データベースシステム 108－9
（1996．5．23）

データベーススキーマの効果的な図示手法

石川 英彦　有澤 博
横浜国立大学 工学部 電子情報工学科

マルチメディアデータを始めとする複雑な構造を持つデータを表現するためには，主体と関連のみならず，主体集合間の包含などの関係を表現することが必要である．主体集合間の関係を表現するための手法は様々のものが提案されているが，高い表現力と図示可能性などの直観性を併せ持つものは殆どない．本稿ではABE(Allowance Band Expression)と呼ぶ集合間関係のダイアグラムによる表現方法を提案する．この手法により，設計者は「汎化」「包含」「(反)排他」などの基本要素を用いて直観的にスキーマを設計することが可能となる．また，本方式を実際に用いる際に必要となる検査アルゴリズムを挙げ，その妥当性を示す．

An effective drawing method to represent database schema

Hidehiko Ishikawa　Hiroshi Arisawa
{hidepong,arisawa}@arislab.dnj.ynu.ac.jp
Division of Electrical and Computer Engineering, Yokohama National University

To represent highly structured data such as multimedia data, the method for representing relationships among entity types is essential. However several approaches to represent such relationships have not succeeded to satisfy both intuition and properiety of calculating cost. This paper presents a new graphical drawing method called ABE, which can express the "traditional" relationships among entity types such as "generalization", "distinction", and more complicated ones. We also show this method can be understood by database designers intuitively, and that it can be used with practical calculating cost.

1　主体集合間の関係

ERモデル[1]を始めとする殆んどのデータモデルは，「主体型(Entity Type)」などの，主体集合を表す基本要素を持つ．主体型の概念によって，ある主体がどのようなカテゴリに分類されるかを区別することができるが，一般に各主体型は排他であり，複数の主体型が同一の主体を共有することは考慮されないことが多い．例えば，学生成績データベースにおいて，主体型として「学生」「科目」を，これらの間の関連型として「履修」を設けた場合，一つの主体は「学生」か「科目」か「履修」のいづれか一つにのみ所属し，どの主体型にも所属しない主体や，複数の主体型に同時に所属する主体の存在は許されない．

このような，互いに排他関係にある主体型の概念を用いて，ある程度以上に複雑な情報を表現することは困難である．例えば，人間を職業別に分類する場合，ある人が複数の職に就くことはあり得るが，データベース中ではその人に対応する主体は，複数の職に対応する主体型に所属することはできない．

これを表現するための一つの方法として，例えば「会社員」である主体eと「人間」である主体hの間に，「同一人物である」という関連を設けることが考えられるが，関連のインスタンスが多くなってし

まったり，辿る必要のある経路が長くなったりなどの不都合がある．また，男性と女性のように互いに排他であるような主体型も存在し得るが，排他であるかないかを区別することも困難である．

複雑な構造を持つデータの例としては，以下に示すようなものが挙げられる．

[例 1] 木構造は，基本的に「親 (Parent)」から「子 (Child)」への $1:n$ 関係によって表現されるが，これらの部分集合として，親を持たない「根 (Root)」や，子を持たない「葉 (Leaf)」，どちらも持っている「中間子 (Internal Node)」やどちらも持たない「単独木 (Mono Tree)」などが存在する．

[例 2] 映像データベースにおいて，映像を主体として表現した場合，「作業映像」は「人間の作業映像」「ロボットの作業映像」などに分けられ，それぞれは固有の属性 (「作業人数」など) を持つが，人間とロボットの協調作業の映像など，いくつかの分類に跨る映像が存在し得る．

集合間の関係，特に包含関係について，古くは Smith らによる汎化 (Generalization) [2] の理論があるが，それ以降は設計論としては議論が深められてはいない．

有澤らの IXG(Intersection eXtended Generalization)，UXG(Union eXtended Generalization) および Co-EXD(Co-Exclustion Dependency) [7] や，Lenzerini の CHS(Class Hierarchy Scheme)[8] などは，論理的記述によるアプローチの例である．IXG および UXG は，単純な汎化関係を「積集合」や「和集合」に拡張したもので，Co-EXD は要素が複数の集合に所属することを許す (つまり，複数の集合の交差を許す) 記法である．

これらの記述法をデータベーススキーマの設計手段として実際に使用するためには，いくつかの問題点がある．これらの記述法は命題論理や一階述語論理などの論理体系との対称性を示すことで記述を規則に従って書き換えることで意味 (「学生であり教員であるものは存在できない」，など) を導くことが可能であることを示しているが，スキーマ設計のために必要と思われる要素と一階述語論理等の要素とはうまく対応しない．

また，「集合 A は集合 B に包含される」「集合 A と集合 B は排他である」の両者を満足させるためには，集合 A には一つの要素も所属することはできない．データベースにおいて特定の集合 (主体型) が空でなければならない状態は異常であり，このような状態 (矛盾) を検出し，解消できなければならないが，検出のための計算量が大きくなったりなどの不都合が生じやすい．例えば，Co-EXD と包含関係の概念は直交していないため，両者の間の冗長性を取り除く必要があるし，CHS の場合，その記述を直接図示することは困難である．そのうえ，矛盾を検出するための計算量が P-SPACE 完全であることが示されている．CHS のスキーマの中には矛盾したものも存在し得るが，これを検出するのに必要な計算量が大き過ぎる．また，理論的な計算量に対し，現実のデータベースシステムにおいてどの程度の計算コストが必要とされるかについての考察は不十分であり，実際に CHS をスキーマ設計の手段として採用することはできない．

本稿では，直観的で単純な図的要素を用いて集合間の関係を表現することを提案する．これは，「許容線 (Allowance Band)」と呼ばれる帯によって集合間を結ぶことで包含や排他，反排他の関係を表現しようとするもの (ABE: Allwoance Band Expression) である．また，ABE を使用するために必要なアルゴリズムを挙げるとともに，現実のデータベースシステム上で必要な計算量について考察を行い，現実的な範囲での計算量的妥当性を示す．

次節では ABE の定義を示し，これを用いることで直観的に集合間の関係を設計できることを示す．3 節では ABE によって設計されたスキーマに従って実際にデータベースを制約するためのアルゴリズムを示し，ABE の妥当性を示し，4 節でまとめを行なう．

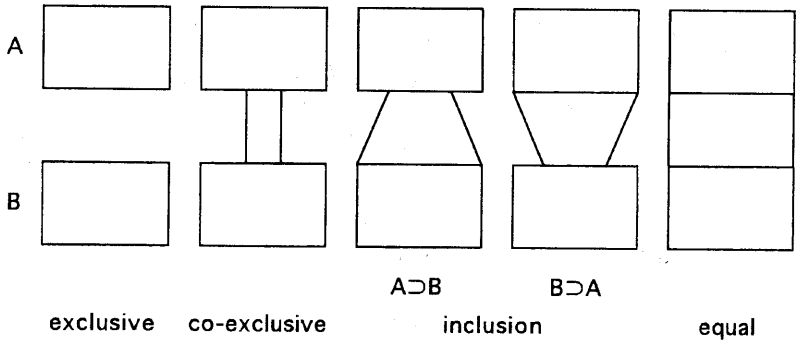

図 1: ABE により表現される 2 集合間の関係

2 ABE によるスキーマ設計

この節では，直観的な図的要素を用いて集合間の関係を表現するための方法である ABE(Allowance Band Expression) を提案する．

主体型間の関係を図を用いて表現する場合，主体型を表す図形かまちまちの大きさを持っていたり，互いに重なったりするのは不都合がある．殆んどの場合主体型にはその属性や関連などのシンボルが接続されるため，主体型同士が複雑に重なり合うのはダイアグラムを必要以上に複雑にしてしまう．また，複数の主体型間の関係を，その重なりだけを用いて平面上で全て表現するのは不可能である．これらの理由から,「ベン図」などに代表される単純なダイアグラムでは主体型間の関係を充分に表現することは困難である．

もし二つの主体型 A, B がともにある主体型 AB を包含しているとすると，AB 中に存在する主体は A, B の両方に所属することになり，A と B とが交差し得る，つまり「反排他」の関係を持つことになる．また，上の条件に加えて,「AB が A を包含する」という条件を付け加えると,「$A \subset AB$」及び「$AB \subset B$」から「$A \subset B$」が導かれ，二つの主体型が包含の関係を持つことになる．二つの主体型に包含される主体型を許容線 (Allowance Band) と呼び，許容線がその端にある主体型を包含するかしないかを被覆と呼ぶ (上の例では，許容線 AB は主体型 A を被覆する) と，許容線とその被覆を図 1のように図示することで 2 集合間の 5 通りの関係を表現することが可能である．

2 本以上の許容線が一つの主体型に接続する場合，許容線同士がどのような関係にあるかを表現する必要があるが，全ての関係を表現することは困難であるし，直観的ではない．そのため，一つの主体型に接続する許容線をいくつかの束 (Bundle) に分け，一つの束の中の許容線は次に示す 3 種類の分割のいづれかを持つという制限を加える (図 2).

- 一致 (Equal): 複数の許容線は互いに等しい
- 排他分割 (Exclusive): 複数の許容線は相互に排他である
- 反排他分割 (Co-Exclusive): 複数の許容線は互いに交わり得る

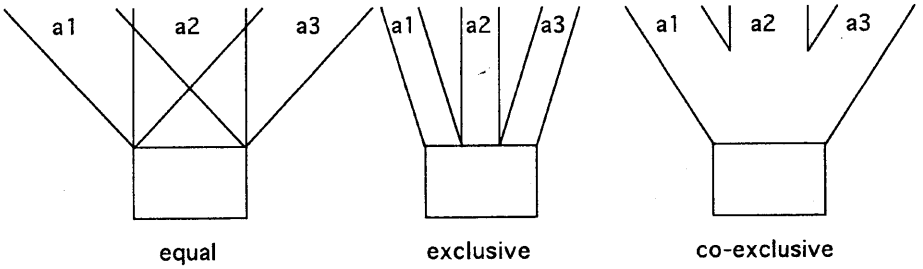

図 2: 束の「分割」

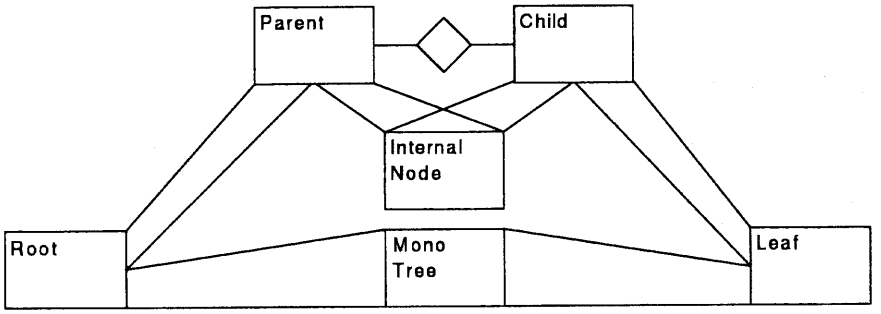

図 3: ABE による木構造のスキーマ

一つの主体型上に複数の束が存在する場合，異なる束の中にある許容線の間の関係は，一意には決定できない．例えば,「学生であり，スタッフであるものは存在し得る」と「学生であり，教師であるものは存在し得る」ということからは,「学生であり，同時にスタッフでも教師でもあるものは存在し得るか?」を論理的に決定することは不可能である．しかし，ABEでは，より直観的に設計を行なうことを可能とするために，一つの主体型の上の複数の束 (及び，束を構成する許容線) 同士は任意に交わり得るという制限を設ける．

主体型と，主体型上に存在し，**分割**と**被覆**のパラメータを持つ束，および二つの束を結ぶ許容線によって，主体型間の関係は表現される．このとき，主体型間の関係，つまりABEのスキーマSは，次のように表現できる．

$S = \langle A, B, E \rangle$

A: 許容線の集合

B: 束の集合

E: 主体型のリテラル集合

図3および以下に，ABEによる木構造 (例 1) のスキーマを示す．

$$
\begin{aligned}
B &= \{B_1, \ldots, B_7\} \\
&\quad b_1 = \langle \text{Root}, whole, exclusive \rangle \\
&\quad b_2 = \langle \text{Parent}, whole, exclusive \rangle \\
&\quad b_3 = \langle \text{Child}, whole, exclusive \rangle \\
&\quad b_4 = \langle \text{InternalNode}, whole, equal \rangle \\
&\quad b_5 = \langle \text{MonoTree}, whole, equal \rangle \\
&\quad b_6 = \langle \text{MonoTree}, whole, equal \rangle \\
&\quad b_7 = \langle \text{Leaf}, whole, exclusive \rangle \\
A &= \{a_1, \ldots, a_6\} \\
&\quad a_1 = \{b_1, b_2\}, a_2 = \{b_2, b_4\}, a_3 = \{b_3, b_4\} \\
&\quad a_4 = \{b_3, b_7\}, a_5 = \{b_1, b_5\}, a_6 = \{b_6, b_7\}
\end{aligned}
$$

3 ABE スキーマの施行

ABE スキーマ Sが存在するとき，次の要件を満たすスキーマ S'を，Sの部分スキーマとよぶ．

$$
\begin{aligned}
&S = \langle A, B, E \rangle, S' = \langle A', B', E' \rangle, \\
&A' \subset A, B' \subset B, E' \subset E
\end{aligned}
$$

Sの部分スキーマ S'が，次の要件を満たす時，「S'は S上で正則」(単に正則) であるとする．

- B'中の束が接続している全ての主体型が E'中に存在する．
- A'中の許容線が属している全ての束が B'中に存在する．
- E'中の主体型を被覆している全ての束が B'中に存在する．
- B'中の全ての束が，その分割に応じて次の条件を満たしている:
  - **排他**: 束に接続する許容線が A'中にただ一本存在する．
  - **反排他**: 束に接続する許容線が A'中に一本以上存在する．
  - **一致**: 束に接続する A 中の許容線が全て A'中に存在する．

□

部分スキーマ $S' = \langle A', B', E' \rangle$ が正則であるということは，S'中の各許容線，束および主体型に要素が存在しても，包含関係や分割には反していないことであるため，S'が S上で正則である時，E'中の各主体型に同時に所属する主体の存在は許される．要素の集合への所属を変更する際に，「要素が E'中の各集合に所属することは許されるか」を知るためには，集合として E'のみを持つ部分スキーマで，正則なものが存在するかを判定できればよい．

以下に示すアルゴリズム **IsProper** は，与えられた集合リテラルのみからなる部分スキーマを，正則条件に従って拡張し，正則な部分スキーマを生成することによって判定を行うものである．

Algorithm IsProper(A, B, E, S)
Input ABE Schema $\langle A, B, E \rangle$, set of the entity type literals E
Output Boolean value which is true if is an entity allowed to belong to entity types in S simultaneously.

```
begin
  A' = {}, B' = {}
  foreach a_i of A(= {a_1, ..., a_n})
    if EtypeOnAB(a_i) ⊂ S
      A' := A' ∪ {a_i}
      B' := B' ∪ BundleOnAB(a_i)
  EndIf
  EndOfForeach
  if S is not CHAIN-CONNECTED with A'
    return FALSE
  EndIf
  foreach b of BundleOnEtype(S)
    if CoverOfBundle(b) ='whole' and ABonBundle(b) ∩ A' = φ
      return FALSE
    EndIf
  EndOfForeach
  foreach b of B'
    if DivisionOfBundle(b) ='equal' and ABonBundle(b) ⊄ A'
      return FALSE
    EndIf
    if DivisionOfBundle(b) ='exclusive' and length(ABonBundle(b))1
      return FALSE
    EndIf
  EndOfForeach
  return FALSE
end                                                          □
```

このアルゴリズムは，許容線の本数 n に対して，オーダー $O(n)$ で AE の成否を判定することができる．

主体型間の排他と包含を記述することができる場合，特定の主体型が常に空であることを強制される場合がある．例えば，「A は Bに包含される」と「A と Bは排他である」とに従うならば，Aにはいかなる主体も所属することはできない．ABE では，排他分割された AB 同士がスキーマの他の部分によって交わる場合に相当する．また，排他分割された AB 同士がスキーマの他の部分によって交わり得る場合も同様である．このような状態を**無為** (Vacuous) と呼ぶが，ABE のスキーマから無為を検出するのは次の手順によって可能である．

1. 排他分割束に接続する共有線それぞれに対し，一本の共有線のみからなる部分スキーマを作り，2. 以降を行なう．

2. 部分スキーマを可能な限り拡張する．

3. 拡張された部分スキーマが正則でなければ，スキーマは無為を含む．

次に示すアルゴリズム **IsVacuous** は，この手順に従ってスキーマの無為を検出するためのアルゴリズムである．

```
Algorithm IsVacuous (S, S_org)
Input ABE Schema S_org(= (A_org, B_org, E_org))
      and ABE sub schema S(= (A, B, E))
Output Boolean value which is false if and only if
      sub schema which include S isn't proper
begin
B_c := {}; B_d := {}
```

```
foreach a in A
  foreach b in BonAB(a)
    next if b ∈ B
    B := B ∪ {b}
    E := E ∪ {ETofB(b)}
    foreach b' in BonET(ETofB(b), S_org)
      if DIVofB(b')='exclusive'
        if ABonB(b', S)=φ
          B_d := B_d ∪ {b}
        elsif ABonB(b', S) ≠ABonB(B', S_org)
          B_c := B_c ∪ {b}
        EndIf
      EndIf
  EndOfForeach
EndOfForeach
if B_c ≠ φ
  foreach b in B_c
    A := A ∪ ABonB(b, S_org)
  EndOfForeach
  return IsVacuous(⟨A, B, E⟩, S_org)
elsif B_d ≠ φ
  result :=true
  foreach T in ×_{b∈B_c}(ABonB(b, S_org))
    result := result ∧ IsVacuous(⟨A ∪ T, C, E⟩, S_org)
  EndOfForeach
  return result
else return IsProper(⟨A, B, E⟩, S_org)
EndIf
end                                                    □
```

無為を検出することは，基本的にグラフの閉路探索問題と同等であり，検出に必要な計算量も閉路探索アルゴリズムと同等であると考えられる．このアルゴリズムの計算量として最も影響の大きいものは，拡張された部分スキーマが正則かを検査する回数 (**IsProper** の実行回数) であると考えられるが，この回数は AB の数など，ABE の記述量に直接は影響されない．これは，一致分割や反排他分割の束中の許容線は全て交わるものとして扱われるためである．逆に，排他分割の束が，別の排他分割の束と交わり得る場合は，探索する必要のある経路が増大する．ただし，交わり得る排他分割束の数は，スキーマの規模に比例して大きくはならないため，このアルゴリズムを実用的な範囲で使用することは可能であると考えられる．

さらに，与えられたスキーマから冗長な部分を検出し，解消するアルゴリズムや，無為を含むスキーマの無為を解消する手法，正則ではない部分スキーマを最小の変更で正則にするための手法を示すことによって，ABE のさらなる妥当性を示すことができると考えられる．

4 結論

データベーススキーマの設計手段として，簡潔で直観性を持つダイアグラム記法である ABE を提案し，ABE 解釈のためのアルゴリズムを示すことで，その妥当性を示した．本論文で示したのは不正を検出するアルゴリズムのみで，実際に使用するためには最小のコストで不正を解消する方法を示すことができる必要がある．

参考文献

[1] Chen, P.P., "The Entity-Relationship Model — Toward a Unified View of Data," ACM Trans. on Database Systems, Vol.4, Issue 4, Mar. 1976, pp. 9–36.

[2] J. M. Smith and D. C. P. Smith, "Database Abstractions–Aggregation," Comm. ACM, Vol.20, No.6, 1977.

[3] Marco A. Casanova and Jose E. Amaral de Sa, "Mapping Uninterpreted Schemes into Entity-Relationship Diagrams: Two Applications to Conceptual Schema Design," IBM J. Res. Develop, Vol.28, No.1, January 1984.

[4] Paris C. Kanellakis, Stavros S. Cosmadakis and Moshe Y. Vardi, "Unary Inclusion Dependencies have Polynomial Time Inference Problems," Proc. of the 15th ACM Symposium on Theory of Computing, 1983.

[5] Richard Hull, Rodger King, "Semantic Database Modeling: Survey, Applications and Research Issues," ACM Computing Surveys, Vol.19, No.3.

[6] Hiroshi ARISAWA, Takashi TOMII, Hitoshi YUI, and Hidehiko ISHIKAWA, "Data Model and Architecture of Multimedia Database for Engineering Applications," IEICE Trans. on Information and Systems, Vol.E78-D, No.11, November 1995.

[7] Hiroshi ARISAWA, Takao MIURA, "ON THE PROPERTIES OF EXTENDED INCLUSION DEPENDENCIES," Proceedings of the Twelfth International Conference on Very Large Data Bases, August, 1986.

[8] Maurizio Lenzerini, "Class Hierarchies and Their Complexity," ACM PRESS, Advances in Database Programming Languages, pp.43–64, 1990.