

データベースにおける型スキーマの発見

三 浦 孝 夫[†] 塩 谷 勇[†]

本稿では、データベースのスキーマと実現値から機械的に型スキーマを発見するための手法を提案する。これまでの手法と異なり、提案する手法はデータベースの実体を解析することから始まる。ここでは実体が複数の型を有するとし、この分類（型スキーマ）が、ISA 階層を用いて、ただだか k 個の近汎クラスに収まるようにする。データベース実現手法の立場からは、この結果は望ましいものである。というのは、どの実体も型情報量が制限されていると考えてよいからである。本稿で提案する手法は、ISA 階層を概念レベルの判定に用い、また同時にそれを意味上の距離計算にも用いている点で特徴的である。さらに、型スキーマを求めるためのアルゴリズムも提案する。

Mining Type Schemes in Databases

TAKAO MIURA[†] and ISAMU SHIOYA[†]

We propose a heuristic method to *mine* type scheme semi-automatically from initial database scheme and the instances. Unlike conventional database design methods, the proposed one starts from examining database entities. We assume one entity may have more than one types and classification (or *type scheme*) might be appropriate when each entity is classified into at most k (least general) classes with respect to ISA hierarchy. Clearly, from the view point of database technique, it is suitable for each entity to keep limited number of type informations. Our method differs from others in evolving ISA hierarchy by introducing semantical *metric*. We propose a sophisticated algorithm to evolve type schemes.

1. 前 書 き

今日の情報システム構築において最も重要な技術の1つはデータベース設計⁵⁾である。実際、利用者が何を意図し何を望んでいるかを明確にし分析して定義するのはこの段階である。一般に、データベース技術ではいくつかの重要な仮定を有し、データベースパラダイムと呼ばれている¹²⁾。最も基本的な仮定は、情報は事前に分類されスキーマを用いて格納されるという点にある。スキーマは処理すべき情報の意図の記述を目的としている。あらかじめ決められた基本機能を組み合わせ、情報の静的・大域的な性質を表現するものであり、一種の知識記述と見なせる。スキーマは情報表現や一貫性制約として記述される。情報の持つ性質をすべてスキーマで記述するため、データベース設計の支援、データベース質問の有効性の検査や最適化、物理特性への効率良いマッピングなどの利点が得られる。概念間の関連もスキーマで示される。

データベースパラダイムの第2の仮定は、データや

スキーマが問題領域と独立に操作できるという点にある。この仮定によって、どの応用分野でもデータベースシステムの提供する基本機能を組み合わせて処理が実現できる。また、大量データを高信頼・高速に処理できる基礎を提供している。

しかし、データベースの成長につれて、変更に従従することが困難になる。変更には設計時の誤りに基づくものもあるが、データベースパラダイムであるが故の問題も生じる。たとえば、社会・組織やその役割の変化によってデータベースの果たすべき処理が変化し、成長過程で得られる情報の質・量が設計時の予測と異なってくる。

この問題はこれまでスキーマ進化として知られる研究テーマで論じられている。しかし、データベース再設計のためにシステムがどのような柔軟な機能を備えるべきかというトップダウン手法が主であり、筆者の理解する限り、現在のデータベース内容（実現値）からみて望ましいスキーマを得るというフィードバックを考慮していない。

データベーススキーマの設計段階は本質的に非決定的であり、意思決定のための対話が多数繰り返される。本稿では、利用者の意思決定を支援するため、評価基

[†] 産能大学経営情報学部

Department of Management and Informatics, SANNO College

準を示し代替案を選択できるための発見的な方法を示す。

はじめに、現在の実体情報の分類（型の設定）やその間の関連（たとえば型階層）が実際に適合しているかどうかに着目する。実体型とは、類似した実体を集めて共通性を与え、他方類似しない実体どうしは互いに異なるように分類する基準をいう。類似性は、一般には特徴値や無例外性（non-exception）などの基準で決められる¹²⁾。データベースの観点からは、この結果を型としてとらえ、型スキーマとして管理する。データベースは対象世界の意味をどの時点でも正しくとらえようとするため、型記述も頻繁に変化する必要がある。各実体がその特徴を保持するならば、その記述や操作は次第に複雑となり、統一的な方式で体系的に管理することが難しくなるであろう。スキーマと実現値の隔離は、データベースがもはや有用でないことの証である。本稿では、スキーマと現在の実体を検査し、実状を反映したものでありながら、同時に簡単な型記述を維持し続けるための方策を示す。

具体的には、データベース実現値のうち、特に実体と実体型に着目してデータベーススキーマの検証と修正を行うための方法を提案する。これは、実体の型付け（型スキーマと呼ぶ）の適合性を調整する方法である。基本的なアイデアは、より簡単な型スキーマを見つけることにある。つまり ISA 階層を用いて型スキーマを減らし、さらに階層内で近い型を合併して小さすぎる型を回復する。前者は型の抽象度を用いた記述の削減であり、後者は型情報の損失をなるべく抑ええた情報損失操作に相当する。損失を少なくするために概念どうしの距離を計算するが、ISA 階層は概念間の近さを表すためにも利用される。

次章では本稿で用いるデータモデル概念を定義する。3章では型スキーマの簡単化手法を示し、4章で実験の評価を述べ、本方式の有効性を明らかにする。

2. 諸 定 義

この章では、AIS (Associative Information Structure) と呼ぶ意味データモデルに基づいて、必要な概念を導入する^{11),12)}。AIS の特徴は、個々の実体値が個別にその内包情報を有する実現値ベースのモデル (instance-based model) にある。

実体 (entity)★とは、対象世界の情報を計算機で処理できるようにコード化したデータである。データ

ベースパラダイムに従いデータは分類されるが、分類基準を実体型 (entity type) または単に型 (type) と呼ぶ。各型 t に対して t に分類された実体の集合を $\Gamma(t)$ とする。 $e \in \Gamma(t)$ ならば、実体 e は型 t を持つという。実体は複数の型を持つことがある。本稿では、どの実体もただか有限個の型を持つとし、さらにデータベースに格納されたすべての実体を表す特別の型 $@E$ も持つとする。 $\Gamma(\{t_1, \dots, t_n\})$ を型 $t_1, \dots, t_n$ を持つ実体集合とする。各型は記号で特定されるとし、異なる記号は異なる型を表現するものとする。

実体は集まって連想 (association) を構成する。ここでもパラダイムに従い、連想を事前に分類し、その基準を述語 (predicate) と呼ぶ。連想 $\langle e_1, \dots, e_n \rangle$ は型付けされている、つまり述語 p の連想によって決定される i 成分 e_i はつねに型 E_i を持つと仮定される。この状況は述語定義 $p(E_1, \dots, E_n)$ で記述され、 p の連想がつねに $\Gamma(E_1) \times \dots \times \Gamma(E_n)$ の要素となる。さらに、連想が複数の述語に分類されてもよい。

共通に表れる述語で特別のものをデータモデルの基本機能とし、その性質を論じることがよくある。本稿では、次章で ISA または汎化 (generalization) と、属性 (attribute) を導入する。後者は関数関係の2項述語 $f(EA)$ であり、型 E の実体 e に対してただか1つの連想 $\langle e, a \rangle$ で $a \in \Gamma(A)$ となるものが存在するときをいう。述語 f は曖昧のない限り $E \rightarrow A$ と表され、型 A を型 E の属性と呼ぶ。

データ実現値の操作のための機能はここでは論じないが、暗黙のうちに論理型言語を仮定している¹¹⁾。これによって、1階述語論理をデータ記述や操作に用いることでの意味の形式化が図れ、表現力の評価基準を設定できる。

例 1 本稿では、新聞記事を用いて様々な概念を示す。以下の例では11の実体型がある：Windows, Unix, PenComputer, Workstation, OS, OS Trends, Small Computer, Laptop Computer, Personal Computer および $@E$ 。意味は語から理解されたい。ここではどの実体（記事）もそのトピックを型と考える。また、2つの述語 develop (OS, Manufacturer) と perform (Workstation, Efficiency) を定義するが、データモデルの説明にのみ用いるとし、以下の例では表れない。図 1 は AIS ダイアグラムであり、スキーマ（の一部）を示す。円記号は型を、菱形は述語を、辺は述語の定義を示す。円記号が重なっていれば、共有する実体があることを強調している。□

★ オブジェクト (object) とも呼ばれるが、この場合振舞いの考慮を意図しており、本稿の範囲でない。

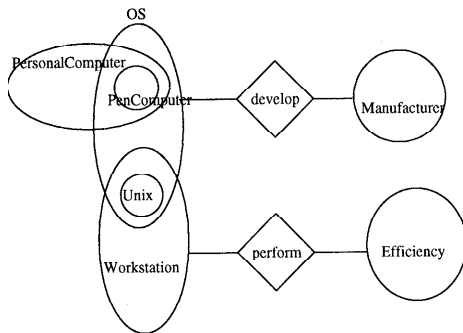


図1 AIS ダイアグラム
Fig. 1 AIS diagram.

3. 型スキーマの発見

3.1 型スキーマ

前述のように、実体 e は複数の型を持つことがあり、 $\tau(e)$ によって e が持つ型の集合を表すとする： $\tau(e) = \{t \mid e \in \Gamma(t)\}$ 。このとき $\tau(e)$ を e の型スキーマ (type scheme) と呼ぶ。本稿ではどの $\tau(e)$ も有限とする。すべての $\tau(e)$ によってデータベースの型付けを表し、データベースの型スキーマと呼ぶ。

一般に $\tau(e)$ は決して小さくない。数多くの実体が存在するため、型スキーマ $\tau(e)$ を個別に管理し、同時に効率良く処理することは容易ではない。型の間に規則や一貫性制約が定義されていればいっそう問題を複雑にするであろう。実体集合の操作と管理を単純にし、効率良く実現するためには、何らかの (型集合に関する) 知識や規則を用いて統一的な機構を導入せざるをえない。本研究では ISA 階層を用いる。

例2 例1において (@E を除いて) 9つの型を用い 11 個の型スキーマを次のように定義する。

実体	型
a, b	Windows, OS
c	Unix, OS, Workstation
d	Windows, PenComputer, SmallComputer, PersonalComputer, OS
e	Unix, PenComputer, SmallComputer, PersonalComputer, Workstation
f	PenComputer, OS Trends, Workstation, OS, SmallComputer, PersonalComputer
g	Workstation
h, i	PersonalComputer
j, k	LaptopComputer, Workstation, PersonalComputer

□

型に関する規則の最も典型的なものは ISA または汎化である^{*}。相異なる2つの型 t_1, t_2 に対して、データベース D において t_2 が t_1 の汎化である、または t_1 ISA t_2 が成り立つとは、 D において $\Gamma(t_1) \subseteq \Gamma(t_2)$ が

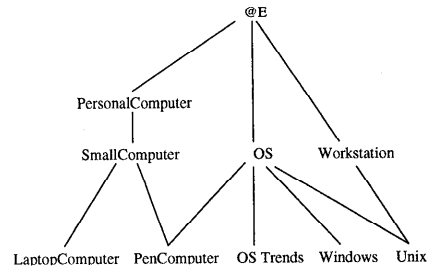


図2 ISA 階層
Fig. 2 ISA hierarchy.

成り立つことをいう。ISA は一貫性制約であり、データベース設計者があらかじめ与えておくとする。汎化の (反射的でない) 推移閉包をとることにより、ISA 階層を得る。この階層によって我々は ISA を正しく推論することができる、すなわち t_1 ISA t_2 と t_2 ISA t_3 から t_1 ISA t_3 を得る。相異なる型 t_1, t_2 に対して、 t_2 が t_1 の直接汎化 (direct generalization) であるとは、 t_1 ISA t_2 であり、しかも t_1 ISA t_3 かつ t_3 ISA t_2 となる t_3 が存在しないときをいう。

ISA は半順序ではない。また ISA 階層は、その名前に反して実際には階層構造とは限らず、複数の直接汎化を持つ型が存在してよい。にもかかわらず我々が注目するのは、現実のモデル化が十分単純で分かりやすく、抽象化のレベルが明確であるため、利用者の評価が得やすいことにある。

型スキーマは ISA 階層に関して整合していなければならない。すなわち、 $t \in \tau(e)$ かつ t ISA t' ならば $t' \in \tau(e)$ が成り立つ必要がある。この条件はデータベースの型整合性と呼ばれ、どの型スキーマも正しく記述されるための条件である。以下では、型整合条件が成立していると仮定する。

例3 例1の ISA 階層では、11 個の親 (直接汎化) があり、どの型もただか 2 つの親を持つ。図2は本例題の階層を表している。□

実体 e の型スキーマ $\tau(e)$ が型 t, t' を要素に持ちしかも t ISA t' であるならば、 $\tau(e)$ から t' を取り除くことで型スキーマの簡単化 (simplification) ができる。型 t が $T = \{t_1, \dots, t_n\}$ において近汎 (least general) とは、 $t \in T$ であり、しかも t_i ISA t となる $t_i \in T$ が存在しないときをいう。実体 e に対して、 $LG(e)$ を $\tau(e)$ 内で近汎なものの集合とする： $\{t \in \tau(e) \mid t \text{ は } \tau(e) \text{ で近汎}\}$ 。これを e の LG 集合

^{*} 型に関する規則は他にもあり、和や積を用いて拡張した拡張包含関連 (extended inclusion) や単項述語として型をとらえる一般型制約が考えられる¹¹⁾。

という。型整合性の定義から、 $t \in LG(e)$ かつ t ISA t' が成り立てば、必ず $t' \in \tau(e)$ となる。逆に、どの $t' \in \tau(e)$ も t ISA t' を満たせば $t \in LG(e)$ となる。

例 4 例 3 で導入した ISA 階層を用いて例 2 の各実体 e に対して $LG(e)$ を生成する。ここで、第 3 項 (実体集合) は当該型を持つ実体の集合を表し、たとえば *Windows* の実体集合は a, b の他に d も含まれることを示している。

型	実体 (実体集合)
<i>Windows</i>	a, b a, b, d
<i>Unix</i>	c c, e
<i>Windows, PenComputer</i>	d d
<i>Unix, PenComputer</i>	e e
<i>PenComputer, OS Trends, Workstation</i>	f f
<i>Workstation</i>	g g, c, e, f, j, k
<i>PersonalComputer</i>	h, i d, e, f, h, i, j, k
<i>LaptopComputer, Workstation</i>	j, k j, k
<i>OS</i>	a, b, c, d, e, f
<i>SmallComputer</i>	d, e, f, j

OS だけを型に持つ実体はないが、OS を型に持つ実体は存在している： a, b, c, d, e, f 。このように、LG 集合として生じてはいないが、その実体集合が空ではないようなものが最後の 2 行で表されている。□

実体 e の型スキーマが位数 (order) k で単純 (simple) であるとは、 $LG(e)$ の要素数が k 以下のときをいう。データベースが位数 k で単純であるとは、どの型スキーマについても位数 k で単純であるときをいう。(位数 k で) 単純であるデータベースではどの実体もたかだか k 個の型を有するため、単純で統一した実現方法を定義できる。また ISA 階層を用いた型推論によって、各型記述 $\tau(e)$ が単純になる。しかも単純化はかなり直観的な近汎概念に基づいてなされるため、利用者は簡単に型スキーマの設計、検査および検証ができる。

例 4 において $k=2$ とすれば、単純でない型スキーマ $\{PenComputer, OS Trends, Workstation\}$ を得る。

3.2 型スキーマにおける連言型

本稿の目的は、型スキーマと実現値から、単純な型スキーマを得るプロセスを論じることにある。ここでは問題の複雑さを示し、位置付けを明確にする。

まず、実体集合を互いに素となるように配置してみる。このような設定は容易である： $\Gamma(t_1), \Gamma(t_2)$ が交われば、これを $\Gamma(t_1) - \Gamma(t_2), \Gamma(t_2) - \Gamma(t_1)$ および $\Gamma(t_1) \cap \Gamma(t_2)$ の 3 つに分割し、型 t_1, t_2 を新しい型 t'_1, t'_2, t'_{12} で置き換えればよい。このとき、($\odot E$ を除いて考えれば) どの実体も 1 つの型を持ち、位数 1 で単

純となる。しかし、この方法は明らかに問題がある。実体が複数の型を有するため、実体集合の指数個の組合せの交差が考えられ、非常に多数の型を生成する可能性がある。

実体 e と ISA 階層に対して、 $LG(e) = \{t_1, \dots, t_n\}$ が単純でない (つまり $n \geq k$) とするとき、新しい型 $\wedge LG(e)$ をその実体集合が $\Gamma(t_1) \cap \dots \cap \Gamma(t_n)$ となるように定義する。このとき $LG(e)$ は ($\odot E$ を除いて考えれば) 位数 1 で単純になる。実際、どの i についても $e \in \Gamma(t_i)$ であるから $\Gamma(\wedge LG(e))$ は空ではない。しかも $\wedge LG(e)$ ISA t_i であり、 e が新しい型 $\wedge LG(e)$ を持つから、 $LG(e)$ は $\{\wedge LG(e)\}$ となる。新しい型 $\wedge LG(e)$ を ($t_1, \dots, t_n$ に関する) 連言型 (conjunctive type) と呼ぶ。

連言型は、互いに素な集合と比べると、実現値に対する型スキーマとして最も細かいものを算出する方法として優れているが、 $\Gamma(\wedge LG(e))$ がかなり小さくなるという欠点を持つ。一般的に言って、細分化された分類はデータベースの観点から見れば好ましいものではない。なぜなら、質問処理において構文解析、有効性検査や最適化等に時間を要し、しかもその割に大量データの一括処理という利点を引き出しにくいからである。また ISA 階層を用いた型の推論にかなりのオーバヘッドを生み、利用者は質問の構成に際してどの型を用いるのか判断に迷う。この意味で、連言型 $\wedge LG(e)$ は単純な型スキーマを生成するが、本研究の目的にはそぐわない。

例 5 例 4 から簡単に分かるように、要素数が 1 の LG 集合は 4 つ (*Windows, Unix, Workstation* および *PersonalComputer*) あり、それぞれに対応する実体集合の要素数は 3, 2, 6, 7 である。同様に、要素数が 2 の LG 集合に対応する実体集合は 3 つあり、要素数はそれぞれ 1, 1, 2 である。要素数 3 の LG 集合に対する実体集合は 1 つだけであり、これは単一要素集合である。明らかに、LG 集合の要素数が増えるほど対応する実体集合の要素数が減少する。本稿では $k=2$ として議論する。□

3.3 型スキーマの統合

本稿で提案する方法は、型スキーマの統合化 (unification) である。統合化とは型スキーマをより単純にする (LG 集合の要素数を減らす) ことを意味し、いくつかの型を同一視する手法である。このとき、対応する実体集合が統合されその要素数が回復する。統合化の過程では、型スキーマが単純でないか、その実体集合の要素数がしきい値 (threshold) v を下回る限り、型スキーマを変更し続ける。

型を統合して実体集合の要素数を回復するとき、もとの型付けは失われるため、情報の損失が生じる。しかし、非常に小さい実体集合となった理由は型付けそのものにあるため、この情報損失は許されると考えてよい。問題は、情報損失と精密な型付けのトレードオフにある。では、どのようにして型スキーマを統合すべきであろうか？ また、どのような視点から決定すべきであろうか？

ここで目標とするスキーマとは実体の型付け基準であり、実体集合の状況から‘型の意味を変更する’という点で発見的である。意味を変更するために距離を用い、型の発見に利用する。一般的なデータベーススキーマとの関連は本稿の枠を越えた議論であることに注意されたい。

まず、非形式的にアイデアを示す。2つの実体 e_1, e_2 に対し、 $LG(e_1) = \{t_1, t_2\}$, $LG(e_2) = \{t_1, t_3\}$ とする。ISA 階層に関して、 $t_2 \vee t_3$ を t_2, t_3 の最小共通祖先 (least common ancestor) とし^{*}, $LG(e_1), LG(e_2)$ を $\{t_1, t_2 \vee t_3\}$ で置き換えれば統合できる。さらに、もし $t_1 \text{ ISA } t_2 \vee t_3$ が成り立てば、 $LG(e_1)$ は $\{t_1\}$ に単純化される。

実体 e_1 に対して、 $LG(e_1)$ は $\{t_1, t_2\}$ から $\{t_1, t_2 \vee t_3\}$ に変更されている。すなわち t_2 は $t_2 \vee t_3$ に弱められており (実際 $t_2 \text{ ISA } t_2 \vee t_3$ である)、さらに一般的な型に変更されることもある。しかし、 $t_2 \vee t_3 \text{ ISA } t_1$ が成立することはない、さもないと $t_2 \text{ ISA } t_1$ が成り立ち、 $LG(e_1)$ は近汎でない要素を含むことになる。この意味で $\{t_1, t_2 \vee t_3\}$ は $\{t_1\}$ よりも強い型記述を表しており、 $\{t_1, t_2\}$ と $\{t_1, t_3\}$ の統合には合理的である。本稿の目的は単純なスキーマを得ることであって、汎化のための計算が簡単に行える必要がある。たとえば、 $t_2 \vee t_3$ を t_2, t_3 の選言型 (disjunctive type) と定義することもできるが、この方法では現状の ISA 階層が修正され、階層上の位置で距離を規定する方法が採用し難くなってしまう。

上述のアイデアを一般化する。型の組 t, t' で $t \text{ ISA } t'$ となるとき、ISA 階層内で $t_0 = t, t_1, \dots, t_N = t'$ となる列で各 t_i が t_{i-1} の直接汎化となるものが存在する ($i = 1, \dots, N, N > 0$)。このような列が複数あれば、最小の N を選ぶ。この N を t と t' の距離 (distance) と呼び、 $dist(t, t')$ で表す。2つの型 t_1, t_2 に対し、ISA 階層で t が t_1, t_2 の近汎型 (least general type) であるとは、(1) t が t_1, t_2 の最小共通

祖先で、(2) $t' \text{ ISA } t$ かつ t' も t_1, t_2 の共通祖先となる型 t' が存在せず、しかも (3) t が (1), (2) を満たす x で、 $dist(t_1, x) + dist(t_2, x)$ の最小値を与えるとときとする。以下では、このような t を $t_1 \vee t_2$ で表す。

例 6 例 3 の階層を用いれば、Windows $\vee$ Unix は OS を表し (Windows と Unix の距離は 2), PenComputer $\vee$ SmallComputer は SmallComputer を表している (PenComputer と SmallComputer の距離は 1)。

$T_1 = \{t_1, \dots, t_n, s_1, \dots, s_m\}$, $T_2 = \{t_1, \dots, t_n, u_1, \dots, u_k\}$ を 2つの型スキーマとする。ここで、 $n, m, k \geq 0$ であり、 $t_1, \dots, s_1, \dots, u_1, \dots$ は相異なるとする。もし $m > 0, k > 0$ ならば、 $T_1 \vee T_2$ を $\{t_1, \dots, t_n, s_i \vee u_j\}$ と定義する。ここで、 s_i, u_j は $s_i \vee u_j$ (ただし $i = 1, \dots, m, j = 1, \dots, k$) の最小距離を与える型であり、 $dist(T_1, T_2)$ を $dist(s_i, u_j) - n$ と定義する。統合化された型集合の要素数は $m > 1, k > 1$ ならば減少する。 $m = 0$ ならば、 $T_1 \vee T_2 = T_1$ および $dist(T_1, T_2) = k - n$ と定義する。同様に、 $k = 0$ のときは、 $T_1 \vee T_2 = T_2$, $dist(T_1, T_2) = m - n$ とする。

$T_1 \vee T_2$ の定義が T_1, T_2 を記述するのに十分であろうか？ たとえば、 $\{t_1, \dots, t_n\} \cup \{s_i \vee u_j, i = 1, \dots, m, j = 1, \dots, k\}$ の方がより正確ではないのだろうか？ 本稿の目的は型スキーマの統合化と単純化の双方にあり、単純化の観点からは逆に後者の定義は厳格すぎるように見える。結局、定義の方法の根拠を得る基準が定まりそうにないため、発見的にとらえざるをえない。また、 $dist(T_1, T_2)$ の定義では、 $-n$ 値は T_1, T_2 の類似性を表している。 n がなければ、違いと類似性を区別することができない。

例 7 例 3 の階層を用いれば、 $\{\text{PenComputer}, \text{Workstation}, \text{OS Trends}\} \vee \{\text{Workstation}\}$ は $\{\text{Workstation}\}$ となり (2つの型集合の距離は 1), $\{\text{PenComputer}, \text{Windows}\} \vee \{\text{PenComputer}, \text{Unix}\}$ は $\{\text{PenComputer}, \text{OS}\}$ (2つの型集合の距離は 1) となる。この結果はさらに単純化され、 $\{\text{PenComputer}\}$ となる。

実体 e の $LG(e)$ が k 以上の要素数を有するか、 $\Gamma(\wedge LG(e))$ の実体数がしきい値 v 以下とする。このとき $LG(e)$ からの距離が最小となる $LG(e')$ を選んで統合する。 $\Gamma(\wedge LG(e))$ と $\Gamma(LG(e'))$ の実体 e'' は、新しい型スキーマを有するように変更されねばならない： $\tau(e'') = LG(e) \vee LG(e')$ 。

この処理を繰り返して、もはや上述の e が見つからなくなり、変化が生じなくなるまで続ける。変化が生

^{*} 候補が複数あれば、辞書式順序などの何かの順序を用いて選ぶとする。

じない状態には、どの型スキーマ $LG(e)$ も単純で対象となる LG 集合がない場合がある。このとき基準を満たす状況になっている。処理対象となる LG 集合があるが、どう統合しても単純にはならない場合とは、統合の定義から、どの型スキーマも k 個以上の共通した型を含む場合である。この場合も変化が生じないため、 $\wedge LG(e)$ を新たに生成することで単純にすることができる。

例 8 例 4 では、要素数の少ない集合の数が示されている。1つの型だけを有する要素数 1 の集合はないが、2つの型を持つものは2つある。したがって、 $k=2, v=2$ と仮定すれば、結果的に、1つの型だけを有する実体集合を2つ、2つの型を有する実体集合を2つ、3つの型を有する（要素数 1 の）実体集合を1つ得る。□

3.4 統合化アルゴリズム

ここでは、UNIFY と呼ばれている型スキーマ統合化アルゴリズムを示す。 $T_1, \dots, T_n$ をデータベース中のすべての（近汎）型スキーマ、 E_i を型スキーマが T_i である実体集合、 F_i を型スキーマが T_i を含む実体集合とする： $E_i = \{e \mid LG(e) = T_i\}$, $F_i = \{e \mid T_i \subseteq \Gamma(e)\}$ 。ここで構造 TYPE を次のように定義する： $\langle T_i, n_i, E_i, m_i, F_i, k_i \rangle$ 。ここで n_i, m_i, k_i はそれぞれ T_i, E_i, F_i の要素数を表す。混乱のない限り、以下では TYPE 構造の型スキーマ集合を TYPE と呼ぶ。定義より E_i は互いに素であるが、 F_i はそうではない。また、ISA 階層はリスト $\langle a, b_1, \dots, b_k \rangle$ の集合として与えられているとする。ここで、 $k > 0$ であり b_i は a の直接汎化を表す。

統合化アルゴリズムは SUP1, SUP, MERGE, UNIFY の4つの部分から構成される：

アルゴリズム SUP1(a, b) は型 a, b の最小共通祖先 $a \vee b$ と距離を得る：

- (1) もし b が a の直接汎化であるとき、結果は b で距離は 1 である。a が b の直接汎化のときも同様。
- (2) さもないければ、a の各直接汎化 a' と b の各直接汎化 b' に対して、SUP1(a' , b) および SUP1(a, b') を実行し、その和が最小となるものを解とする。

SUP(S, T) は型集合 S, T の最小共通祖先 SVT と距離を得る：

- (1) U を $S \cap T$ とする。
- (2) もし $S \subseteq T$ ならば、 U を解とし、距離を $|T| - |S| - |U|$ とする。 $T \subseteq S$ も同様。
- (3) いずれでもないければ、各 $s \in S - U, t \in T - U$ に対して、SUP1(s, t) を実行し解（型と距離）を v, d とする。d が U の要素の汎化かどうかを検査し、その場合 U から削除

する。残りが最小共通祖先であり、距離は d である。

MERGE(T1, T2, LG) は、型スキーマ T1, T2 を LG に併合して TYPE に組み込む：

- (1) T1, T2 が次のような TYPE 項目を持つとする：
 $\langle T1, n_1, E_1, m_1, F_1, k_1 \rangle$ および $\langle T2, n_2, E_2, m_2, F_2, k_2 \rangle$ 。
- (2) もし LG が項目 $\langle LG, n, E, m, F, k \rangle$ としてすでに TYPE に生じているとき E_1, E_2 を E に、 F_1, F_2 を F に加え、次の項目とする： $\langle LG, n, E \cup E_1 \cup E_2, m', F \cup F_1 \cup F_2, k' \rangle$ 。 E, E_1, E_2 に共通要素がありうるので、 $m' \leq m + m_1 + m_2$ 。 F, F_1, F_2 についても同様。
- (3) もし LG が新しい TYPE 項目なら、次を TYPE に加える： $\langle LG, n, E_1 \cup E_2, m', F_1 \cup F_2, k' \rangle$ 。(2) と同様に、 E_1, E_2 には共通要素が存在してもよく、 $m' \leq m_1 + m_2$ であってよい。 F_1, F_2 についても同様。

統合化アルゴリズム UNIFY は、これらを用いて次のように構成する。入力として型スキーマの集合をとり、終了時に新しい型スキーマ TYPE を生成する：

- (1) 型スキーマ集合を $T_1, \dots, T_h$ とする。
- (2) 入力の型スキーマ T に対して以下を繰り返す。
 (2.1) SUP(T, T_i) を検査し ($i = 1, \dots, h$)、戻り値 S で距離が最小となるものを選び、このときの T_i を T' とする。
 (2.2) MERGE(T, T', S) を実行する。
 (2.3) (2) へ戻る。
- (3) 最後に新しい型スキーマ TYPE を出力とする。

新しい型スキーマを得るために、TYPE 型スキーマで位数 k で単純でないか、または要素数がしきい値 v より小さいものにマークを付け、UNIFY への入力とする。上述の手順で、最終的に単純な型スキーマを得るが、細かい（実体数が少ない）型スキーマを生成する可能性があるため、マークを付けている。

上述の手順では1回のステップでは細かい型スキーマは消えない。というのは、型スキーマ間の距離を、ISA 階層上の意味構造を用いて定義しているが、要素数を用いられないからである[☆]。しかし、型スキーマの統合によって必ず要素数が増加するため、この計算ステップが永久に繰り返されることはない。

どの型スキーマ T も k 個以上の共通の型を持つとする。このとき、SUP 手続きを用いたのでは、どのような型スキーマの組であっても共通祖先集合 LG の要素数を減らすことができない。しかし、意味的な立場からみれば、共通の型 T の連言型 $\wedge T$ を扱っていることと同じであるから、 T を $\wedge T$ で置き換えて考え

[☆] 実際、要素の多い型スキーマが意味的に近いとはいえない。

ればよい。

T_1, T_2 に対して SUP を実行しているとき、共通祖先集合として $\mathcal{Q}E$ を得ることがある： $T_1 \vee T_2 = \mathcal{Q}E$ 。直観的にいえば、このことは T_1, T_2 が直交している、つまり統合の結果、型情報を失ってしまうことを意味する。統合が望ましくないとすべきであろう。実際、次章の実験では、このケースを特別に扱い、統合が望ましくないとし、距離 ∞ を与えている。

汎化記述はデータ量に比して十分小さいと考えても、データベースに定義された型の数に対し、型スキーマは最大指数個存在する。すなわち、統合化アルゴリズムの計算時間は、本質的に向上しない。実際、アルゴリズム自体でベキ集合演算を用いていないため、各ステップあたり TYPES のサイズの多項式時間で完了するが、1 ステップ終了時に TYPE 構造が修正され、型スキーマの組合せの数だけの可能性分だけ繰り返され、最悪の場合、指数回分の繰返しが起きる。何らかの発見的方法による最適化が議論できる。

例 9 例 4 において、単純でない型スキーマ 1 つと 2 つの単一要素集合があった。 $\{PenComputer, OS Trends, Workstation\}$ を比較するため、UNIFY アルゴリズムで、型スキーマを調べあげる。この結果、 $\{Windows, PenComputer\}$ および $\{Workstation\}$ が最小距離 1 を与えることが分かる。例 7 のように、前者を用いて統合化する。その結果、最小祖先集合は $\{PenComputer\}$ となり、2 つの型スキーマは $\{PenComputer\}$ で置き換えられ、その実体集合 ($\Gamma(PenComputer)$) は $\{d, e, f\}$ となる。

この時点で、1 つだけ細かい型スキーマ $\{Unix, PenComputer\}$ が残っている。選択枝は $\{Unix\}$ (距離 0) と $\{PenComputer\}$ (距離 0) であるが、前者とすれば、最終的に次のような単純な型スキーマ集合を得る。この結果、要素数が 1 および 2 である LG 集合に対して、各々対応する実体集合は 5 つ (それぞれの要素数は 3, 2, 3, 6, 7) および 1 つ (要素数は 2) となる。

型	実体 (実体集合)
<i>Windows</i>	a, b a, b, d
<i>Unix</i>	c, e c, e
<i>PenComputer</i>	d, f d, e, f
<i>Workstation</i>	g g, c, e, f, j, k
<i>PersonalComputer</i>	h, i d, e, f, h, i, j, k
<i>LaptopComputer, Workstation</i>	j, k j, k
<i>OS</i>	a, b, c, d, e, f
<i>SmallComputer</i>	d, e, f, j

□

★ ただ、一般には最初のステップで単純化され、問題候補が大幅に絞られる。このため実際の問題にはなりにくい。

4. 実験—キーワード索引

本章では、UNIFY アルゴリズムを用いた実験結果を示す。実験に用いたデータは、日経バイト 1994 年 2 月号で示された 1985 年から 10 年間の同誌記事の索引情報 *Nikkei Byte Article Keyword Index* を基本に、拡張したものを用いている。この索引には 2315 件の記事に関する情報 (そのうち 2 件は誤りであり、実際には 2313 件) が含まれ、各情報は表題、掲載刊号、ページ、主題、およびいくつかのキーワードを含む。たとえば 666 番目の記事は次のものである：

Page Printer on Windows - PART III Basic

Benchmark - Graphics: 1991.09: no.90: p.256:

Topics: Windows: Printer: Benchmark Test

表題が Page Printer ... で、掲載刊号 (no.90)、ページ (p.256)、主題 (Topics)、キーワード (Windows, Printers, Benchmark Test) であることを表す。記事データベースでは「記事 (表題、掲載号、ページ、主題、{キーワード})」をスキーマと考えることができるが、本実験はそれ自体の改善を目的とするものではない。記事を実体に、キーワードを型に対応させて実験を行い、本研究で提案する方式の有効性を検証する。したがって各記事は $\mathcal{Q}E$ 型の実体 (この場合には記事と対応する) と見なし、キーワードを型と考える。キーワードのクラスタリングと似ているが、基準や方法が概念クラスタリング手法などが異なる。

索引には 198 個のキーワードが定義されており、実際には 146 個のキーワードが索引全体で、延べ 2792 回生じる。

索引のキーワード上の ISA 階層は、キーワードの組情報 (キーワード、親キーワード) の集合として記述する。階層には 288 個の直接汎化関係があり、各キーワードは最大でも 4 つの親を持つにすぎない。次は、階層記述の一部を示す例である：

```

APL:language
AX:IBM compatible:Personal Computer
Ada:language
Apple II:Apple product:Personal Computer
Apple product:@E

```

型スキーマを処理するため、ISA 階層を用いて各実体 e ごとに近汎型集合 $LG(e)$ を計算する。この結果元の型スキーマは保持する必要がない。次の表 1 は、 $\tau(e)$ と $LG(e)$ のときのそれぞれのキーワードの数ごとの記事数を示す。近汎型集合で 4 および 5 個の要素

★★ 本来、索引は日本語で作成されているが、処理を容易にするために英語に変換してある。

表1 日経索引要約

Table 1 NIKKEI index summary.

キーワード数	型スキーマの記事数	近汎型での記事数
0	304	304
1	1371	1398
2	513	496
3	106	98
4	18	16
5	1	1
(合計)	2313	2313

表2 平均要素数

Table 2 Average cardinality.

キーワード数	実体集合数	平均要素数
1	138 (1398)	10.13
2	281 (496)	1.77
3	80 (98)	1.23
4	15 (16)	1.07
5	1 (1)	1.00
(合計)	515 組合せ	

キーワード数	1 要素集合数	2 要素集合数
1	11	191
2	181	54
3	68	7
(合計)	260	252

を有する実体は17個存在している。

次に索引(実体集合)の要素数について考察する。次の表2は要約である。同一のキーワード集合を持つ実際集合の数と、それらの平均要素数を示す。

注目すべきは、キーワード数が3であっても平均要素数が1.23であり、またキーワード数が1, 2, 3の場合でさえ、単一要素からなる集合が260も存在している。そこで、この実験では、単純性を決定するための位数 k と、集合の細かさを規定するしきい値 v をそれぞれ3, 2とする。実験では、Sun4/2 SunOS4.1.3上のGNU awkでアルゴリズムが試作され、合計で250行程度である。

そこで、問題となる16種の単純ではない型スキーマに対し、UNIFYアルゴリズムを適用する。たとえば、記事(314)のx86 MicroProcessor, RISC, UNIX, UNIX workstation キーワード集合(その要素数は1)は、Window system, UNIX, UNIX workstationのキーワード集合(要素数は1)と統合され、結果的にUNIX, UNIX workstationの近汎型集合に吸収され要素数3となった。このときの距離は0である。同様にキーワード集合BASIC, C/C++, FORTRAN, Pascal(実体集合の要素数は1)はBenchmark Test, C/C++, FORTRAN, Pascal(実体集合の要素数は1)と統合され、C/C++, FORTRAN, Pascal(要素数3, 距離-1)になった。このように、単純でない近汎型スキーマを統合することで、表3に示すようにすべてを単純に

表3 要素数を考慮しない統合

Table 3 Result of non-simple unification.

キーワード数	実体集合数	平均要素数
1	138 (1401)	10.15
2	282 (514)	1.82
3	79 (94)	1.19
4	0 (0)	0.00
5	0 (0)	0.00

キーワード数	1 要素集合数	2 要素集合数
1	11	19
2	181	53
3	53	2
4	0	0
5	0	0
(合計)	256	74

表4 統合処理結果

Table 4 Result of full unification.

キーワード数	実体集合数	平均要素数
0	0 (304)	0.00
1	127 (1462)	11.52
2	101 (506)	5.01
3	27 (41)	1.52
4	0 (0)	0.00
5	0 (0)	0.00
(合計)	255 組合せ (2313)	

キーワード数	1 要素集合数	2 要素集合数
1	0	15
2	0	51
3	0	2
4	0	0
5	0	0

できる。

しかし、依然として250個を超える単一要素集合が残っており、このままでは、型付けが細かいといわざるをえない。UNIFYアルゴリズムで、しきい値を考察に加えて、これらの細かな型を処理する。この結果を表4に示す。最終的には、実際に生じる型集合の種類が515通りから256通りに半減した。また、この実験では、1ステップで、単純でかつ細かくない型スキーマを発見できた。

5. 関連研究

多くの場合、実体型を用いて対象世界を記述するとき、親型・部分型や型の一貫性制約や述語との関連を考慮する必要がある。その記述を解析するときは、新たな型を生成することになる。このように、データベース(またはそのスキーマ)を意味の記述とする知識発見のプロセスは、データベースにおける知識発見(knowledge discovery in databases, KDD)と呼ばれ、現在多くの研究者の興味を引くテーマとなっている^{1)~4),7),8),15),19)}。

機械学習(machine learning)とりわけ帰納的推論

(inductive inference) は KDD と強い関連を有することが指摘されている⁶⁾。しかし、ここでは主としてスキーマ情報が利用され、実現値や問題領域固有の知識は利用されない。

一般的に、相異なる実体が異なるグループに分類されるための基準が仮定されることが多い。たとえば、これらの実体が同じ特性 (property) を持つときと定義されるもの等がある。このような分類条件は、類似性が特性値で表せるときは、型生成条件の 1 つと見なすことができる。最も簡単な知識発見の例はデータベース質問である。これは、与えられた条件を満たす実体集合を生成するが、ここで質問を型 t の定義 (意味) と見なし、対応する実体集合 $\Gamma(t)$ を質問の答と考えればよい。

特性値を与えるための条件を得ることは簡単ではないが、いくつかの発見的で定量的あるいは定性的な方法が知られている⁶⁾。現実的な視点からの統計的適用として、データベース格納値の統計情報を解析し、たとえば「質問の条件や解に多く生じる」ことからスキーマを発見するなどの方法も考えられる。統計学的手法、たとえばクラスタ分析や主成分分析などは、実現値を定量的な視点から解析し一般的な傾向を導くのに有効である。たとえば、データベース上で効率良く相関関係を求める研究などはこの分野のものである^{1),2)}。しかし、ここでは数値分析が主たる対象であり、データモデル的な意味解析やデータベース固有の分析を行うことを意図していない。また、得られた結果を解析してスキーマ発見に至ることは容易ではない。

他方、定性的な方法を用いれば、スキーマ発見のための新たな手法を得る可能性がある。代表的には「グループ化」手法がある。型 t と属性 a に対して、 $\Gamma(a)$ が $\{a_1, \dots, a_n\}$ のとき、 S_i を $\Gamma(t)$ の要素で、 a -値が a_i となるものの集合と定義する。属性の定義から、 $S_1, \dots, S_n$ は互いに素となる。型 t_i をその実体集合 S_i と対応させれば、グループ化によって部分型 $t_1, \dots, t_n$ を得ることができる。たとえば、駅 **station** が路線に属することを表すスキーマ **belong(station, line)** に対して、駅を路線でグループ化することで、「東海道線の駅」といった部分型を生成できる。

複数の属性を組み合わせた分類の生成も考えることができる。たとえば、路線と都道府県を組み合わせると、「静岡東海道駅」などを生成できる。この場合、グループ化の方法によっては非常に多数の細かな (すなわち要素数の少ない) 集合を生む可能性がある。決定木 (Decision tree) は、この問題を回避するために提案された型発見手法の 1 つである^{17),18)}。ここでは、

グループ化は新たに生成される集合が、一定の要素数を保持するときにだけ実行される。どの属性をどの順でグループ化するかという問題は、型発見においてどのような意味を定義するかということに対応しており、完全に機械化することが難しい。なお、決定木はグループ化に限っていることに注意したい。

領域値上の ISA 階層を用いて、関係モデル上の実現値 (タプル) から述語 (関係スキーマ) を生成するための、特筆すべきアプローチが提案されている^{3),4),7),15)}。領域値を ISA 階層を用いて上位レベルに抽象化される過程で、複数のタプルが集約され、最終的に関係がいくつかのタプルに細分化される。非常に単純な手法でありながら、関係からより精密な意味を表す部分関係を得る方法として有効である。この手法の問題は、タプルにだけ着目している点にある。階層が領域値に関して記述されているのに、関係発見の過程は新たな関係を求めるだけで、新たな領域を得ようとはしていない。しかも得られた結果は (スキーマの一部である) ISA 階層にフィードバックされない点も重要である。

6. 結 び

本稿では、データベース (スキーマと実現値) を構築したあとで、実状に合わせたスキーマを得るための手順を提案した。この手法は、モデル構築アプローチにとって本質的な役割を有する。

統合化アルゴリズム UNIFY は、データベースに対する知識発見の応用例と見なすことができる。データベースパラダイムにおいてスキーマは実現値の意味を記述する方法であるため、機械学習の考え方と似ているが、スキーマと実現値の双方を用いてデータベース自体を進化させる点で異なる手法を用いている。本稿の手法は発見的ではあるが、実験からも分かるように有効である。

本稿では、型スキーマの統合を論じたが、述語についても同様の考察ができる¹⁴⁾。もっと一般に、スキーマ発見の一般的な枠組みを論じることが可能である。

本手法と併合することによって、意図を類推できる超高水準な質問処理を実現することができる¹³⁾。実際、このような処理ではスキーマの変更が生じてもしっかり動作するため、質問処理の独立性も達成できる。

謝辞 本稿ドラフトに対して貴重なコメントをいただいた B. Thalheim 先生 (Tech. Univ. of Cottbus, ドイツ) および T. Bench-Capon 先生 (Univ. of Liverpool, 英国) に感謝します。なお本研究の一部は文部省科学研究費重点領域研究「高度データベース」(課題番号 09230219) の助成による。

参考文献

- 1) Agrawal, R., Ghosh, S., et al.: An Interval Classifier for Database Mining Applications, *Very Large DataBase (VLDB)*, pp.560-573 (1992).
- 2) Agrawal, R. and Srikant, R.: Fast Algorithms for Mining Association Rules, *VLDB*, pp.487-499 (1994).
- 3) Cai, Y., Cercone, N. and Han, J.: An Attribute-oriented Approach for Learning Classification Rules from Relational Databases, *Intn'l Conf. on Data Eng. (ICDE)*, pp.281-288 (1990).
- 4) Cai, Y., Cercone, N. and Han, J.: An Attribute-oriented Induction in Relational Databases, *Knowledge Discovery in Databases*, Piatetsky-Shapiro, G and Frawley, W.J. (Eds.), pp.213-228, MIT Press (1991).
- 5) Elmasri, R. and Navathe, S.: *Fundamentals of Database Systems*, Benjamin (1989).
- 6) Frawley, W.J., Piatetsky-Shapiro, G. and Matheus, C.J.: Knowledge Discovery in Databases - An Overview, *Knowledge Discovery in Databases*, Piatetsky-Shapiro, G and Frawley, W.J. (Eds.), pp.1-30, MIT Press (1991).
- 7) Han, J., Cai, Y. and Cercone, N.: Knowledge Discovery in Databases: An Attribute-oriented Approach, *VLDB*, pp.547-559 (1992).
- 8) Ioannidis, Y. and Lashkari, Y.: Incomplete Path Expressions and their Disambiguation, *Special Interest Group on Management of Data (SIGMOD)*, ACM, pp.138-149 (1994).
- 9) Michalski, R., et al.: *Machine Learning*, Vol.1-4, Morgan Kauffman (1983, 1986, 1990, 1994).
- 10) Mineau, G., Gecsei, J. and Godin, R.: Structuring Knowledge Bases Using Automatic Learning, *ICDE*, pp.274-280 (1990).
- 11) Miura, T. and Ariswa, H.: Logic Approach of Data Models, *Future Databases*, pp.145-159 (1990).
- 12) Miura, T.: Database Paradigms towards Model Buildings, *Object Role Modelling*, pp.228-258 (1994).
- 13) Miura, T. and Shioya, I.: Strategy Control for Database Queries, *Conf. and Workshop of Database and Experts Systems Applications (DEXA)*, pp.231-240 (1995).
- 14) Miura, T. and Shioya, I.: Paradigm for Scheme Discovery, *Intn'l Symposium on Cooperative Database Systems for Advanced Applications (CODAS)*, pp.101-108 (1996).
- 15) Ng, R. and Han, J.: Efficient and Effective Clustering Methods for Spacial Data Mining, *VLDB*, pp.144-155 (1994).
- 16) Ohsuga, S: How Can Knowledge-based Systems Solve Large-scale Problems?: Model-based Decomposition and Problem Solving, *Knowledge-Based Systems*, Vol.6, No.1, pp.38-62 (1993).
- 17) Quinlan, R.: Induction of Decision Trees, *Machine Learning*, Vol.1, No.1, pp.81-106 (1986).
- 18) Quinlan, R.: Learning Logical Definition from Rules, *Machine Learning*, Vol.5, No.3, pp.239-266 (1990).
- 19) Piatetsky-Shapiro, G. and Frawley, W.J. (Eds.): *Knowledge Discovery in Databases*, MIT Press (1991).

(平成 8 年 11 月 22 日受付)

(平成 9 年 4 月 3 日採録)

三浦 孝夫 (正会員)



京都大学理学部卒業, 工学博士 (東京大学)。現在, 産能大学経営情報学部情報学科助教授。データモデル, 知識表現, 演繹データベース, 複合オブジェクトなどの分野の研究に従事。電子情報通信学会, ACM 各会員。著書に「データモデルとデータベース」(全2巻, サイエンス社)。

塩谷 勇 (正会員)



東京電機大学大学院修士課程修了。現在, 産能大学経営情報学部情報学科助教授。時系列モデルの同定, 論理プログラミング, グラフ文法, 論理データベースの研究に従事。電子情報通信学会, ACM 各会員。