関係論理表現に現れる集約関数の最適関係代数表現への変換法†

中野良平†† 斉藤和巳††

代表的な関係データベース言語は関係論理に基づくが，データベースマシンのサポート言語は多くの場合関係代数である．したがって，関係論理で表現した検索を，データベースマシンでの実行を想定して，最適な関係代数表現に変換する研究が重要になる．関係論理表現に集約関数が入って来ると，閉じないアルファが現れるので，関係代数への変換は容易でない．本論文は関係論理表現に現れる集約関数をKlugの補正も考慮に入れた最適な関係代数表現に変換する体系を述べたものである．Klugの補正に効率良く対処するため，関係代数演算に新しいタイプの集約演算を導入する．新変換法の基本的アイデアは，代数表現への変換が容易な標準集約形を中継地点とし，それの生成と解決という2フェーズの変換体系にある．同法は3種の基本変換則と3種の発見的変換則から構成される．新変換法の目的は，集約関数を含んだ関係論理表現を人間が考え出すような最適な関係代数表現に変換することにある．変換プログラムを作成し，考えられる様々な複雑な検索に適用して，極めて満足すべき結果が得られることを確認した．

1. まえがき

代表的な関係データベース言語SQL[1]，QUEL等は関係論理に基づいているが，関係モデルの性能問題克服の期待を担うデータベースマシンのサポート言語は多くの場合関係代数である[2]．したがって，関係論理で表現した検索を，データベースマシンでの実行を想定して，最適な関係代数表現に変換する研究が重要になる．

集約関数を除外した条件下での関係論理／関係代数変換法については既に報告した[3]．論理式分類を利用し，網羅性確保のための基本変換則に，特定のパターンに反応し特に効率的な変換を行う発見的変換則を加え，関係論理の再帰構造に沿って逐次的に変換を進めることにより，記述力等値性証明アルゴリズム[4),5]ではとても望めない簡潔な関係代数表現を生成できる．

集約関数の導入により変換は著しく困難になる．それは，今までアルファはすべて閉じていたのに対し，開アルファが現れるようになるためである．開アルファ中にはその外側で変域が規定された変数がある．閉アルファだけから成る閉アルファは基本的には内側の閉アルファから順に変換していけばよいが，開アルファを含む場合にはその方法が採れなくなり，変換の基本的な見通しが失われる．

Klug[6]は関係論理と関係代数に集約関数を導入し，両者の記述力が同等であることを証明した．関係論理表現から関係代数表現への変換（逆は簡単）に当たり，彼は関係論理表現の構成要素（項，論理式，アルファ）ごとに関係代数表現を生成する方法を提案した．そのため変換結果の関係代数表現は自らも認めるように極めて煩雑となり，見直しが必要とされていた．なお，彼は同論文の中で，集約関数を変換する際に補正が必要な場合があることを指摘した．これを本論文ではKlugの補正と呼ぶ．

また，Kim[7]はSQLを最適な正準形に変換する研究の中で集約関数を扱っているが，正当性，最適化の面で十分でない．すなわち，Klugの補正を考慮しておらず，また，唯一つのパターンに対する変換則（本論文の変換則AB2に相当）を示しているにすぎない．

本論文では，関係論理表現に現れる集約関数をKlugの補正も考慮に入れた最適な関係代数表現に変換する方法を述べる．新変換法の基本的アイデアは，まず集約関数を含むアルファを標準集約形と呼ぶ特殊なアルファに変換し，次いでそれを解決することにより，集約関数抜きの関係論理／関係代数変換法[3]に帰着するというものである．以下，まず関係論理と関係代数に集約関数を導入する．その際，Klugの補正に巧妙に対処できる新集約演算を提案する．次いで，新変換法の概要を述べ，それを構成する基本変換則，発見的変換則の詳細を説明する．

なお，例題は以下のスキーマを使用する．

emp（氏名，給料，店長，店）
sales（店，商品，数量）
supply（会社，店，商品，数量）
class（商品，型）

† Reduction of Aggregate Functions in Relational Calculus Alpha to Optimal Algebraic Expressions by RYOHEI NAKANO and KAZUMI SAITOH (Knowledge Systems Laboratory, NTT Communications and Information Processing Laboratories).

†† NTT 情報通信処理研究所知識処理研究部

loc（店，階）

2. 集約関数の導入

2.1 関係論理の規定

Klug[6] は Codd[4] の関係論理を基に，範囲式の分離記法，集約関数の導入，範囲式定義の拡張を行った．本論文は Klug の関係論理に準ずる．関係論理の構文規定を**図1**に示す．

アルファは関係論理の基本要素であり，評価すれば派生関係となる．アルファは，目標リスト，範囲規定，および条件式から成り，コロンをデリミタとして，

目標リスト：範囲規定：条件式

の形式とする．範囲規定は1個以上の範囲式をカンマで区切ったものである．より具体的なアルファの形式は，t_i を項，R_j を範囲，u_j を組変数，f を条件式とするとき，以下である．

$(t_1, \cdots, t_n) : R_1(u_1), \cdots, R_m(u_m) : f$

アルファに現れるどの組変数もそのアルファ中の範囲式で規定されているとき，そのアルファを閉アルファと呼ぶ．閉アルファでないアルファを開アルファと呼ぶ．検索文を表す全体のアルファは必ず閉アルファである．

さて，集約関数を関係論理に導入するには項の一つとして目標リストや条件式に書けるようにすればよい．多くの場合，例1のように，集約の対象となるアルファ（集約対象アルファと呼ぶ）は開アルファである．例1で最も外側のアルファの範囲を関係 class としたのは，供給された全商品が登録されているからで，その中には現在供給されていない商品も含む．また，関係 supply は現在の供給状態を表すものとする．

《例1》 各商品の供給会社数を求めよ．

$(u[1], \text{count}((v[1]) : \text{supply}(v) : v[3] = u[1]))$
$: \text{class}(u) : ( )$

スキーマの任意のインスタンス I の下での閉アルファ $(t_1, \cdots, t_n) : R_1(u_1), \cdots, R_m(u_m) : f$ の意味規定，すなわち，評価法を示す．これは以降の証明で使用する．

$\{(t_1, \cdots, t_n) | u_1 \in R_1(I) \wedge \cdots \wedge u_m \in R_m(I) \wedge$
$f(u_1, \cdots, u_m) = 1\}$

2.2 関係代数の規定

本論文の関係代数は Codd の提案[4]を基本に，選択条件の論理式化，準結合の導入，および複数属性対による結合／準結合／除算の拡張を行ったものとする．関係代数の構文規定を**図2**に示す．関係代数の演算は，

```
〈アルファ〉::= 〈関係スキーム〉
              | 〈目標リスト〉:〈範囲規定〉:〈条件式〉
〈関係スキーム〉::= 〈文字列〉
〈目標リスト〉::= (〈目標要素〉)
〈目標要素〉::= 〈項〉 | 〈組変数〉 | 〈目標要素〉, 〈目標要素〉
〈項〉::= 〈定数〉 | 〈組変数〉 [〈属性番号〉] | 〈集約関数〉
〈組変数〉::= 〈英小文字〉
〈範囲規定〉::= 〈範囲式〉 | 〈範囲式〉, 〈範囲規定〉
〈範囲式〉::= 〈範囲〉 (〈組変数〉)
〈範囲〉::= 〈範囲項〉 | 〈範囲〉∨〈範囲項〉
〈範囲項〉::= 〈範囲因〉 | 〈範囲項〉∧〈範囲因〉
〈範囲因〉::= 〈範囲元〉 | 〈範囲因〉∧～〈範囲元〉
〈範囲元〉::= 〈アルファ〉 | (〈範囲〉)
〈条件式〉::= 〈ブール項〉 | 〈条件式〉∨〈ブール項〉
〈ブール項〉::= 〈ブール因〉 | 〈ブール項〉∧〈ブール因〉
〈ブール因〉::= 〈ブール元〉 | ～〈ブール元〉
〈ブール元〉::= 〈原子式〉 | (〈条件式〉)
              | 〈限量子〉〈範囲式〉〈条件式〉
〈原子式〉::= 〈項〉〈比較演算子〉〈項〉
〈比較演算子〉::= = | <> | < | <= | > | >=
〈限量子〉::= ∃ | ∀
〈集約関数〉::= 〈集約関数名1〉 (〈アルファ〉)
              | 〈集約関数名2〉[〈属性番号〉] (〈アルファ〉)
〈集約関数名1〉::= count
〈集約関数名2〉::= min | max | sum | avg
```

図1 関係論理の構文規定
Fig. 1 Syntax of relational calculus.

射影，選択，結合，準結合，除算，集合演算（和集合，共通集合，差集合），集約演算1，集約演算2から成る．属性は関係ごとに1から振られた属性番号で識別する．定数と区別するため，属性番号にはシャープ記号（#）を冠して表記する．なお，射影，結合，除算，または集約演算を施してできる派生関係の属性並びは，対象となった関係のものとは全く異なるので，属性番号は改めて1から採番する．

集約関数の関係代数演算を集約演算と呼ぶ．集約演算として2種用意する．集約演算1は Klug の提案で，集約演算2は本論文の新提案である．

集約演算2の使用法を例1を用いて説明する．例1の関係論理表現と等価な関係代数表現は一見以下で良いように思える．

((class[#1]) [#1=#2] (supply[#1, #3]))
〈#1 ; count〉

しかし，この表現では，現在供給中でない商品，すなわち関係 class にはあるが関係 supply にない商品，が結合演算時に欠落する．正しくは，供給会社数が0の商品の項を次のように追加すべきである．

```
〈関係代数表現〉::= 〈関係スキーム〉|〈関係代数表現〉〈単項演算〉
                  |〈関係代数表現〉〈二項演算〉〈関係代数表現〉
                  |(〈関係代数表現〉)
〈単項演算〉::= 〈射影〉|〈選択〉|〈集約演算1〉
〈二項演算〉::= 〈結合〉|〈準結合〉|〈除算〉|〈集合演算〉
              |〈集約演算2〉
〈射影〉::= [〈属性リスト〉]
〈属性リスト〉::= 〈属性〉|〈属性〉,〈属性リスト〉
〈属性〉::= #〈属性番号〉
〈選択〉::= [〈選択条件式〉]
〈選択条件式〉::= 〈選択項〉|〈選択条件式〉∨〈選択項〉
〈選択項〉::= 〈選択元〉|〈選択項〉∧〈選択元〉
〈選択元〉::= 〈選択原子式〉|(〈選択条件式〉)
〈選択原子式〉::= 〈属性〉〈比較演算子〉〈定数〉
                |〈属性〉〈比較演算子〉〈属性〉
〈結合〉::= [〈結合条件式〉]|[]
〈結合条件式〉::= 〈結合述語〉|〈結合条件式〉,〈結合述語〉
〈結合述語〉::= 〈属性〉〈比較演算子〉〈属性〉
〈準結合〉::= [∋;〈結合条件式〉]|[～∋;〈結合条件式〉]
〈除算〉::= [(〈属性リスト〉)／(〈属性リスト〉)]
〈集合演算〉::= [+]|[*]|[-]
〈集約演算1〉::= (〈関数リスト〉)
                |(〈グループ化属性〉;〈関数リスト〉)
〈集約演算2〉::= [〈グループ化属性〉;〈関数リスト〉]
〈グループ化属性〉::= 〈属性リスト〉
〈関数リスト〉::= 〈関数〉|〈関数リスト〉,〈関数〉
〈関数〉::= 〈集約関数名1〉|〈集約関数名2〉[#〈属性番号〉]
```

図 2 関係代数の構文規定
Fig. 2 Syntax of relational algebra.

```
((class[#1]) [#1=#2] (supply[#1, #3]))
  〈#1 ; count〉
  [+] ((class[#1] [-] supply[#3]) [ ] {0})
```

和集合の後半が Klug の補正項である．グループ化のベースとなる属性値集合（例では class [#1]）が集約時の振り分けキーとなる属性値集合（例では supply [#3]）に含まれないとき，両者を等結合すると共通部分しか残らないので，グループ化のベースとなる属性値集合の一部が欠落する．それを補うのが Klug の補正である．

一方，集約演算2は Klug の補正に巧妙に対応できる．例1はこれを用いると以下のように簡潔に書ける．

```
supply [#1, #3] [#2 ; count] (class[#1])
```

商品集合 class [#1] をグループ化のベースとし，そこへ supply [#1, #3] [#2] をキーに振り分けを行い，その数をカウントする．グループ化のベースは集約対象アルファが空 (ϕ) でカウント＝0 でも消えないので，供給会社数が0の商品情報も正しく残ることになり，例1の関係論理表現と同じ結果を得る．

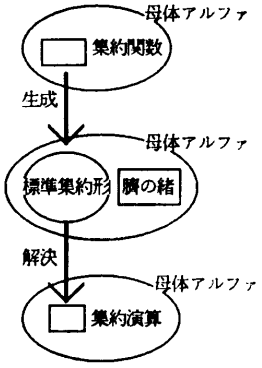

図 3 集約関数変換の概略フロー
Fig. 3 General flow of aggregate function reduction.

集約演算1はグループ化のベースとなる属性値集合と集約時の振り分けキーとなる属性値集合が同一の場合であり，集約演算2の特殊ケースである．

集約演算の意味規定を以下に述べる．今，fun を集約関数，e_1, e_2 を関係代数表現，A をグループ化属性，I をスキーマの任意のインスタンスとするとき，

$$(e_1\langle A\,;\,\mathrm{fun}\rangle)(I)=\{(u[A],\ f):u\in e_1(I)\wedge f=\mathrm{fun}(\{v:v\in e_1(I)\wedge v[A]=u[A]\})\}$$

$$(e_1[A\,;\,\mathrm{fun}]e_2)(I)=\{(w,f):w\in e_2(I)\wedge f=\mathrm{fun}(\{v:v\in e_1(I)\wedge v[A]=w\})\}$$

ただし，count(ϕ)=0，count 以外の fun(ϕ)＝空値

3. 集約関数の新変換法

3.1 新変換法の概要

本論文で提案する集約関数変換法の基本的なアイデアは，集約関数を目標リストに含む特殊なアルファ（標準集約形）を考えて，それを変換の中継地点とすることにある[8]．すなわち，まず一般の集約関数を含むアルファ（母体アルファ）を標準集約形を含むアルファに変形し，その後，標準集約形を集約演算を用いて解決するという2フェーズの変換過程をとる（図 3）．もちろん，母体アルファが始めから標準集約形の場合には，第2フェーズのみを実行すればよい．

標準集約形とは，集約関数が目標リストに現れ，目標リストが集約対象アルファ中の開属性の集合とその集約関数のみで構成され，かつ条件式が空のアルファである．

標準集約形の生成時，一般には，その標準集約形と母体アルファを結ぶ臍の緒も生成する．臍の緒とは標

準集約形と母体アルファとの間の属性間の対応関係を表した等結合型原子式または等結合型論理式である．

新変換法の変換過程を具体例で示す．

《例2》 店員の氏名，給与，および所属店の平均給与を求めよ．

$(u[1], u[2], \mathrm{avg}[2]\ ((v) : \mathrm{emp}(v) : v[4]=u[4]))$
$: \mathrm{emp}(u) : (\ )$

集約対象アルファは $((v) : \mathrm{emp}(v) : v[4]=u[4])$ である．第1フェーズを実行すると，標準集約形 S，および臍の緒 $s[1]=u[4]$ が生成され，母体アルファは以下となる．

$(u[1], u[2], s[2]) : \mathrm{emp}(u), S(s) : s[1]=u[4]$

ただし，

$S=(w[4], \mathrm{avg}[2]\ ((v) : \mathrm{emp}(v) : v[4]=w[4]))$
$: \mathrm{emp}(w) : (\ )$

第2フェーズを実行すると以下のように集約関数が解決される．この後は，集約関数抜きの変換法を適用すればよい．

$(u[1], u[2], s[2]) : \mathrm{emp}(u),$
$(\mathrm{emp}\ \langle \#4 ; \mathrm{avg}\ [2]\rangle)\ (s) : s[1]=u[4]$

変換の詳細フローを**図4**に示す．最初に，基本変換を説明する．関係論理表現の再帰構造に沿って変換を進めて行く中で，集約関数に出会ったとする．その場所は現在変換対象であるアルファの目標リストか条件式のいずれかである．いずれも基本的には標準集約形と臍の緒を生成するが，目標リストの場合は範囲として追加する（変換則 AB1）のに対し，条件式の場合は準結合の形とする（変換則 AB2）のが異なる．標準集約形は集約演算2に直結するアルファなので見通し良く変換できる（変換則 AB3）．

次に，発見的変換は，上記基本変換において特定の条件を満たす場合に，より簡素で効率的な変換を行う．発見的変換のポイントは2点ある．すなわち，Klug の補正省略と結合省略（AJ）である．Klug の補正省略には，集約対象アルファが空になることがあっても問題でないケース（AK1）と，空にならないケース（AK2）がある．発見的変換則の成立条件は排他的ではないので，AK1 & J，AK2 & J のように相乗して成立することも考慮する必要がある．なお，スルーは始めから標準集約形であったため何もしない場合である．

3.2 集約関数の基本変換

集約関数を含む関係論理表現の変換可能性に関し，以下の定理が成立する[6]．

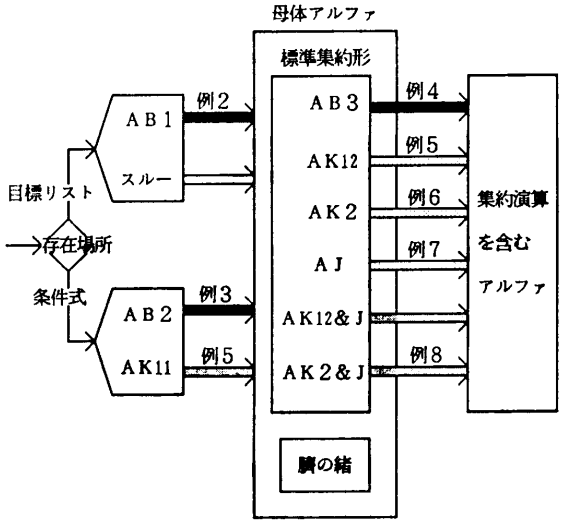

図4 集約関数変換の詳細フロー
Fig. 4 Detail flow of aggregate function reduction.

［変換定理］ 2.1節で規定した関係論理で表現した任意の閉アルファは2.2節で規定した関係代数表現に変換できる．

（証明） 集約関数抜きの関係論理で表現した任意の閉アルファが関係代数表現に変換できることは証明済みとする[3]．したがって，ここでは集約関数を含む任意の関係論理表現が集約演算（関係代数演算）と集約関数抜きの関係論理表現に変換できることを示せばよい．

最初に，集約関数を1個のみ含む任意の関係論理表現を考える．集約関数は項の一種であるから，必ず目標リストまたは条件式に現れる．目標リストに現れた場合は変換則 AB1 により，また条件式の場合は変換則 AB2 により，各々，標準集約形および臍の緒を追加した母体アルファに変換できる．臍の緒を追加した母体アルファは集約関数抜きの関係論理表現であり，一方，標準集約形は変換則 AB3 により集約演算2と集約関数抜きの関係論理表現に変換できる．ゆえに，集約関数を1個だけ含む任意の関係論理表現は集約演算と集約関数抜きの関係論理表現に変換できる．

次に，集約関数が多数存在する場合は，外側のものから一つずつ変換して行くことにより，最終的には集約関数をすべて解消できる． （証明終）

以降の定式化で使用する記号の意味を**表1**に示す．なお，変換則は左辺から右辺へ変換する．

3.2.1 標準集約形の生成—1

集約関数の最初の基本変換則は目標リスト中の集約

表 1 記号の説明
Table 1 Notation of symbols.

記 号	意 味
{ }	集合
I	スキーマの任意のインスタンス
len (L)	リストLの要素数
α	アルファ
tl	目標リスト
$X(\chi), \Psi(\psi), \cdots$	範囲規定，ただし，X, Ψ は範囲ベクトル，χ, ψ は対応する組変数ベクトル
$x, y, \cdots$	組変数
$f, g, \cdots$	論理式
$f_e, \cdots$	等結合型単位式の論理積から成る論理式
$f_j, \cdots$	結合型単位式の論理積から成る論理式

関数を対象とし，それを基に標準集約形と臍の緒を生成し，各々，母体アルファの範囲規定，条件式に追加する．臍の緒は集約対象アルファ中の開属性と標準集約形の属性とを対応付ける論理式である．なお，母体アルファの条件式は変換によって既に空であるとする．例2の前半が本変換則の適用例である．

[AB 1: 集約関数基本—1]

$((tl, \mathrm{fun}(\alpha(\chi))) : X(\chi), \Psi(\phi) : (\,)) \ (I)$

$= ((tl, s[n+1]) : X(\chi), \Psi(\phi), S(s) : f_e(s, \chi))(I)$

ただし，

- 標準集約形 $S = (\chi_a, \mathrm{fun}(\alpha)) : X(\chi) : (\,)$
- 臍の緒 $f_e(s, \chi) = (s[1] = \chi_a[1] \wedge \cdots \wedge s[n] = \chi_a[n])$
- $n = \mathrm{len}(\chi_a)$

(証明) 左辺

$= ((tl, \mathrm{fun}(\alpha(\chi))) : X(\chi), \Psi(\phi) : (\,)) \ (I)$

$= \{(tl, \mathrm{fun}(\alpha(\chi))) \mid \chi \in X(I) \wedge \phi \in \Psi(I)\}$

$= \{(tl, \mathrm{fun}(\alpha(\chi'))) \mid \chi \in X(I) \wedge \phi \in \Psi(I) \wedge \chi' \in X(I) \wedge \chi'_a = \chi_a\}$

$= \{(tl, s[n+1]) \mid \chi \in X(I) \wedge \phi \in \Psi(I) \wedge s \in \{(\chi'_a, \mathrm{fun}(\alpha(\chi'))) \mid \chi' \in X(I)\} \wedge f_e(s, \chi) = 1\}$

$= ((tl, s[n+1]) : X(\chi), \Psi(\phi), S(s) : f_e(s, \chi))(I)$

= 右辺 (証明終)

3.2.2 標準集約形の生成—2

集約関数が条件式中にあるときには，例3のように標準集約形を準結合の形で組み込む．これは，条件式中の集約関数が母体アルファに対し選択的に効く性質を反映したものである．準結合は結合より負荷が軽いので実行上有利である．

《例3》 給料が自分の店の平均より多い店員を求めよ．

$(u[1]) : \mathrm{emp}(u) : u[2] > \mathrm{avg}[2] \ ((v) : \mathrm{emp}(v) : v[4] = u[4])$

これは次のように標準集約形 S と臍の緒 $s[1] = u[4]$ を含む表現に変形できる．

$(u[1]) : \mathrm{emp}(u) : {}^{\exists}S(s) \ (u[2] > s[2] \wedge s[1] = u[4])$

ただし，

$S = (u[4], \mathrm{avg}[2] \ ((v) : \mathrm{emp}(v) : v[4] = u[4])) : \mathrm{emp}(u) : (\,)$

[AB 2: 集約関数基本—2]

$(tl : X(\chi), \Psi(\phi) : (\mathrm{fun}(\alpha(\chi))\theta t) \wedge g) \ (I)$

$= (tl : X(\chi), \Psi(\phi) : ({}^{\exists}S(s) \ (s[n+1]\theta t \wedge f_e(s, \chi))) \wedge g) \ (I)$

ただし，

- $S = (\chi_a, \mathrm{fun}(\alpha)) : X(\chi) : (\,)$
- $f_e(s, \chi) = (s[1] = \chi_a[1] \wedge \cdots \wedge s[n] = \chi_a[n])$
- $n = \mathrm{len}(\chi_a)$

(証明) 左辺

$= (tl : X(\chi), \Psi(\phi) : (\mathrm{fun}(\alpha(\chi))\theta t) \wedge g) \ (I)$

$= \{tl \mid \chi \in X(I) \wedge \phi \in \Psi(I) \wedge (\mathrm{fun}(\alpha(\chi))\theta t) = 1 \wedge g = 1\}$

$= \{tl \mid \chi \in X(I) \wedge \phi \in \Psi(I) \wedge \chi' \in X(I) \wedge \chi'_a = \chi_a \wedge (\mathrm{fun}(\alpha(\chi'))\theta t) = 1 \wedge g = 1\}$

$= \{tl \mid \chi \in X(I) \wedge \phi \in \Psi(I) \wedge \{s \mid s \in S(I) \wedge f(s, \chi) = 1 \wedge (s[n+1]\theta t) = 1\} \neq \phi \wedge g = 1\}$

$= (tl : X(\chi), \Psi(\phi) : ({}^{\exists}S(s) \ (s[n+1]\theta t \wedge f_e(s, \chi))) \wedge g) \ (I)$

= 右辺 (証明終)

3.2.3 標準集約形の解決

集約関数の3番目の基本変換則は標準集約形を解決するものである．標準集約形を解決するには，集約関数を取り去ったアルファをグループ化のベースとし，集約対象アルファと母体アルファを一つに合併したアルファを集約の対象とする集約演算2に変換すればよい．これにより，Klug の補正も対処できる．

《例4》 階ごとに，その階より上にある店の数を求めよ．

$(u[2], \mathrm{count}((v[1]) : \mathrm{loc}(v) : v[2] > u[2])) : \mathrm{loc}(u) : (\,)$

これは集約演算2を用いた以下の表現と等価である．

$((u[2], v[1]) : \mathrm{loc}(u), \mathrm{loc}(v) : v[2] > u[2]) \ [\#1 ; \mathrm{count}] \ ((u[2]) : \mathrm{loc}(u) : (\,))$

[AB 3: 集約関数基本—3]

$((tl_1, \mathrm{fun}(tl_2 : \Omega(\omega) : g)) : X(\chi) : (\,)) \ (I)$

$= (((tl_1, tl_2) : X(\chi), \Omega(\omega) : g) \ [\#1, \cdots, \#n ; \mathrm{fun}'] \ (tl_1 : X(\chi) : (\,))) \ (I)$

ただし，

• $n=\mathrm{len}(tl_1)$
• fun′ は fun において集約属性番号に n を加えたもの
• $\chi_\sigma=tl_1$（標準集約形の条件）

（証明）　左辺

$=((tl_1,\ \mathrm{fun}(tl_2:\Omega(\omega):g)):X(\chi):(\ ))\ (I)$

$=\{(tl_1,\ \mathrm{fun}(\{tl_2:\omega\in\Omega(I)\wedge g=1\})):\chi\in X(I)\}$

条件 $\chi_\sigma=tl_1$ より，集約対象アルファが変形できて，

$=\{(tl_1(\chi),\ \mathrm{fun}'(\{s|s\in S(I)\wedge s[1,\cdots,n]=tl_1(\chi)\}))|\chi\in X(I)\}$

ただし，$S(I)=\{(tl_1(\chi'),\ tl_2(\omega))|\chi'\in X(I)\wedge\omega\in\Omega(I)\wedge g(\chi',\omega)=1\}$

$=\{(w,\mathrm{fun}'(\{s|s\in S(I)\wedge s[1,\cdots,n]=w\}))|w\in\{tl_1|\chi\in X(I)\}\}$

$=\{(w,f)|w\in\{tl_1|\chi\in X(I)\}\wedge f=\mathrm{fun}'(\{s|s\in S(I)\wedge s[1,\cdots,n]=w\})\}$

集約演算 2 が適用できて，

$=(((tl_1,\ tl_2):X(\chi),\ \Omega(\omega):g)\ [\#1,\cdots,\#n\ ;\mathrm{fun}']\ (tl_1:X(\chi):(\ )))\ (I)$

＝右辺　（証明終）

3.3 集約関数の発見的変換

前述の変換定理は集約関数の基本変換則のみを用いて証明した．すなわち，基本変換だけでも集約関数変換は完結するが，発見的変換を導入すれば，より簡潔な関係代数表現が生成できる．以下の発見的変換則は基本変換則による変換体系を乱すものでないので，基本変換則に加えても定理が成立することは明らかである．

3.3.1 Klug の補正省略（1）

Klug の補正省略の第 1 のケースは集約対象アルファが空になっても問題にならない場合である．集約関数 count が条件式中に現れ，かつ，count＞0 を満たすとき Klug の補正は省略できる．なぜなら，Klug の補正とは count＝0 になり抜け落ちるケースを補うものであるが，せっかく補ってもしょせん，条件を満たさないからである．したがって，この場合は集約演算 1 でよい．なお，この省略の成立は第 1 フェーズ（標準集約形の生成時）に分かるが，実際に集約演算 1 に変換するのは第 2 フェーズ（標準集約形の解決時）であるため 2 フェーズに渡る処理となる（AK11 と AK12）．標準集約形は Klug の補正不要の情報を表現できないので，集約関数名 count を一時的に pcount に置き換えることでフェーズ間の引き継ぎが可能となる．以下の変換則では，AK11 と AK12 をまとめて AK1 として記す．

《例 5》　3 か所以上の階で売られている商品を求めよ．

$(u[2]):\mathrm{sales}(u):\mathrm{count}((v[2]):\mathrm{loc}(v):{}^{3}\mathrm{sales}(w)\ (w[1]=v[1]\wedge w[2]=u[2]))\rangle=3$

例では count＞＝3 であるので，Klug の補正は省略でき，変換則 AK1 が適用でき，その前半により，

$(u[2]):\mathrm{sales}(u):{}^{3}S(s)(s[2]\rangle=3\wedge s[1]=u[2])$

ただし，$S=(u[2],\ \mathrm{pcount}((v[2]):\mathrm{loc}(v):{}^{3}\mathrm{sales}(w)\ (w[1]=v[1]\wedge w[2]=u[2]))):\mathrm{sales}(u):(\ )$

標準集約形 S に変換則 AK1 の後半を適用して，pcount を集約演算 1 に変換する．

$S=((u[2],\ v[2]):\mathrm{sales}(u),\ \mathrm{loc}(v):{}^{3}\mathrm{sales}(w)\ (w[1]=v[1]\wedge w[2]=u[2]))\ \langle\#1\ ;\mathrm{count}\rangle$

[AK1：Klug の補正省略—1]

$(tl_1:X(\chi),\ \Psi(\phi):(\mathrm{count}(tl_2:\Omega(\omega):g(\chi,\omega))\theta c)\wedge h)\ (I)$

$=(tl_1:X(\chi),\ \Psi(\phi):({}^{3}S(s)(s[n+1]\theta c\wedge f_\sigma(s,\chi)))\wedge h)\ (I)$

ただし，

• θc が $\rangle 0$ を満たすこと（ex.$\rangle 2,\rangle=4,=3,\neq 0$）
• $S=((\chi_\sigma,\ tl_2):X(\chi),\ \Omega(\omega):g)\ \langle\#1,\cdots,\#n\ ;\mathrm{count}\rangle$
• $f_\sigma(s,\chi)=(s[1]=\chi_\sigma[1]\wedge\cdots\wedge s[n]=\chi_\sigma[n])$
• $n=\mathrm{len}(\chi_\sigma)$

（証明）　左辺

$=(tl_1:X(\chi),\ \Psi(\phi):(\mathrm{count}(tl_2:\Omega(\omega):g(\chi,\omega))\theta c)\wedge h)\ (I)$

変換則 AB2 を適用して，

$=(tl_1:X(\chi),\ \Psi(\phi):({}^{3}S(s)(s[n+1]\theta c\wedge f_\sigma(s,\chi)))\wedge h)\ (I)$

ただし，

$S=(\chi_\nu,\ \mathrm{count}(tl_2:\Omega(\omega):g)):X(\chi):(\ )$

ここまでを変換則 AK11 とする．標準集約形 S に変換則 AB3 を適用する．

$S=((\chi_\sigma,\ tl_2):X(\chi),\ \Omega(\omega):g)\ [\#1,\cdots,\#n\ ;\mathrm{count}]\ (\chi_\sigma:X(\chi):(\ ))$

今，θc が＞0 を満たすので，Klug の補正は不要で，

$S=((\chi_\sigma,\ tl_2):X(\chi),\ \Omega(\omega):g)\ \langle\#1,\cdots,\#n\ ;\mathrm{count}\rangle$

（証明終）

3.3.2 Klug の補正省略（2）

Klug の補正省略の第 2 のケースは集約対象アルファが空にならない場合である．集約対象アルファが空にならなければ，集約演算 1 を用いることができる．集約対象アルファの条件式中のどの開属性も閉属性と

の間で等号含みの単位式を構成し，さらにその単位式が論理積で結ばれ，かつ，母体アルファと集約対象アルファの範囲が同一かまたは前者が後者の選択型の場合には，空アルファは生じない．

《例6》 階ごとにその階以上の階にある店の数を求めよ．

$(u[2],\ \mathrm{count}((v[1]):\mathrm{loc}(v):v[2]\rangle=u[2])):$
$\mathrm{loc}(u):(\ ))$

上記表現には変換則 AB3 が適用できるが，グループ化属性 $u[2]$ に属す値は $v[1]$ を空にすることはないので，集約演算1が適用できる．すなわち，以下となる．

$((u[2],\ v[1]):\mathrm{loc}(u),\ \mathrm{loc}(v):v[2]\rangle=u[2])$
$\langle\#1;\ \mathrm{count}\rangle$

[AK2: Klug の補正省略—2]

$((tl_1,\ \mathrm{fun}(tl_2:\Omega(\omega):g_j(\chi,\omega))):X(\chi):(\ ))\ (I)$
$=(((tl_1,\ tl_2):X(\chi),\ \Omega(\omega):g_j(\chi,\omega))$
$\langle\#1,\cdots,\#n;\mathrm{fun}'\rangle)\ (I)$

ただし，

- $g_j(\chi,\omega)=\cdots\wedge\chi[i]\theta\omega[j]\wedge\cdots$，
  ここで，θ は $=$，$\langle=$，$\rangle=$ のいずれか
- $X=\Omega$ または $X=\Omega[\mathrm{sel}]$
- $n=\mathrm{len}(tl_1)$
- fun′ は fun において集約属性番号に n を加えたもの
- $\chi_\theta=tl_1$（標準集約形の条件）

（証明） 範囲に関する条件（$X=\Omega$ または $X=\Omega[\mathrm{sel}]$）と $g_j(\chi,\omega)$ に関する条件から，tl_1 のどの値に対しても集約対象アルファは空にならず，Klug の補正が不要となる．そこで，変換則 AB3 において集約演算2を集約演算1に置換すれば右辺が得られる．

（証明終）

3.3.3 結合省略

集約関数の結合省略は，標準集約形において集約対象アルファの条件式が特殊な場合に成立する．すなわち，集約対象アルファの条件式中のどの開属性も閉属性と等号で結ばれ，かつ，その単位式が論理積で結ばれている場合に成立する．結合が省略できるのは，実は集約演算2でグループ化のベースとなる属性と集約時の振り分けキーとなる属性との間で等結合相当の処理が行われるためである．

《例7》 現在供給されている商品の販売店数を求めよ．

$(u[3],\ \mathrm{count}((v[1]):\mathrm{sales}(v):v[2]=u[3]))$
$:\mathrm{supply}(u):(\ )$

これには変換則 AB3 が適用できるが，しょせん，集約演算2で supply[3] と sales[2] が照合されるので，グループ化される側で事前に等結合する必要はない．すなわち，

$((v[2],\ v[1]):\mathrm{sales}(v):(\ ))\ [\#1;\ \mathrm{count}]$
$((u[3]):\mathrm{supply}(u):(\ ))$

[AJ: 結合省略]

$((tl_1,\ \mathrm{fun}(tl_2:\Omega(\omega):g_\theta(\chi,\omega)\wedge h)):X(\chi):(\ ))(I)$
$=(((\omega_\theta,\ tl_2):\Omega(\omega):h)\ [\#1,\cdots,\#n;\mathrm{fun}']$
$(tl_1:X(\chi):(\ )))\ (I)$

ただし，

- $g_\theta(\chi,\omega)$ は $\chi[k]=\omega[j]$ を論理積で結んだ論理式
- $h=h(\omega)$
- $n=\mathrm{len}(tl_1)$
- fun′ は fun において集約属性番号に n を加えたもの
- $\chi_\theta=tl_1$（標準集約形の条件）

（証明） 変換則 AB3 との違いから以下が成立すればよい．

$((tl_1,\ tl_2):X(\chi),\ \Omega(\omega):g_\theta(\chi,\omega)\wedge h)$
$=((\omega_\theta,\ tl_2):\Omega(\omega):h)$

g_θ と h の条件から，しょせん，集約演算2で突き合わせがなされるので，グループ化される側で結合しなくてよいことは明らかで，上式が成立する．

（証明終）

3.3.4 発見的変換則の相乗

集約関数の発見的変換則は相乗して成立することがある．AK1 & J，または AK2 & J がそれである．前者は単に AK1 と AJ の各条件が同時に成立すればよいのに対し，後者は AK2 と AJ の各条件の論理積より厳しい条件が必要になる．その違いは Klug の補正不要の背景が異なるためで，前者では抜け落ちるケースがあっても構わないのに対し，後者では抜け落ちるケースの発生を抑えながら結合省略を可能にしなければならないためである．以下では後者を記すが，前者もほぼ同様である．

《例8》 各店に供給されている商品の種類数を求めよ．

$(u[2],\ \mathrm{count}(v[3]:\mathrm{supply}(v):v[2]=u[2]))$
$:\mathrm{supply}(u):(\ )$

上記標準集約形には AK2 & J が適用でき以下となる．

$((v[2],\ v[3]):\mathrm{supply}(v):(\ ))\ \langle\#1;\ \mathrm{count}\rangle$

[AK 2 & J: 相乗変換則—1]

$((tl_1, \mathrm{fun}(tl_2 : X(\chi') : g_e(\chi, \chi'))) : X(\chi) : (\,)) \ (I)$

$= (((\chi_e, tl_2) : X(\chi) : (\,)) \langle \#1, \cdots, \#n ; \mathrm{fun}' \rangle) \ (I)$

ただし,

- $g_e(\chi, \chi')$ は $\chi[j]=\chi'[j]$ を論理積で結んだ論理式
- $n=\mathrm{len}(tl_1)$
- fun′ は fun において集約属性番号に n を加えたもの
- $\chi_e=tl_1$ (標準集約形の条件)

(証明) 左辺

$= ((tl_1, \mathrm{fun}(tl_2 : X(\chi') : g_e(\chi, \chi'))) : X(\chi) : (\,)) \ (I)$

範囲条件, 論理式条件から変換則 AK 2 が適用でき,

$= (((tl_1, tl_2) : X(\chi), X(\chi') : g_e(\chi, \chi')) \langle \#1, \cdots, \#n ; \mathrm{fun}' \rangle) \ (I)$

条件式 g_e から結合が省略でき, さらに $\chi_e=tl_1$ より,

$= (((\chi_e, tl_2) : X(\chi) : (\,)) \langle \#1, \cdots, \#n ; \mathrm{fun}' \rangle) \ (I)$

= 右辺　(証明終)

4. 変 換 例

前節までに述べた変換法が正しく動作するか, さらに当初のねらいどおり人間が作成するような効率的な関係代数表現を生成するかを検証するため, 変換プログラムを作成し検索例を試験した. 変換プログラムは構文解析および記号変換に向く prolog で作成した.

以下に Klug の例題[6]の変換結果を示す. 各例題とも, 最初に変換対象である関係論理表現, 次いで変換結果の関係代数表現を記す.

Klug の例題のスキーマを以下に示す.

student (学生名, 学年, 平均評点)
dept (学部名, 部長名, 大学名)
faculty (教授名, 所属学部名)
grad (院生名, 指導教授名, 助成金額)

《Klug 1》 学年ごとに, 学生の平均評点の平均を求めよ.

- (u[2], avg[3] ((v) : student(v) : v[2]=u[2])) : student(u) : ( )
- student ⟨#2 ; avg[3]⟩

《Klug 2》 学年ごとに, その学年以上の学生の平均評点の平均を求めよ.

- (u[2], avg[3] ((v) : student(v) : v[2]⟩=u[2])) : student(u) : ( )
- (student [#2⟨=#2] student) [#2, #4, #5, #6] ⟨#1 ; avg[4]⟩

《Klug 3》 LS 大学の学部ごとに, 教授 1 人当たりの指導院生の助成金の平均を求めよ.

- (u[1], avg[2] ((v[1], sum[2] ((w[1], w[3]) : grad(w) : w[2]=v[1])) : faculty(v) : v[2]=u[1])) : dept(u) : u[3]=LS
- ((faculty [#1=#1] ((grad [#2, #1, #3]) [#1 ; sum[3]] (faculty [#1]))) [#2, #1, #4]) [#1 ; avg[3]] (dept [#3=1s] [#1])

上記変換結果と人間 (筆者) の作成例とは一点を除いて全く同じであった. 相違箇所は Klug 3 で, 変換結果には並び換えだけの不要な射影 grad [#2, #1, #3] がある. 検索の複雑さを考慮するとき, 変換結果の最適性は満足できる. 他の検索例でも同様の結果が得られた.

5. む す び

集約関数が入って来ると, 閉じないアルファが現れるので, 関係論理から関係代数への変換は容易でなくなる. 本論文では, 関係代数表現への変換が容易な標準集約形を中継地点とし, それの生成と解決という 2 フェーズの変換体系を採った. 集約関数の居場所により標準集約形の生成形も異なる. また, Klug の補正に効率良く対処するため新集約演算を考案し導入した. 標準集約形の生成／解決に関する体系を明らかにし, それを構成する基本変換則と発見的変換則の詳細を述べた. 新変換法の目的は, 集約関数を含んだ関係論理表現から人間が考え出すような最適な関係代数表現を生成することにある. prolog を用いて変換プログラムを作成し, 考えられる様々な複雑な検索に適用し, 極めて満足すべき結果が得られることを確認した. 今後は, 四則への拡張, 空値の及ぼす影響の明確化, および非正規形への拡張等を研究する必要がある.

謝辞 本研究に際し, ご指導いただいた NTT 電気通信研究所の橋本昭洋部長 (基礎研究所情報科学研究部), 村上国男部長 (情報通信処理研究所知識処理研究部), 吉田清グループリーダ, ならびに有益なコメントを頂いた査読委員の方々に感謝いたします.

参 考 文 献

1) ISO/TC97/SC21: Database Language SQL, ISO/TC97/SC21/WG5-15 (1985).
2) Luo, D. et al.: Data Language Requirements of Database Machines, *Proc. NCC*, pp. 617-626 (1982).

3) 中野良平，斉藤和巳：ルールに基づいた関係論理／関係代数変換法，情報処理学会データベースシステム研究会資料，86-DB-54 (1986).
4) Codd, E. F.: Relational Completeness of Database Sublanguages, *Courant Computer Symposium 6*, pp. 65-98, Prentice-Hall, Englewood Cliffs, N. J. (1972).
5) Ullman, J. D.: *Principles of Database Systems*, Second Edition, Computer Science Press, Rockville, Maryland (1982).
6) Klug, A.: Equivalence of Relational Algebra and Relational Calculus Query Languages Having Aggregate Functions, *J. ACM*, Vol. 29, No. 3, pp. 699-717 (1982).
7) Kim, W.: On Optimizing an SQL-like Nested Query, *ACM Trans. Database Syst.*, Vol. 7, No. 3, pp. 443-469 (1982).
8) 中野良平，斉藤和巳：集約関数まで拡張した関係論理／関係代数変換法，情報処理学会データベースシステム研究会資料，87-DB-57 (1987).

(昭和62年2月18日受付)
(昭和62年10月14日採録)

中野 良平（正会員）

昭和22年生．昭和46年東京大学工学部計数工学科卒業．同年日本電信電話公社（現NTT）入社．以来，統計解析，分散処理，データベース，知識処理の研究に従事．現在NTT情報通信処理研究所主幹研究員．人工知能学会，ACM，IEEE各会員．

斉藤 和巳（正会員）

昭和38年生．昭和60年慶応義塾大学理工学部数理科学科卒業．同年日本電信電話(株)入社．現在，同情報通信処理研究所にて知識処理の研究に従事．人工知能学会会員．