし	該当なし	1	1	該当なし	1	3	該当なし
DM	A	1	1	2	2	1	1	3	該当なし
	B	2	該当なし	該当なし	3	1	1	該当なし	該当なし
	C	該当なし	該当なし	1	1	1	2	該当なし	1
Webapp	A	該当なし	該当なし	該当なし	4	該当なし	該当なし	1	1
	B	該当なし	1	1	1	1	2	2	該当なし
	C	3	該当なし	2	1	2	2	該当なし	該当なし

図 11　男女の学科別各授業の A, B, C の生徒人数

このクエリにより表頭属性は生徒の性別を学科でグルーピングし縦結合したもので, 表側属性は各評定を授業名でグルーピングし横結合したことになる. 実際の値は該当する生徒の人数となっている. 実行結果は以下の図 11 となる.

4.7.3　例 3:各テストの平均点と各自の点数

クエリは以下の様になる.

各テストの平均点と各自の点数

```
Generate HTML
        cross_tab([c.day ！ [c.name!  avg[p.score]
        ],], ,
        [s.dept,[s.name]!]! ,
        p.score)@{side-width=200}
FROM student s, class c, performance p
WHERE s.id = p.s_id AND c.id = p.c_id
```

このクエリにより表頭属性は授業名を開講曜日でグルーピングしその科目の平均点と縦結合したもので, 生徒名を学科でグルーピングし横結合したことになる. 実際の値は生徒の点数となっている. 実行結果は以下の図 12 となる.

5.　評　　価

5.1　既存技術との比較

2. 章で説明したようにクロス表を作ることができる既存の技術がある. この技術と提案手法を比較していく. 表の構造をモデル化しそのモデル内で各技術の表現力を考えることで行う.

5.1.1　モデル化

モデル化の基準として以下の三つがある.

- 外延集合と内包集合

外延集合とは具体的な値を列挙するものである. 例えば表頭または表側の属性に具体的な定数 {'M','F'} を列挙することである.

内包集合とは集合と論理式により集合を規定することで, 'member.gender' などの表記のことを言う.

モデル化に於いては外延集合となるものを E 型, 内包集合となるものを I 型とする.

- 構造の種別

表頭や表側で使える構造を分類する. 分類の方法としてはただの集合, 木構造 (根が一つの木構造), 森構造 (複数の木構造のリスト) の三つに分けられる.

モデル化ではただの集合を S 型, 木構造を T 型, 森構造を F 型とする.

		mon		tue
		DBS	DM	Webapp
		69.0	70.0	63.0
CS	kenji	62	91	39
	seki	93	94	35
	shu	88	45	76
	suzuki	87	66	43
	tabata	86	58	44
	takei	74	72	47
EE	hakoda	51	84	75
	saito	96	51	63
	shishido	81	67	45
	sugiyama	75	45	44
	tamura	87	88	71
ME	horaguchi	54	89	75
	kajisa	36	99	67
	kotani	64	81	23
	ooki	65	34	49
	tagawa	43	97	65
	tanaka	54	81	85
SD	hibi	64	61	98
	hunagi	76	87	43
	kawaguchi	88	31	91
	kitahara	71	97	86
	maemura	59	43	77
	matsuda	64	71	100
	takaba	54	63	81

図 12　各テストの平均点と各自の点数

- 各木構造での属性の個数

木構造内での属性の指定を複数でも良いのか単独属性しか許さないか, ということで分類する. 複数の属性を許可するものを M 型とする.

5.1.2　各手法のモデル化

上記のモデル化手法を用いて各手法の表現力をモデル化する.

- 提案手法

提案手法では利用したい属性を属性名で指定することもでき, 実際の値として入力することもできる. そのため I 型となる.

また, 提案手法では各表頭と表側と表の値となる属性は全て森構造で表現することを許可している. よって F 型であることがわかる.

各木構造の属性は複数であって良いので M 型であることがわかり, M 型ということになる.

以上のことから今回用いたモデル化において提案手法の表頭, 表側, 値は IFM 型であることがわかった.

- Microsoft SQL Server

まず表頭属性は外延集合となっており, その他の表側と値は内包集合となっている. また,　表頭と表側と値が表現できる構造はただの集合である. 指定できる属性の個数については表側のみ複数の属性の指定が可能である.
以上のことから表頭属性は ES 型, 表側属性は ISM 型, 値は IS 型となっていることがわかる.

- Microsoft Excel

Microsoft Excel はどの属性も内包集合となっている. また木構造までの表現が可能となっており属性指定は全て複数可能である.
以上のことから ITM 型であることがわかる.

上記のことから提案手法が IFM 型であり最も多様な形を表現することができるとわかった.

5.2　アンケート評価

既存研究を使用してもらい '記述は容易だったか', '望んだレイアウトが再現できたか', '改善点' の 3 点を評価項目とした. 評価の際にはこちらでサンプルクエリを用意しそれを改変することでまずは使い方を知ってもらい, その後こちらで用意したクロス表の再現および自由な使用によって行ってもらうものとした. 評価対象は同研究室のメンバーで SSQL に使い慣れてる人, 使い慣れてない人のどちらもを対象とした.

（1） 記述は容易だったか

- 複雑なレイアウトを関数一つで再現できるのは良かったが, 引数に複数の TFE があるため紛らわしくなる.(SSQL 経験者)
- 引数と引数の間がわかりにくい.(SSQL 経験者)
- 普段から SSQL に慣れ親しんでいる人であればほとんど新しい知識無しに容易に記述出来ると感じた. 自分は少ししか触ったことが無かったので少し苦戦した. (SSQL 非経験者)

（2） 望んだレイアウトを再現できたか

- 再現することができた.(SSQL 経験者)
- 時々思った構造にならないことがあったが, 概ね意図したレイアウトを実現できた.(SSQL 非経験者)

（3） 改善点

- 幅の調整が大変. せめて GUI があるといい.(SSQL 経験者)
- 100px にセル幅が固定されているが文字列長によって可変にする.(SSQL 経験者)
- 作る cross table のイメージと Where 句における依存関係のイメージが必要になってしまう.(SSQL 経験者)
- 少し複雑な表頭属性などを記述しようとすると, [] の入れ子が深くなってしまい, 反復子をどこに付ければ良いかがこんがらがってしまうので, もう少し直感的に記述できるようになると便利だと感じた. (SSQL 非経験者)

これらの評価からまず '記述の容易さ' については SuperSQL 経験者でも非経験者でも分かりにくくなることがあることがわかった. また 'レイアウトの再現性' については概ね意図したレイアウトを実現できたことがわかった. 最後に '改善点' としては幅指定の大変さや構造指定の複雑さ, クエリの条件記述の大変さがあることがわかった.SQL と SuperSQL をよく知っている人でも大変に感じてしまうということなのでレイアウトに関しては生成したのちに GUI で修正できるようにすると良いと感じた.

6. ま と め

6.1　結　　論

本研究では SuperSQL におけるクロス表の生成機能の実装をした. このシステムの実装により容易に SuperSQL でクロス表を実装することが可能になり SuperSQL の表現の幅が広がった. このシステムは SuperSQL 上のシステムである, ということよりクロス表を生成する際に他システムとは異なり様々な構造を指定できるという点で有用であると言える.

6.2　今後の課題

この生成機能ではセルの横幅を 100px と自動指定している. しかし左上の空白の枠の幅については自動指定になってしまっている. これを自動で設定する様にすればより使いやすいシステムになると考えられる. また, 実用例で挙げた表よりも複雑な表の実現を目指していきたいと考えている.

文　　献

[1] SuperSQL: http://SuperSQL.db.ics.keio.ac.jp
[2] M. Toyama: "SuperSQL: An Extended SQL for Database Publishing and Presentation", Proceedings of ACM SIGMOD '98 International Conference on Management of Data, pp. 584-586, 1998
[3] Toshiyuki Seto, Takuhiro Nagafuji, Motomichi Toyama. "Generating HTML sources with TFE enhanced SQL", SAC '97 Proceedings of the 1997 ACM symposium on Applied computing, pp. 96-100, 1997.
[4] Microsoft SQL Server 技術マニュアル : https://msdn.microsoft.com/ja-jp/library/ms130214.aspx
[5] Microsoft Excel クイックリファレンス : http://download.microsoft.com/download/0/0/5/005D55B9-82E2-489D-BB55-1B6B529F1B8F/QuickGuide_Excel2013.pdf
[6] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals" Data Mining and Knowledge Discovery 1, 2953 (1997)
[7] Stephen B. Johnson, Damianos Chatziantoniou "Extended SQL for Manipulating Clinical Warehouse Data" Proc AMIA Symp. 1999:819-23.