セル結合を含む表のデータモデル

大西　洋† 田島　敬史††

† 京都大学大学院情報学研究科　〒 606–8501 京都府京都市左京区吉田本町 36–1
†† 京都大学大学院情報学研究科　〒 606–8501 京都府京都市左京区吉田本町 36–1
E-mail: †ohnishi@dl.kuis.kyoto-u.ac.jp, ††tajima@i.kyoto-u.ac.jp

あらまし　セル結合を含む表は，Web ページやスプレッドシートなどの電子データをはじめ，一般の印刷物でも多用される．属性値の冗長性が低下する，集約値を記録できるなど，セル結合の利用は人間にとって利点が多い一方，結合セルの値が何を表すのかというセマンティクスを明示できないため再利用性が低く，機械的な利用が困難だという欠点もある．本研究で提案するデータモデルでは，集約関数によりセル結合のセマンティクスを表現できることに加え，OLAP で用いられる roll-up/drill-down 操作を導入してセルの階層関係を操作できる．これにより，セルの結合や分割を機械的に行えるため，セル結合を含むままで集計処理を行える．

キーワード　集計処理, スプレッドシート, OLAP, 関係データモデル, 関係代数

1. はじめに

1.1 背　景

長方形状のセル (cell) が並べられ，縦方向の列 (column) と横方向の行 (row) からなる表 (table) は，非常に古典的だが重要なデータ記録・整理・表現手法の一つである．現在でも表は実世界の様々な場面で用いられており，情報システムの内部など，表の利用場面はますます増加している．

表をシステム内で用いるための基盤として，関係データベースを操作できる RDBMS(Relational Database Management System) が広く用いられる．RDBMS では関係表と呼ばれる表が利用でき，関係表に最適化された構成となっている．しかし表のうちには，RDBMS で利用可能な関係表以外のものも存在する．

関係表でない表の一つに，縦方向や横方向の複数のセルを結合した，セル結合を含む表がある．図 1 にセル結合を含む表の例を示す．図 1 では，セル「項目 3」が「属性 1」と「属性 2」を，セル「項目 b」が「属性 2」と「属性 3」を，それぞれ横方向に結合している．また，セル「項目 α」は「属性 4」の列を縦方向に結合している．

<table>
<tr><th>col1</th><th>col2</th><th>col3</th><th>col4</th></tr>
<tr><td>val1</td><td>vala</td><td>valA</td><td rowspan="3">valα</td></tr>
<tr><td>val2</td><td colspan="2">valb</td></tr>
<tr><td colspan="2">val3</td><td>valC</td></tr>
</table>

図 1　セル結合を含む表の例

このようなセル結合を含む表は，コンピュータ上で様々な方法で作成できる．例えば LibreOffice Calc や Microsoft Excel などの表計算ソフトには「セルの結合」という機能が存在し，セル結合を含む OpenDocument や OOXML を作成できる．また HTML で表のセルを表す td タグには，行・列方向にセル結合を行う rowspan, colspan 属性が存在し，Web ブラウザ上でセル結合を含む表を描画できる [16].

1.2 現状の問題点

セル結合を含む表は実世界で多用されるが，使うべきでないとされることも多い．例えば，使うべきでない Excel の機能を述べた TechRepublic の記事 [15] では，使うべきでない機能の筆頭に「Avoid merging cells」を挙げている．また，オープンデータ流通推進コンソーシアムの資料 [18][17] では，オープンデータの要件に「すべてのセルが，他のセルと結合されていない」ことを挙げている．

しかし現実には，セル結合を含む表は多数存在する．それほどまでに広くセル結合を含む表が用いられるのは，代表的な表計算ソフトの一つである Microsoft Excel が殆どの PC にプリインストールされていることだけが理由ではないと考えられる．先述の TechRepublic の記事 [15] でも「Don't hesitate to use merged cells if you really need them」としており，セル結合の使用を完全には否定していない．関係表では得られない，セル結合を用いることで得られる利点があるからこそ，セル結合が用いられると考えるのが自然である．このようなセル結合を含む表に対し妥当なデータモデルを構築でき，セル結合を含む表のうち「良い表」(良いセル結合) と「悪い表」(悪いセル結合) が整理できれば，セル結合を含む表を適切に用いることができるようになるはずである．

1.3 研究目的

そこで本研究では，セル結合を含む表を表現可能なデータモデルを提案する．それにより，セル結合に関する理論体系を明確化し，より適切にセル結合を使用する方法を提案することを研究目的とする．

3.2 節で詳述するが，セル結合を含む表には，冗長性の低減や更新不整合の解消などの利点がある．また欠点として，セル結合の意味が明瞭でないこと，そのままではデータとして活用できないことなどがあるため，これらを克服するデータモデルがあれば良いといえる．従って本研究では，セルを結合する意味を集約関数の概念にまとめ，roll-up や drill-down という操作で属性階層を上下させることで活用可能としたデータモデル

を提案する．

本研究の応用例としては，表を基に集計や可視化などの処理をする際に，セル結合を含んだままでも処理を行えるようになることが挙げられる．

なお本研究で提案するデータモデルでは，インスタンス部分に用いられるセル結合以外に，属性名に用いられるセル結合を扱うこともできる．ただし，このような表は 2.1 節で述べるように多数の先行研究があり，また機械的な処理に活かすという研究目的からも外れるため，本研究では主要な対象としていない．またセル結合には，図 1 のように結合後のセルが長方形になるものの他，図 2 のように結合後のセルが階段状になるものもある．図 2 のようなセル結合は年表などでよく用いられるが，こうした表は集計などの用途に用いるのは難しいことから，本研究では対象としていない．

<table>
<tr><th>col1</th><th>col2</th><th>col3</th><th>col4</th></tr>
<tr><td colspan="2">val1</td><td>valA</td><td rowspan="2">valα</td></tr>
<tr><td colspan="2"></td><td></td></tr>
<tr><td colspan="2">val2</td><td colspan="2"></td></tr>
</table>

図 2　長方形でない結合セルを含む表

2. 関 連 研 究

本研究で扱うセル結合を含む表は，データベースの正規化理論においては非第 1 正規形 (non-first normal form, $(\mathrm{NF})^2$) と呼ばれている．そのため 2.1 節では，$(\mathrm{NF})^2$ の概要を述べて先行研究を挙げる．次に 2.2 節で，本研究で援用する OLAP 関連の概念と先行研究について述べる．最後に 2.3 節で，セル結合を含む表に関する先行研究について述べる．

2.1 $(\mathrm{NF})^2$

データベースの正規化理論における第 1 正規形 (1st normal form, 1NF) の定義は，「スキーマを定義するすべてのドメインがシンプル」[3] であることである．ドメインがシンプルであるとは，ドメインが他のドメインで定義されない，即ちドメインが他のドメインの直積や冪集合になっていないことを意味する [21]．つまり，属性値として (日本, 東京) のような組 (tuple) や，{ 日本, アメリカ } のような集合 (set) を含むことが許されないということである．

図 3 のような表は 1NF の定義を満たさないが，このような表は $(\mathrm{NF})^2$ または nested relation と呼ばれる．

$(\mathrm{NF})^2$ については，1970 年代から 1980 年代中頃にかけて盛んに研究された．Makinouchi[9] は，非商業目的のデータ処理に $(\mathrm{NF})^2$ の需要があることを指摘し，関数従属性 (FD) や多値従属性 (MVD) を $(\mathrm{NF})^2$ へ拡張した．Jaeschke ら [7] は，nest, unnest 演算子を導入して関係代数を $(\mathrm{NF})^2$ へ拡張した．Kambayashi ら [8] は，group-by, row-nest, column-nest, relation-nest の 4 種の演算子で 1NF から $(\mathrm{NF})^2$ へ変形でき，関数従属性や結合従属性を操作できることを示した．Abiteboul ら [1] は，$(\mathrm{NF})^2$ での属性の階層構造を format と呼び，format に基づくデータモデルを提案した．Ozsoyoglu ら [13] は以上の研究を踏まえ，NNF(nested normal form) と呼ばれる $(\mathrm{NF})^2$ の正規形を提案した．

<table>
<tr><th>name</th><th>address</th><th>birthdate</th><th>movies</th></tr>
<tr><td>Fisher</td><td><table><tr><th>city</th><th>street</th></tr><tr><td>Maple</td><td>H'wood</td></tr><tr><td>Locust</td><td>Malibu</td></tr></table></td><td>9/9/99</td><td><table><tr><th>title</th><th>year</th><th>length</th></tr><tr><td>Star wars</td><td>1977</td><td>124</td></tr><tr><td>Empire</td><td>1980</td><td>127</td></tr><tr><td>Return</td><td>1983</td><td>133</td></tr></table></td></tr>
<tr><td>Hamill</td><td><table><tr><th>city</th><th>street</th></tr><tr><td>Oak</td><td>B'wood</td></tr></table></td><td>8/3/88</td><td><table><tr><th>title</th><th>year</th><th>length</th></tr><tr><td>Star wars</td><td>1977</td><td>124</td></tr><tr><td>Empire</td><td>1980</td><td>127</td></tr><tr><td>Return</td><td>1983</td><td>133</td></tr></table></td></tr>
</table>

図 3　nested relation の例
(DBInfoBlog「Nested Relations」より転載)

本研究でもこれらの研究と同様に $(\mathrm{NF})^2$ の表を対象としているが，次の 2 点で先行研究と異なっている．まず，$(\mathrm{NF})^2$ に関する先行研究の多くは表の中に表が入っているような入れ子構造 (nested) の表を対象としているが，本研究では扱う表は，セル結合を含む表である．セル結合を含む表は入れ子構造を含む表とも解釈できるが，セルの中にセルがあるのではなく，複数のセルに跨って結合されているとみなすのが自然である．セルを結合するというユーザの行為は何らかの意図に基づいていると考えられるため，通常の入れ子構造の表とは異なると考えられる．

次に，$(\mathrm{NF})^2$ に関する先行研究の多くは階層的な属性を考慮しているが，これらの研究で扱っている階層性と，本研究で扱う階層性とは異なっている．$(\mathrm{NF})^2$ に関する先行研究では，表の中に表があるような階層性を考慮しているため，階層関係にある上下の属性については，形式的な関係性を持つことが少ない．例えば，属性「授業」のセルに属性「学生」「教科書」があるような階層性では，MVD「授業 $\twoheadrightarrow$ 学生 | 教科書」が成り立つ．このような MVD は，関係スキーマの階層性を記述したり，情報無損失分解を行う上では役立つが，この MVD に対してそれ以上の解析処理を行うことはできない．本研究で対象とする属性の階層性は，例えば属性「学部生」「院生」を結合したセルがあるときに，ここに属性「大学生」があると想定し，階層性を考えるものである．この階層性では，先の MVD とは逆向きの FD「学部生, 院生 $\rightarrow$ 大学生」が成り立つ．この FD は，更に集約関数 f が明示されたとき，$f(v_{\text{学部生}}, v_{\text{院生}}) = v_{\text{大学生}}$ となり，機械的に上位の属性値を導出できる (注1)．

2.2 データキューブ (OLAP キューブ)

Gray ら [6] は，データベースにおける集約演算を統合してデータキューブまたは OLAP キューブと呼ばれる構造を提案した．

OLAP は OnLine Analytical Processing(オンライン分析処理)[4] もしくは OnLine Application Processing の略で，ビッグデータ処理などで用いられる技術である．OLAP で関係表に対し発行されるクエリは複雑かつ稀にしか発行されないものが

(注1)：この例の集約関数は，表 1 における関数 sum である．

多いが，そうしたクエリに対して OLAP では高速に集計処理を行うことが必要になる [2]．こうした高速処理は，RDBMS で管理された関係表で集計するには性能面で不十分なことが多いため，データキューブと呼ばれるデータ構造が用いられる．

データキューブでは，関係表の属性を次元属性 (dimentional attribute) と従属属性 (dependent attribute, measured attribute) に分解する．次元属性は関係表の候補キーで，従属属性は次元属性以外の属性である．例えば，関係表「天気 (日時, 緯度, 経度, 高度, 温度, 気圧)」に対して，属性「日時」「緯度」「経度」「高度」は次元属性で，属性「温度」「気圧」は従属属性である．これらの次元属性を座標軸にとり，座標空間上で決定される点と従属属性の属性値を対応させたとき，この座標空間をデータキューブという．

データキューブ上で集計作業を行う際は，空間から超平面を切り取り，その超平面上にある点に対して何らかの集約関数 (aggregation function) を作用させる．集約関数は属性値の組を引数として値を返す関数であり，例を表 1 に示す．集約関数には，属性値の総和を求める関数 sum や，属性値の平均値を求める関数 avg などがある．

表 1　集約関数の例 ($\mathbf{v} := (v_i)$ とする)

関数名	定義	意味
sum	$\mathrm{sum}(\mathbf{v}) := \sum v_i$	属性値の総和.
avg	$\mathrm{avg}(\mathbf{v}) := \frac{1}{\mathrm{count}(\mathbf{v})}\mathrm{sum}(\mathbf{v})$	属性値の平均値.
max	-	属性値の最大値.
min	-	属性値の最小値.
count	-	$\mathbf{v}$ の要素数.

このように，データキューブ上のある超平面を集約関数によって処理 (aggregate) した結果を返す処理を roll-up という．また逆に，集約された (roll-up された) 結果を再び元の空間に戻した結果を返す (de-aggregate) 処理を drill-down という．

このようにデータキューブはデータ処理のための集計機能を提供しているが，集計の基になる表はあくまで関係表であり，欠損値がある場合は考慮されていない．本研究で対象とするセル結合を含む表は，表の一部の値が欠損し，代わりに一部のセルに対して集計が行われている，いわば「中途半端に」roll-up された表であるとみなすことができる．

2.3 「ネ申 Excel」とバッドデータ

セル結合を含む表のうち特に推奨されないものに，「ネ申 Excel」と呼ばれる表がある．奥村 [22] によれば「ネ申 Excel」(注2) とは，表計算ソフトを利用して作成された「巧妙だがデータとしての再利用性の低い Excel ファイル」のことである．

例えば図 4 は，東日本大震災直後に観測されたセシウムの大気中での飛散量を記録した表である．図 4 では，観測機器の不備や観測体制の変化などにより，セシウムの個々の同位体の飛散量を記録したときと，すべての同位体の合計値しか記録していないときがある．

このように表中にセル結合を含んでいるために機械的な処理

(注2)：「かみエクセル」などと読む．由来の詳細は [22] を参照のこと．

検査機関	採取日	^{131}I	^{134}Cs	^{137}Cs
○○検査機関	H23.10.1	5	5	5
◇◇検査機関	H23.10.1	-	25	25
△△検査機関	H23.10.1	10	10	
☆☆検査機関	H23.10.1	-	25	25

^{134}Cs,^{137}Cs の合計値

図 4　放射線量の測定表
(厚生労働省「食品中の放射性物質の検査結果について」より転載)

が困難なデータを，一般にバッドデータと呼ぶ．Murrell[10] はバッドデータが生じる原因として，機械でなく人間が読むことを意図してデータが作られることを指摘している．例えば行政機関では，基本的に人間に認識可能な形で情報を公表することを最優先にしているため，図 4 のような人命に関わる緊急性の高いデータがバッドデータとして提供されている [12]．

3. セル結合の利点と欠点

本章では，セル結合を用いる利点・欠点を列挙した上で，セル結合を含む表のデータモデルを提案するにあたり考慮すべき点をまとめる．

3.1 セル結合の欠点

まず，セル結合を用いる欠点として，次のものが挙げられる．

- セル結合を用いると，結合セルの位置を示して値を取得したり，位置を認識して処理したりするのが困難になる．特に HTML の表の場合は，結合セルがある位置についてはセルの定義自体がなくなる．例えば，2 行からなる表において，1 行目の最初の td タグで rowspan 属性の値が 2 になっていた場合，2 行目の最初の td タグが 2 列目のセルを表すことになってしまう．このように，DOM での td タグの位置と実際の表中での位置が対応しなくなると，DOM の解析処理は複雑になる．
- セル結合を用いると，データの解析が困難になり再利用性が低下する [22]．オープンデータとして公開されたものを統計処理をしようとした場合，結合セルの値を取得するのが困難になる．
- セル結合を用いると，セルの意味が不明瞭になり再利用性が低下する．Tajima ら [14]，奥村 [22] にあるように，セルの結合は様々な意図でなされるので，単純にセル結合を解除して，同じ値を各セルに入れるだけでは不十分である．表の作者がどのような意図でセルを結合したかは機械的に認識できないため，セル結合を含むデータの利用は困難である．
- セル結合を用いると，表計算ソフト上での操作が困難になる．例えば，セル結合を含む列について表を整列しようとした場合，セル結合を含む行の属性値が不明なため，セル結合を含む表は整列できない [15][5][11]．また，マクロ言語による表の操作にも通常のセルと異なる措置が必要になる [11]．表中の列に設定した書式設定が結合セルには適用されなかったり，表のコピーに支障が出たり，セル結合を含む行の前後でカーソルを移動させた際に，カーソルが別の行へ移動してしまったりする点も問題である [19]．
- セル結合を用いると，表の読み込み速度が低下する．ビッグデータのような巨大な表の場合，セル結合を含むと読み込み速度に影響を及ぼす [20]．

- セル結合を用いると，表が 1NF でなくなり DBMS で扱えなくなる [22]．理論上の問題だけでなく，現在存在する DBMS ではセル結合を含む表を扱えない．本研究では関係スキーマを拡張し，DBMS でセル結合を含む表を扱えるようにすることで得られる理論的・実利的な有用性を示す．

3.2 セル結合の利点

次に，セル結合を用いる利点として，次のものが挙げられる．

- セル結合を用いると，表の視認性が向上する．つまり，目視で表を認識し易くなる．人間が表を読む場合，同じ値を含むセルが幾つも並んだ表があると，すべてのセルに目を通して初めて，それらのセルが同じ値だと判断できる．似たような値を含むセルが並んでいると，それらのセルの中に別の値が紛れ込んでいる可能性もあるため，よほど注意深く一つ一つのセルを読む必要があり，可読性が非常に低い．セル結合を用いることで，多くのセルに目を通さなくても，一つのセルの値を読むだけでその部分を認識できるため，可読性が高くなる．
- セル結合を用いると，冗長性が低減される．セル結合を含む表は，結合前の属性と結合後の属性を分けた 1NF に変換できる．この 1NF では，セル結合を含む行に対して，結合前の属性値を空値とせざるを得ないため，インスタンス中に空値が多くなる場合があり冗長である．一方，結合前の属性値が既知の行に対しては，結合前の属性から結合後の属性への FD が成り立つので，結合後の属性値を記録する場合も，記録せず空値とする場合も冗長である．また，セル結合を含む表を 2 つの関係表に分割することも可能だが，同じ目的の表が 2 つあるのはそれ自体冗長である．セル結合を用いれば，こうした冗長性を生じることなく 1 つの表でまとめることができ，外見上も空間計算量の面でも有用である．
- セル結合を用いると，1NF で生じる更新不整合が解消される．一つの表の中に同じ値を含むセルが複数ある場合，その値を変更しようとしたときには，それらすべてのセルを変更しなければならない．その際，変更を忘れたセルが存在すると，変更前の値が残ってしまうため，更新不整合が生じる．セル結合を用いれば，1 つのセルの値さえ書き換えれば値を更新できるようになり，更新不整合が生じない．
- セル結合を用いると，セル個々に入れる値が何らかの理由で不明なときにも，それらの集約値が分かっていれば，値を記録できる．例えば図 4 では，セル結合を用いることで，2 つの同位体の飛散量の合計値を結合後の属性値として記録することができる．

3.3 方　　針

以上の利点と欠点を，セル結合を含む表の作成者側・利用者側の視点と，表を処理する人間・機械の視点の 2 軸で分類してまとめると，図 5 のようになる．

図 5 より，セル結合には冗長性の低減や更新不整合の解消などの利点があり，表の作成者や利用者といった人間にとって良い構造だといえる．一方で，セル結合の意味が明瞭でないこと，そのままではデータとして解析可能でないことなどの欠点があり，表を機械的に利用する際に困難な構造だといえる．従って次章で，集約関数の概念を用いてセルを結合する意味を表し，

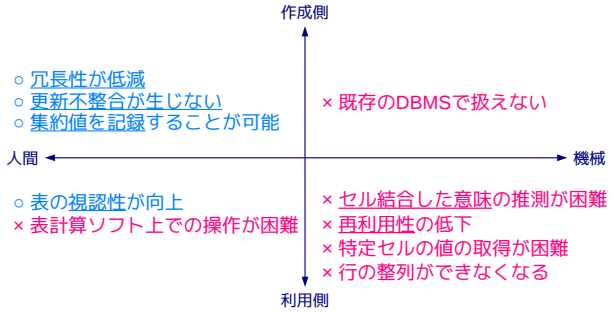

図 5　セル結合の利点と欠点

roll-up や drill-down という操作を導入して属性階層を上下させることで解析可能とした，機械的に処理しやすいデータモデルを提案する．

4. 提案するデータモデル

本章では，セル結合を含む表のデータモデルと，セル結合の適切さに関する分類，それによって可能となる事柄について述べる．提案するデータモデルにおいても，関係代数と同様に，表のスキーマとインスタンスが基本的な要素となる．まず 4.1 節では，セル結合を含む表のスキーマを定義する．次に 4.2 節では，セル結合の意味を明示する集約関数を定義し，スキーマの集約関数を変更する方法について述べる．4.3 節では，セルや組を定義した上で，セル結合を含む表のインスタンスを定義する．

続いて 4.4 節で，属性間の階層関係に従って表を操作する roll-up, drill-down の概念を述べる．これらの概念は，2.2 節で述べたデータキューブにおける概念を援用したものである．また，4.5 節での準備を基に，4.6 節で関係代数と類似した代数操作を定義する．

4.1 ス キ ー マ

定義 1 (セル結合を含む表)　関係スキーマを R，属性の階層関係を示す有向グラフを T，使用可能な集約関数の集合を F としたとき，3 つ組 (R, T, F) をセル結合を含む表の**スキーマ**と呼ぶ．

R は，表中の属性を $A_1, \cdots, A_n$ とするとき，$R(A_1, \cdots, A_n)$ と表される．なお，本論文では属性 A_i とその定義域を同一視し，いずれも単に A_i と表す．

T は節点集合 V_T と枝集合 E_T の組 (V_T, E_T) で表される．T の節点は，R の属性とその上位もしくは下位にある属性に対応する点であり，V_T はそれらの節点を元とする集合である．セル結合があるとき，T にはそれに対応する「結合前の属性 $\rightarrow$ 結合後の属性」という枝が存在し，E_T はそれらの枝を元とする集合である (注3)．また E_T には，$A_1, \cdots, A_n$ に対応する節点 (葉) についてのみ，始点と終点が同じ枝 (自己ループ) を含むことができる．この自己ループは，行方向のセル結合が存在することに対応する．葉以外の節点に関する自己ループや，その他の閉路は T には含まれないものとする．

(注3)：つまり T の枝は，根から葉へ向かうのではなく，葉から根へ向かう．

また F の元は，表 1 に列挙されたものをはじめとする集約関数である．集約関数は，複数の属性値を一つの属性値へ変換する関数であり，4.2 節で詳しく述べる．集約関数は定義域と値域が明示されており，属性の階層関係を表現しているため，集約関数が既知であれば，T がなくても原理上，セル結合を含む表を記述できる．しかし，集約関数が未知の場合は $F = \emptyset$ となりうるため，T がなければ属性の階層関係を表現できない．

なお，2.1 節で述べた nested relation には，$(\mathrm{NF})^2$ の表を記述する関係スキーマに似た記法が存在する．この記法に依れば，表 3 は下のように表される．

R(name, address(city, street), birthdate, movies(title, year, length))

本研究で扱うセル結合を含む表の一部はこの記法で表せるが，nested relation の記法で表せない表も存在する．これは，ある属性がセル結合するときに常に同じ属性とセル結合するとは限らないからであり，例えば表 1 は nested relation の記法では表現できない．表 1 を本研究のデータモデル (R, T, F) で表すと，グラフ T は図 6 中のようになる．図 6 中にはセル結合後の属性として「属性 12」「属性 23」があるが，これらの属性名は便宜上つけたものである．

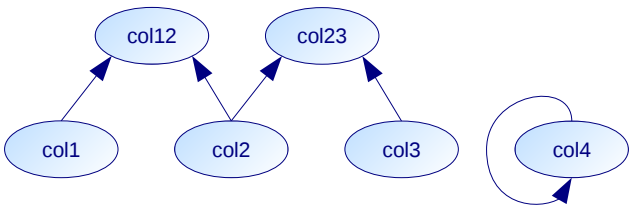

図 6　表 1 の階層関係グラフ T

4.2　集約関数

セル結合を含む表 (R, T, F) における F の元は，集約関数と呼ばれる関数である．本研究では集約関数を次のように定める．

定義 2 (集約関数)　セル結合を含む表 (R, T, F) に属性 $A_{k_1}, \cdots, A_{k_m} (\in V_T)$ 及び属性 $A_l (\in V_T)$ があるとする．このとき，$\prod_{i=1}^{m} A_{k_i}$ から A_l への関数を属性 $A_{k_1}, \cdots, A_{k_m}$ から属性 A_l への**集約関数**という．

集約関数にも通常の関数と同様に，準同型や全単射の性質を定義できる．集約関数の例を表 2 に示す．なお，$\mathbf{v} := (v_i)$ であり，集約関数に引数として与える属性値のベクトルを示す．表 2 では，2.2 節で述べたもの以外にも，本研究で集約関数として扱う関数が存在する．

スキーマ (R, T, F) に存在しない集約関数 $f (\notin F)$ を追加する際は，同時に T の木構造も更新する必要がある．従って，元々存在しない集約関数を追加したり，既にある集約関数を削除したりするために必要な演算を定義する．

定義 3 (集約関数の追加と削除)　$S := (R, T, F)$ をスキーマ，

表 2　集約関数の例と全単射性

関数名	全単射	意味	例
sum	×	属性値の総和	$\mathrm{sum}(1,2,3) = 6$
avg	×	属性値の平均値	$\mathrm{avg}(1,2,3) = 2$
max	×	属性値の最大値	$\mathrm{max}(1,2,3) = 3$
min	×	属性値の最小値	$\mathrm{min}(1,2,3) = 1$
count	×	$\mathbf{v}$ の要素数	$\mathrm{count}(1,2,3) = 3$
cat	×	属性値を連結した文字列	$\mathrm{cat}(1,2,3) = 123$
set	×	属性値の集合	$\mathrm{set}(1,2,1) = \{1,2\}$
tuple	○	属性値の組	$\mathrm{tuple}(1,2,3) = (1,2,3)$
pack	○	$\mathbf{v}$ がすべて同じ値なら v_1，そうでなければ $\mathrm{tuple}(\mathbf{v})$	$\mathrm{pack}(1,2,3) = (1,2,3)$ $\mathrm{pack}(1,1,1) = 1$
δ	○	属性名変更 (4.4 節参照)	

$f : \{A_\lambda\}_{\lambda \in \Lambda} \to A_i$ を集約関数とする．このとき，スキーマ

$$\oplus_f S := (R, \oplus_f T, F \cup \{f\})$$
$$\oplus_f T := (V_T \cup \{A_\lambda\}_{\lambda \in \Lambda} \cup \{A_i\}, E_T \cup \{(A_\lambda, A_i) \mid \lambda \in \Lambda\})$$

を S への集約関数 f の**追加**と呼ぶ．また，スキーマ

$$\ominus_f S := (R, \ominus_f T, F \setminus \{f\})$$
$$\ominus_f T := (V_T, E_T \setminus \{(A_\lambda, A_i) \mid \lambda \in \Lambda\})$$

を S からの集約関数 f の**削除**と呼ぶ．

以上のように集約関数を追加することで，元の表に存在しなかった集約処理を行うことができるようになる．

4.3　インスタンス

本節では，表の各行に相当するデータと，行を跨ぐセル結合のデータモデルを提案する．まず，4.1 節のスキーマに基づくインスタンスのデータモデルを提案するにあたり，表の基本単位となるセルを定義する．

定義 4 (セル)　3 つ組 $(v_{ij}, g, f) \in A_j \times G \times \{\mathrm{True}, \mathrm{False}\}$ を行 i の属性 A_j に対応する**セル**と呼ぶ．ここで，G は後述するグループの集合で，$g (\in G)$ はこのセルが属するグループを表す．また f は値 v_{ij} が有効なとき True，無効なとき False となる．

次に，複数のセルからなり，表の行に相当する組を定義する．さらに，組の有限集合として，次のようにインスタンスを定義する．

定義 5 (組)　属性 A にセルを対応させる写像 $x : \{A_j\} \to A_j \times G \times \{\mathrm{True}, \mathrm{False}\}$ をスキーマ S 上の**組**と呼ぶ．また，スキーマ S 上の組の有限集合 $X := \{x_i\}$ をスキーマ S の**インスタンス**，その部分集合を**グループ**と呼ぶ．

表中に行方向のセル結合が存在するとき，1 つのセル結合を構成する組に対してグループが 1 つ定義される．このとき，グループに対しても，属性にセルを対応させる写像として組が定義される．

4.4 roll-up, drill-down

次に，スキーマやインスタンスを結合・分割する一般的な操作として，roll-up と drill-down を定義する．roll-up は，集約関数 $f(\in F)$ に基づいて属性 $\mathrm{dom}(f)$ に対応するセルを結合する操作を表す．

定義 6 (roll-up) I をスキーマ (R,T,F) のインスタンス，$f:(A_\lambda)_{\lambda\in\Lambda}\to A_i$ を集約関数とする．このとき，集合

$$\triangle_f I := \{\triangle_f x \mid x \in I\}$$

$$\triangle_f x(A) := \begin{cases} f(x(\mathrm{dom}(f))) & (A = \mathrm{im}(f)) \\ x(A) & (A \notin \mathrm{dom}(f)) \\ \text{undefined.} & (\text{otherwise}) \end{cases}$$

はスキーマ $\triangle_f (R,T,F) := (R \cup \mathrm{im}(f) \setminus \mathrm{dom}(f), T, F)$ のインスタンスであり，これを I の集約関数 f による **roll-up** と呼ぶ．

また，roll-up の逆操作として drill-down を定義する．drill-down は，集約関数 $f(\in F)$ の逆関数 f^{-1} に基づいて属性 $\mathrm{im}(f)$ に対応する結合セルを分割する操作を表す．

定義 7 (drill-down) I をスキーマ (R,T,F) のインスタンス，$f:(A_\lambda)_{\lambda\in\Lambda}\to A_i$ を集約関数とする．f の逆関数 f^{-1} が存在するとき，集合

$$\nabla_f I := \{\nabla_f x \mid x \in I\}$$

$$\nabla_f x(A) := \begin{cases} f^{-1}(x(\mathrm{im}(f))) & (A \in \mathrm{dom}(f)) \\ x(A) & (A \neq \mathrm{im}(f)) \\ \text{undefined.} & (\text{otherwise}) \end{cases}$$

はスキーマ $\nabla_f(R,T,F) := (R \cup \mathrm{dom}(f) \setminus \mathrm{im}(f), T, F)$ のインスタンスであり，これを I の集約関数 f による **drill-down** と呼ぶ．

roll-up は任意の集約関数に対して行うことができるが，drill-down は，roll-up に用いられた集約関数 f に逆関数 f^{-1} が存在するときのみ可能な操作である．drill-down を行える，即ち逆関数が存在する集約関数を，**drill-down 可能**な集約関数と呼ぶ．関数に対して逆関数が存在することは全単射であることと同値なので，全単射な集約関数は drill-down 可能である．

表 1 に挙げた関数のうち，drill-down 可能な集約関数は関数 pack と関数 tuple である．∇_{pack} は，結合されたセルを分割し，結合後の属性値を結合前の各属性の値とする操作である．∇_{tuple} は，組になったそれぞれの要素を結合前の各属性の値とする操作である．

roll-up, drill-down の定義より，次の定理が成り立つ．

定理 8 (roll-up と drill-down の関係) $S=(R,T,F)$ をスキーマ，I を S のインスタンスとする．S に drill-down 可能な集約関数 $f(\in F)$ が存在するとき，

$$\nabla_f S \triangle_f S = \triangle_f S \nabla_f S = S$$
$$\nabla_f I \triangle_f I = \triangle_f I \nabla_f I = I$$

特に，属性名を変更する関数は特殊な集約関数とみなすことができ，次のように定義できる．

定義 9 (属性名変更) $S:=(R,T,F)$ をスキーマ，I を S のインスタンス，$A_{\mathrm{old}}\in V_T$ とする．集約関数 $f:A_{\mathrm{old}}\to A_{\mathrm{new}}$ を $f(v):=v$ と定めるとき，

$$\delta_{A_{\mathrm{old}}\to A_{\mathrm{new}}} S := \triangle_f \oplus_f S$$
$$\delta_{A_{\mathrm{old}}\to A_{\mathrm{new}}} I := \triangle_f \oplus_f I$$

をそれぞれ，S と I の A_{old} から A_{new} への**属性名変更**と呼ぶ．

特に，すべての属性名に接頭辞 i を付加する操作を，$\delta_i := \delta^{A\in R_i}_{A\to i.A}$ と定義する．

4.5 木構造・関数集合に対する演算

本節では，4.6 節でセル結合を含む表に対する演算を定義するにあたり，まず木構造を一般化した有向グラフに対する演算を定義する．有向グラフは節点の集合 V と，一方の節点を始点，他方を終点とする枝の集合 $E(\subset V^2)$ の組 (V,E) として定義される．$T_i=(V_i,E_i)(i=1,2)$ を有向グラフ，V を節点の集合，E を枝の集合とするとき，

$$T_1 \cup T_2 := (V_1 \cup V_2, E_1 \cup E_2)$$
$$T_1 \cap T_2 := (V_1 \cap V_2, E_1 \cap E_2)$$
$$\delta_{A\to B} T := (\delta_{A\to B} V, \delta_{A\to B} E)$$
$$\delta_{A\to B} V := (V \cup \{B\}) \setminus \{A\}$$
$$\begin{aligned}\delta_{A\to B} E := &\{(e_i,e_j)\in E \mid e_i, e_j \neq A\} \\ &\cup \{(B,e_j)\in E \mid (A,e_j)\in E\} \\ &\cup \{(e_i,B)\in E \mid (e_i,A)\in E\}\end{aligned}$$

次に，写像 x やその集合 F に対して，x の集合 V への制限 $x|_V$ と，A_{old} から A_{new} への属性名変更 $x|_{A_{\mathrm{old}}\to A_{\mathrm{new}}}$ を定義する．これらは次節以降で，集約関数や組に対して用いられる．

$$x|_V(A) := \begin{cases} x(A) & (A \in V) \\ \text{undefined.} & (\text{otherwise}) \end{cases}$$

$$x|_{A_{\mathrm{old}}\to A_{\mathrm{new}}}(A) := \begin{cases} \text{undefined.} & (A = A_{\mathrm{old}}) \\ x(A_{\mathrm{old}}) & (A = A_{\mathrm{new}}) \\ x(A) & (\text{otherwise}) \end{cases}$$

$$\delta_{A\to B} F := \{f|_{A\to B} \mid f \in F\}$$

4.6 演　　算

本節では，前節までで提案したデータモデルに対して，関係代数と同様に各種の演算を定義する．まず，通常の関係代数とほぼ同様の演算として，和集合や共通部分をとる操作を定義する．

定義 10 (和集合, 共通部分) $I_i(i = 1,2)$ をスキーマ (R, T_i, F_i) のインスタンスとする．このとき，集合 $I_1 \cup I_2$ はスキーマ $(R, T_1 \cup T_2, F_1 \cup F_2)$ のインスタンスであり，これを I_1 と I_2 の**和集合**と呼ぶ．また，集合 $I_1 \cap I_2$ はスキーマ $(R, T_1 \cap T_2, F_1 \cap F_2)$ のインスタンスであり，これを I_1 と I_2 の**共通部分**と呼ぶ．

インスタンスの和集合を構成する際に，単に組の和集合をとるのみでは，和集合に類似するグループが含まれる可能性がある．そのため，組が属するグループについて，グループの同類性を判断する何らかの指針を設けてグループの統合処理を行う必要がある．また，もしセル結合に用いられた集約関数が drill-down 可能であった場合には，予めすべての組を一度 drill-down し，組の和集合に対して同じ集約関数で roll-up を行う，ということも考えられる．即ち，単に集合 $I_1 \cup I_2$ をとるのではなく，集合 $\triangle_f (\nabla_f I_1 \cup \nabla_f I_2)$ をとることで，類似したグループを統合できる．

次に，2 つのインスタンスに対する直積演算を定義する．

定義 11 (直積) $I_i(i = 1,2)$ をスキーマ (R_i, T_i, F_i) のインスタンスとする．このとき，集合

$$I_1 \times I_2 := \delta_1 I_1 \times \delta_2 I_2$$

はスキーマ $(R_1 \times R_2, \delta_1 T_1 \cup \delta_2 T_2, \delta_1 F_1 \cup \delta_2 F_2)$ のインスタンスであり，これを I_1 と I_2 の**直積**と呼ぶ．

直積では，もとの表中のセル結合が列方向・行方向共にそのまま保存される．即ち列方向のセル結合では，属性間の階層関係が直積をとる前後で変化しない．また行方向のセル結合では，直積をとる前の各インスタンスに含まれていたグループが直積をとった後も残る．

図 7 に，2 つのインスタンス $X = \{x_1, x_2, x_3\}, Y = \{y_1, y_2, y_3\}$ の直積 $X \times Y$ の例を示す．直積 $X \times Y$ のセル結合では，結合関係にあるセルを 2 次元上で結合して表現できないため，図 7 中では色を付けてセルの結合を表現している．図 7 では，インスタンス X にグループ $g_1 := \{x_1, x_2\}, g_2 := \{x_2, x_3\}$ が，インスタンス Y にグループ $g_3 := \{y_2, y_3\}$ が存在する．これらをそれぞれ $G_X(:= \{g_1, g_2\}), G_Y(:= \{g_3\})$ と表すと，直積のインスタンス $X \times Y$ には，$g_1 \times Y, g_2 \times Y, X \times g_3$ の 3 つのグループが存在している．従って $X \times Y$ のグループ集合は $\{g \times Y \mid g \in G_X\} \cup \{X \times g \mid g \in G_Y\}$ となり，直積をとる前のグループが失われていないことが分かる．

次に，列方向・行方向に抽出処理を行う演算として，射影，選択の各演算を定義する．また，2 つの表から条件を満たす組のみを抽出する演算として，結合演算が定義される．条件式は $A < B, A = B$ などの式，もしくはその組み合わせであり，一般に $A \odot B$ のように表される．

定義 12 (射影) I をスキーマ (R, T, F) のインスタンス，$A(\subset V_T)$ を R の属性集合とする．このとき，集合

$$\pi_A I := \{x|_A \mid x \in I\}$$

X	A1	A2	A3
x1	v11		v13
x2			v23
x3	v31	v32	

Y	B1	B2	B3
y1	w11		w13
y2	w21	w22	
y3	w31		

X × Y	X.A1	X.A2	X.A3	Y.B1	Y.B2	Y.B3
x1y1	v11	v11	v13	w11	w11	w13
x1y2	v11	v11	v13	w21	w22	w22
x1y3	v11	v11	v13	w31	w22	w22
x2y1	v11	v11	v23	w11	w11	w13
x2y2	v11	v11	v23	w21	w22	w22
x2y3	v11	v11	v23	w31	w22	w22
x3y1	v31	v32	v23	w11	w11	w13
x3y2	v31	v32	v23	w21	w22	w22
x3y3	v31	v32	v23	w31	w22	w22

図 7　インスタンス X, Y の直積 $X \times Y$

はスキーマ $(A, T \cap A, F|_A)$ のインスタンスであり，これを I の属性集合 A への**射影**と呼ぶ．

射影 $\pi_A I$ の対象として，元の関係スキーマに存在しない属性 $A(\in V_T \setminus R)$ を指定することも想定される．そのような場合には，適切な集約関数 $f(\in F)$ を用いて roll-up もしくは drill-down を事前に行えばよい．即ち，$\mathrm{im}(f) = A$ となる集約関数 f があれば $\pi_A \triangle_f I$ を，$A \subset \mathrm{dom}(f)$ となる集約関数 f があれば $\pi_A \nabla_f I$ をとれば，元々存在しなかった属性値を求めて射影できるようになる．

また，次のように組 x の定義を拡張すれば，射影は関係代数と同じ定義のままで，集計処理に活用できる．

$$x(A) := \begin{cases} x(A) & (A \in \mathrm{dom}(x)) \\ (\triangle_f\, x)(A) & (\exists f; A = \mathrm{im}(x)) \\ (\nabla_f x)(A) & (\exists f; A \subset \mathrm{dom}(x)) \\ \text{undefined} & (\text{otherwise}) \end{cases}$$

定義 13 (選択) I をスキーマ (R, T, F) のインスタンスとする．このとき，集合

$$\sigma_{A \odot B} I := \{x \in I \mid x(A) \odot x(B)\}$$

はスキーマ $(\sigma_{A \odot B} R, T, F))$ のインスタンスであり，これを I の条件式 $A \odot B$ による**選択**と呼ぶ．

定義 14 (結合) $I_i(i = 1,2)$ をスキーマ (R_i, T_i, F_i) のインスタンスとする．このとき，集合

$$I_1 \bowtie_{A \odot B} I_2 := \{x \in I_1 \times I_2 \mid x(A) \odot x(B)\}$$

はスキーマ $(R_1 \cup R_2, T_1 \cup T_2, F_1 \cup F_2)$ のインスタンスであり，これを I_1 と I_2 の条件式 $A \odot B$ による**結合**と呼ぶ．

選択や結合についても，条件式中に元の関係スキーマに存在しない属性 $A(\in V_T \setminus R)$ を指定することも想定される．この場合も，射影と同様にして，必要なときに roll-up や drill-down を行うよう，演算か組の定義を拡張すれば対応できる．

以上のように演算を定義すると，次の定理が成り立つ．

定理 15 (結合と直積・選択)

$$I_1 \bowtie_{A \odot B} I_2 = \sigma_{A \odot B}(I_1 \times I_2)$$

5. 結　　論

本研究では，セル結合を含む表を対象として，セル結合を含む表の理論的な枠組みや分類基準を提案した．

本研究の応用例として，C_2 の表については，roll-up や drill-down により，セル結合を含んだままでも集計や可視化などの処理を行うことができるようになることが挙げられる．

本研究における研究課題として，次のものが挙げられる．

- 本研究では，セル結合を含む表のデータモデルを提案したが，実際に DBMS へ実装し評価することはまだできていない．従って，実装上の空間的・時間的な計算量についての評価が行えていない．そのため，既存の DBMS に提案したデータモデルを実際に実装し，セル結合を含む表を扱える実装上の利点・欠点を考証する必要がある．
- 本研究では，1NF の表をセル結合を含む表に変換する手法を提案していない．1NF の表をセル結合を含む表に変換する際は，単に集約関数 pack を用いて同じ属性値を含むセルを結合すれば良いだけでなく，より深く表の構造を解析することが必要になる．結合するのに適切なセルを発見する手法が研究されれば，与えられた表を人間にとってより読みやすい形に変換することが可能となる．

これらの研究課題に対しては，今後の研究で取り組むことが期待される．

謝　　辞

本研究は JSPS 科研費 26280112 の助成を受けたものです．

文　　献

[1] Serge Abiteboul and Nicole Bidoit. “Non First Normal Form Relations to Represent Hierarchically Organized Data”. In: Proc. of the 3rd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems. 1984, pp. 191–200. DOI: 10.1145/588011.588038.

[2] Surajit Chaudhuri and Umeshwar Dayal. “An Overview of Data Warehousing and OLAP Technology”. In: SIGMOD Rec. 26.1 (1997), pp. 65–74. DOI: 10.1145/248603.248616.

[3] Edgar F. Codd. “A relational model of data for large shared data banks”. In: Commun. ACM 13.6 (1970), pp. 377–387. DOI: 10.1145/362384.362685.

[4] Edgar F. Codd, C. T. Codd S. B.Salley, and C. T. Salley. Providing OLAP (On-Line Analytical Processing) to User-Analysts: An IT Mandate. 1993.

[5] Excel-User.com. Avoid Merged cells in Excel. 2012. URL: http://www.excel-user.com/2012/01/avoid-merged-cells-in-excel.html.

[6] Jim Gray et al. “Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals”. In: Data Mining and Knowledge Discovery 1.1 (1997), pp. 29–53. DOI: 10.1023/A:1009726021843.

[7] G. Jaeschke and H. J. Schek. “Remarks on the Algebra of Non First Normal Form Relations”. In: Proc. of the 1st ACM SIGACT-SIGMOD Symposium on Principles of Database Systems. 1982, pp. 124–138. DOI: 10.1145/588111.588133.

[8] Yahiko Kambayashi, Katsumi Tanaka, and Koichi Takeda. “Synthesis of unnormalized relations incorporating more meaning”. In: Information Sciences 29.2-3 (1983), pp. 201–247. DOI: 10.1016/0020-0255(83)90016-6.

[9] Akifumi Makinouchi. “A Consideration on Normal Form of Not-necessarily-normalized Relation in the Relational Data Model”. In: Proc. of the Third International Conference on Very Large Data Bases - Volume 3. 1977, pp. 447–453.

[10] Paul Murrell. “機械ではなく人間が使うことを意図したデータ”. In: バッドデータハンドブック. O’Reilly Japan, 2013. Chap. 3, pp. 33–54. ISBN: 9784873116402.

[11] OfficeArtilces.com. Merged Cells in Microsoft Excel. URL: http://www.officearticles.com/excel/merged_cells_in_microsoft_excel.htm.

[12] Haruhiko Okumura. “The 3.11 Disaster and Data”. In: Journal of Information Processing 22.4 (2014), pp. 566–573. DOI: 10.2197/ipsjjip.22.566.

[13] Z. Meral Ozsoyoglu and Li-Yan Yuan. “A New Normal Form for Nested Relations”. In: ACM Trans. Database Syst. 12.1 (1987), pp. 111–136. DOI: 10.1145/12047.13676.

[14] Keishi Tajima and Kaori Ohnishi. “Browsing Large HTML Tables on Small Screens”. In: Proc. of the 21st Annual ACM Symposium on User Interface Software and Technology. 2008, pp. 259–268. DOI: 10.1145/1449715.1449758.

[15] TechRepublic. 10 ways to keep Excel from biting you in the butt. 2011. URL: http://www.techrepublic.com/blog/10-things/10-ways-to-keep-excel-from-biting-you-in-the-butt.

[16] W3C. XHTML Modularization 1.1 - Second Edition. 2010. URL: http://www.w3.org/TR/xhtml-modularization.

[17] オープンデータ流通推進コンソーシアム. オープンデータ化のためのデータ作成に関する技術ガイド. 2013. URL: http://www.opendata.gr.jp/committee/docs/gijyutsu_siryo3-4_20130226.pdf.

[18] オープンデータ流通推進コンソーシアム. オープンデータ流通推進コンソーシアムの取組と提言. 2013. URL: http://www.kantei.go.jp/jp/singi/it2/densi/dai3/siryou3.pdf.

[19] サラリーマン八神かかしの「パソコンおシゴト日記」. それでも嫌いなセルの結合. URL: http://ameblo.jp/oshigoto-pc/entry-10288618647.html.

[20] ExcelVBA ラウンジ. 結合セルは処理速度が遅くなる? URL: http://www.keep-on.com/excelyou/2000lng4/200011/00110458.txt.

[21] 増永良文. データベース入門. Computer Science Library 14. サイエンス社, 2006. ISBN: 4781911404.

[22] 奥村 晴彦. “「ネ申 Excel」問題”. In: 情報処理学会情報教育シンポジウム SSS2013 論文集 (2013), pp. 93–98. URL: http://oku.edu.mie-u.ac.jp/~okumura/SSS2013.pdf.