TR-535

演繹・オブジェクト指向データベース

横田一正，西尾章治郎(大阪大学)

February, 1990

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

演繹・オブジェクト指向データベース*

横田 一正† 西尾 章治郎‡

1. はじめに

関係モデルの拡張あるいは新しいデータベース(データモデル)として、本特集の演繹データベース(Deductive DataBase; DDB)以外にも、さまざまな提案がなされている。たとえば、非正規関係、複合オブジェクト、意味データモデル、あるいはオブジェクト指向データベース(Object-Oriented DataBase; OODB)などである。この背景としては、応用分野の広がり、ハードウェアの進歩、プログラミング・パラダイムの影響、知識処理技術の進展、データベース言語とプログラミング言語の間のインピーダンス・ミスマッチの問題、などのさまざまな要因が考えられる。これらが互いに関連しあって、新しいデータベースへの要求の大きなうねりとなっている。

これらの中で大きな流れとなっているのが、DDBとOODBの二つである。DDBの長所としては高い推論能力、(一階述語論理を基にした)明確な形式的基礎、短所としてはモデル化能力の貧困さ、

などが指摘されている。一方、OODBの長所としては高いモデル化能力、豊富な拡張性、応用への適応性、短所としてはデータモデルなどの基本概念に対するコンセンサスのなさ、低い推論能力、形式的基礎の貧困さ、などが指摘されている。したがって、これら二

---

*Deductive and Object-Oriented Databases by Kazumasa YOKOTA (Fourth Research Laboratory, Institute for New Generation Computer Technology) and Shojiro NISHIO (Education Center for Information Processing, Osaka University)

†(財) 新世代コンピュータ技術開発機構 第4研究室

‡大阪大学 情報処理教育センター

つの長所を統合したデータベースが，次世代知的データベースの一つとして強く求められている．これが本稿の主題である**演繹・オブジェクト指向データベース** (Deductive and Object-Oriented Database; DOOD) である．

2. ではDDBの拡張としてのDOODの研究の枠組について述べる．**3.** ではOODBの一つの側面である受動的オブジェクトの諸性質のDDBへの組込みについて，**4.** ではOODBのもう一つの側面である能動的オブジェクトとDDBの関係について，研究の現状を説明する．さらに，**5.** では，昨年の12月の「第一回演繹・オブジェクト指向データベース国際会議」(DOOD89) を踏まえ，今後の研究動向について簡単に概観したい．

2. 演繹・オブジェクト指向データベースの研究の枠組

これまで活発に研究開発が行われてきた演繹データベース (DDB) に対して，前章で述べたような欠点を改善するために，さまざまな拡張が提案されている．それらは以下のように分類できる．

(1) 論理的拡張

否定，選言，存在限量子，空値，確信度などの論理的要素のDDBへの導入．

(2) カプセル的[1]拡張

a. 構造的拡張

非正規関係，複合オブジェクトなどの構造データの導入

b. 手続き的拡張

データと手続きのカプセル化

c. オブジェクト・アイデンティティの導入

識別子によるオブジェクトの直接的表現

(3) (計算) パラダイム的拡張

a. 制約パラダイム

オブジェクトを項にした制約論理型言語における導出および制約処理系による質問処理

b. オブジェクト指向パラダイム

[1] この言葉は通常の「カプセル化」より広い意味で用いている．つまりデータと手続きの一体化だけでなく，複数の実体を一体化することもここでは意味している．

オブジェクト間のメッセージ・パッシングとオブジェクトの内部処理による質問処理

この (1), (2), (3) は互いに直交した拡張で、互いに組合せが可能である. 本稿では、演繹・オブジェクト指向データベース (DOOD) を、このように DDB の拡張の形式的枠組の中で考えたい. このような意味での DOOD が研究対象としているのは、その中の「オブジェクト指向 (データベース) 的」拡張、つまり、主として (2) と (3)b が対象と考えられる. 言うまでもなく (1) を除外しているのではなく、(1), (2), (3) の拡張のさまざまなレベルの組合せを考えることが必要である. この意味で、DOOD には、さまざまなデータベースが上記の枠組の中に存在しうる.

まず、DOOD の構成要素の一つであるオブジェクト指向データベース (OODB) を考えてみたい. OODB は前章で述べた背景の中で、Smalltalk-80 などのオブジェクト指向言語 (OOPL) の成功の影響を強く受けてきた. その内容について統一化の提案[4] もあるが、合意が得られているとは必ずしも言いがたい. そこで本稿では、OODB の '必要条件' とされているものをいかに数多く満たしているかより、むしろ、そのもとになるオブジェクト指向パラダイムをいかに反映させるかを考えたい. OODB におけるオブジェクトの概念は、メソッドの有無により構造的オブジェクトと行動的オブジェクト、あるいは静的オブジェクトと動的オブジェクト、オブジェクトの自律性の有無により受動的オブジェクトと能動的オブジェクト、などに分類される. ここでは、実体の自然なモデリングを目的とした受動的オブジェクトと、計算過程の自然なモデリングを目指した能動的オブジェクトとの、二つの側面に分けて考えることにする.

OODB の研究はこれら二つに対応して、実体をいかに自然にモデル化するかの流れと、OOPL にデータベース機能を取り込む流れの二つがある. データベースは実世界の実体をいかに計算機に取り込むかを研究対象にしてきたので、最近の OODB の研究では受動的オブジェクトを中心に検討することが多いように思える. (いくつかは能動的オブジェクトの特徴とも重なるが) その特徴としては、オブジェクト・アイデンティティ、複

合オブジェクト，カプセル化，メソッドと情報隠蔽，型とクラス，階層構造，などが考えられている．しかし，個々の内容については，漠然とした合意はあるものの，応用に依存することも多く，必ずしもその内容にコンセンサスは得られていない．もう一方の能動的オブジェクトの側面については，質問処理やデータベース管理を各オブジェクトが自律的に行うことであり，メッセージ・パッシング，協調問題解決，オブジェクトの固有性，各オブジェクトによる同時実行制御や障害回復，さらに永続性管理，などが議論されている．これらは，分散（あるいは並列）データベース管理システムで考えられてきた自律性にも対応しているが，DDB の形式的枠組といかに関連させるかが大きな問題である．

いくつかの OODB がそうであるように，オブジェクトの受動的側面にだけ着目し，それを管理するデータベース・システムを考えることもできる．しかし，“オブジェクト指向”のもともとの考えでは，オブジェクト指向の二つの側面をともに考える必要がある．これが，OODB を‘オブジェクト・ベース’や‘抽象データ型ベース’と区別する重要な点でもある．この二つの側面は独立したものではなく，型階層，手続きの解釈，管理機能など多くの面で関連している．この二つは，上述した DDB のオブジェクト指向的拡張の，カプセル的拡張とパラダイム的拡張（の (3)b）にそれぞれ対応しており，さまざまなレベルで組合せが可能である．これまで個々の点について多くの拡張が研究されてきているが，DOOD の研究としては，それらを上記のような枠組で統一的に考えることが必要とされている．

3. 演繹データベースへのオブジェクト指向概念の導入

演繹データベース（DDB）の基本的な表現の単位は一階項（述語）であった．これに，先に述べた受動的オブジェクトの諸性質を導入するためには，項表現をいかに拡張するかがキーとなる．本章では，項表現の拡張の主要な課題を述べ，諸性質をいかに導入するかを，いくつかの研究成果を基に解説する．

3.1 拡張の課題

オブジェクト指向概念の導入による項表現の拡張の

目的には、表現能力の向上、表現の効率化、推論能力の強化などがある。表現能力に関しては複合オブジェクトのような構造データの組込みがあり、表現の効率化としては一階項の非効率性(引数の固定した数や位置など)の改善があり、推論能力の強化としては継承関係の推論のような発想推論の組込みがある。以下それらの主要な課題を考え、**3.2** 以降で具体的な研究例を紹介しよう。

- 組表現

引数の数や位置を(属性-値対による)組表現によって自由にする。同時に(無限)構造の表現も行う。この組表現によって、部分情報の扱いも問題とされている。

- 型階層

表現間の冗長性を減少させるために、型とその階層、さらに継承関係を導入する。ただし、型の位置づけ、階層の種類については議論がある。

- 集合の扱い

組表現だけでなく、集合構成子を追加することによって、複合オブジェクトを表現可能にする。集合は元来高階の概念であるので、その意味論やそれをオブジェクトにするかどうかについては議論がある。

- オブジェクト・アイデンティティ

オブジェクト識別子によってオブジェクトを特定し、そしてその共有を可能にする。これは組表現での識別子の拡張にもなっている。識別子の扱いも多様である。

- メソッド

各オブジェクトに対するメソッド(とその手続き)の定義を可能にする。また、メソッドに対応する手続きの表現とその言語としての能力にも多くの議論がある。

- その他

実体と型の区別、個体値と集合値の区別、継承関係の扱い、構成子の多種化など多くの議論がある。

3.2 ψ-項 — 型階層をもった組表現[2)]

組表現に基づくデータ表現は、データベースの分野だけでなく、知識表現言語の中や自然言語での素性構造などで多くの研究がなされてきた。その中で、非常に美しく形式化されているのが、Aït-Kaci の ψ-項である[2)]。それは以下のように要約できる。

(1) 型(構造)を ψ-項という組として表現する。これは

先に指摘した一階項の欠点を克服するためである．たとえば，人の情報を表現する ψ- 項を考えてみよう．

$$
\begin{aligned}
X: 人\,[&識別 \rightarrow 名前\,[姓 \rightarrow Y: 文字列,\\
&\qquad 名 \rightarrow 文字列],\\
&住所 \rightarrow \top,\\
&父親 \rightarrow 人\,[識別 \rightarrow 名前\,[姓 \rightarrow Y]],\\
&恋人 \rightarrow 人\,[恋人 \rightarrow X]]
\end{aligned}
$$

'[', ']' が組構成子で，その構成要素の中で '$\rightarrow$'[2] の左が属性名で右が型である．たとえば '人'，'名前'，'文字列'，'$\top$' は型で，'識別'，'姓'，'名'，'住所'，'父親'，'恋人' は属性名である．さらに ':' の左の記号 'X'，'Y' はタグと呼ばれ，それぞれが同じ型をもつこと，つまり等値制約を表現している．たとえば上記の例では，父親の姓と同じであること，そして恋人は相思相愛であることを意味している．

形式的には，空属性名を含む属性名の集合から構成される木領域 Δ から，型集合への関数を ψ，タグ集合への関数を τ とすると，ψ- 項は (Δ, ψ, τ) の三つ組で定義される．上記の例は図 -1 のように図示できる (ただし，空属性名 e とタグ Z_i は上記の表現では省略している)．

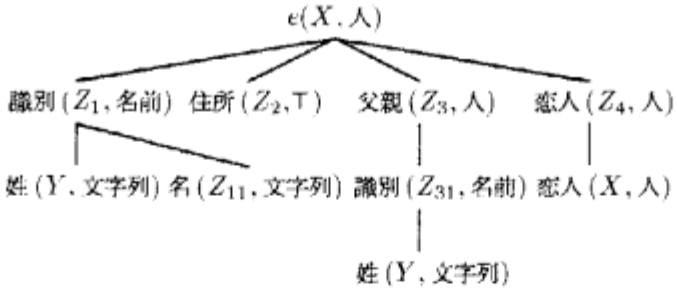

図 -1　ψ- 項の定義

(2) ψ- 項のもともとのねらいは継承の形式化であった．そのために，型間の包摂関係 (半順序) を ψ- 項間の包摂関係に拡張し，それから束を構成する．たとえば，学生 $\preceq$ 人，労働者 $\preceq$ 人，大学院生 $\preceq$ 学生，といった型間の半順序関係 $\preceq$ を，ψ- 項間に拡張し，これによって属性の継承を行えるようにしている．(1) の例の $\top$ はこの束でのトップを表している．ψ- 項は一階項の拡張になっており，属性の継承をともなうこの束演算の交わ

[2] 元の記法は $\Rightarrow$ であるが，ここでは他と統一するために $\rightarrow$ を用いている．

り操作は一階項の単一化の拡張となっているが，この拡張部分は一階論理の発想推論となっている[9]．

さらに参考文献[2]は，このψ-項を基にして，以下のように展開している．

(3) ψ-項間の交わり操作(単一化)のアルゴリズムを与える．

(4) ψ-項に集合表現を導入し，ε-項に拡張する．ε-項で集合は，ψ-項の集合の略記記法としての地位しか与えられていない．

(5) (2),(3),(4)と項書換えに基づいたプログラミング言語KBLを定義する．

ψ-項をオブジェクト指向データベース(OODB)の観点からみると，**3.1**の課題の一部(組表現，型階層，継承)しか解決しておらず，データベースの視点も少ないが，確定節への組込み[3],[6]や複合オブジェクトへの応用[5]のほか，次節以降の拡張項の研究，つまり，DDBのオブジェクト指向的拡張の研究に大きな影響を与えている．

3.3 O-論理[14],[17]

Maierは，ψ-項ではそれほど意識して論じられなかったオブジェクト・アイデンティティの概念をより明確にし，さらにルールの記述も可能なO-項(ψ-項と記法上はよく似ている)を提案し，その項表現のもとでO-論理を構築した[17]．彼はこの論文の冒頭で，データベースへの応用を目指した論理とオブジェクト指向プログラミングの融合のための形式的な基盤を与えることが，O-論理を提案する目的であることを明言している．ただし，この論文自体は，O-論理の提案と，O-論理が内包する意味論，限量子，集合，矛盾の扱いなどに関するいくつかの問題点を提起した段階で留まり，KiferとLuによるO-論理の見直しによって一応完全な理論が構築された[14]．O-論理の提案は，DOOD構築のための以後の理論的研究に多大の影響を及ぼしており，アイデアとしては画期的なものであった．以下のO-論理の話題は，KiferとLuの改訂[14]を踏まえたものであるが，彼らの改訂点の概略を列挙すると次のようになる．

(1) 属性値として，オブジェクトの集合を扱えるようにした．ただし，意味論的には一階になる工夫がなさ

れている[8].

(2) オブジェクトの集合上に束構造を導入し，対象とする型のある属性値が与えられていないとき，その上位の型の対応する属性値を継承することにより，値の継承を可能にした．

(3) ある属性値に論理的な矛盾が生じた場合には，その値として束構造の“矛盾値”を代入して局所的な矛盾にとどめ，他の正常な値をもつ属性値に関する推論には影響を与えないようにした．

(4) 導出に基づく健全で完全な証明手続きを与えた．

O-項は，それ自身内部構造を有しないオブジェクトそのもの(基本オブジェクトとよばれる)の集合から再帰的に定義される．項は，必ずしも基本オブジェクトと関連していなくてもよく，オブジェクト変数をもとに記述されていてもよい．この変数により，ψ-項におけるタグのように等値制約を表現できる．O-項では型名のほかに，オブジェクトから属性値への関数あるいは属性値集合への写影と解釈できるラベルの集合が導入される．このようにして構成される単項あるいは複数のO-項に対して，O-論理では，さらに論理的な結合子 $\vee$, $\wedge$, $\Leftarrow$(含意), $\neg$, および，限量子 $\forall$, $\exists$ を導入した表現が含まれる．

まず，オブジェクト変数を用いないO-項の例として，‘職員’という型に属するオブジェクト‘花子’を表現するO-項は，次のように記述される．

職員: 花子 [名前→文字列:“花子”,
年齢→整数:30,
趣味→{ ゲーム: チェス, ゲーム: テニス },
職場→学科:CS_2[学科名→
文字列:“情報工学科”]]

ここで，‘:’の左側は型，右側はオブジェクトを表す．また，‘$\rightarrow$’の左側は属性を表すラベルであり，右側は属性値である．‘趣味’に対する属性値を見れば明らかなように，集合も扱える項表現になっており，複合オブジェクトの記述も容易である．

O-項では，基本オブジェクト，オブジェクト変数，さらにそれらにオブジェクト構成子と呼ばれる関数を組み合わせて(場合によっては，異なる複数の関数を多重に用いることも可能)，オブジェクト識別子を構成で

きる．特に，このようにして得られたものを(項の表現を取ることより)オブジェクト識別子項という．たとえば，後述する $add(E,\ P)$ は，オブジェクト変数 E, P, オブジェクト構成子 add を用いて得られるオブジェクト識別子項である．この識別子項がどのように用いられるかを，O-項

有向枝:E[始点→節点:X, 終点→節点:Y]

で表現される関係によって構成されているグラフを対象として示す．このグラフに対して，節点間のパスの集合は，次のような識別子項を用いたO-項によるルールによって記述される．ただし，$f \Leftarrow g$ は「g ならば f が成立する」という含意を表す．

パス:$add(E,nil)$ [始点→X, 終点→Y]⇐

有向枝:E[始点→節点: X, 終点→節点:Y].

パス:$add(E,P)$ [始点→X, 終点→Y]⇐

有向枝:E [始点→節点:X, 終点→節点:Z],

パス:P[始点→節点: Z, 終点→節点:Y].

この例において，$add(E,nil)$ は $add(E,P)$ と同様オブジェクト識別子項である．最初のルールは，空のパス集合に有向枝を加えることで，1本の有向枝からなるパスが構成されることを示し，2番目のルールは，すでに構成されているパスに隣接する有向枝を1本ずつ加えることにより新たなパスの集合が得られることを示す．この二つのルールによりすべてのパスからなる集合が表現される．ここで，すべてのパスの集合を求める問合せ

パス:P[始点→節点:X, 終点→節点:Y]?

を発すると，

パス:$add_{1,k}$[始点→節点:a_i, 終点→節点:b_i]

の形をしたO-項で構成される解の集合が得られる．ただし，$add_{1,k}$ は $add(e_1, add(e_2, \cdots, add(e_k, nil), \cdots))$ であり，また，一般に，e_j は有向枝の識別子，a_i, b_i は節点の識別子である．

このように，識別子項はO-項の表現能力を高めるうえで非常に有効な働きをしている．

3.4 F-論理[13]

Kifer と Lausen は，人工知能の研究においてよく用いられるフレームの概念をベースにして，O-論理をさらに発展させたフレーム論理(F-論理と省略してよば

れる) を提案した[13]. 実際, O-論理の表現はすべてF-論理で記述できる. フレーム記述のように, F-論理のもとになっているF-項においては, 特に型と実体の相違は強調しない. また, オブジェクト変数を有するオブジェクト識別子項がラベルの位置に現われることができるので (つまり, ラベルもオブジェクトと扱う), メソッドの宣言的な定義も可能である.

F-項における事実の記述はO-項とほぼ同様に行えることより, ここでは, F-項を用いたルール記述がメソッド記述をも可能にしている例を示すことにする.

X[子供 (Y) → {Z}] ⇐

　人:Y[子供_obj → 子供 (Y)[メンバ → { 人:Z}]],

　人:X[子供_obj → 子供 (X)[メンバ → { 人:Z}]].

この記述において, '人' は型, 'X', 'Y' および 'Z' はオブジェクト変数であり, '子供_obj' および 'メンバ' は属性を表すラベルである. また, '子供 (X)' および '⇐' の右辺に現われている '子供 (Y)' は, X, Y それぞれの子供集合オブジェクトを表す (オブジェクト) 識別子項である. したがって, たとえば, この識別子項の変数に具体的な値を代入した '子供 (花子)' は, 「花子のすべての子供を表すオブジェクト」 という意味がある. '⇐' の右辺の第1項が Y の子供の集まりを, 第2項が X の子供の集まりを表しており, その両項に共通の変数 Z を配置することによって, 右辺全体では「X と Y が共通してもつ子供の集合」を表していると考えれる.

ここで注目すべきことは, '子供 (Y)' という項が ⇐ の左辺にも現れ, 特に属性ラベルの位置に記されていることである. この項により, 上記のルールは, 「各人 X に対して, ある人 Y (が入力として与えられると) との間に共通してもつ子供の集合を与える」 メソッドを記述していると解釈することが可能である.

このように, 出現場所に対応して意味を変えることにより, ラベル自身もある性質をみたすオブジェクトとしてみなすことが可能である.

以上のようにF-項の記述能力は高階になるが, O-項の場合と同様に, 意味論的には一階で収まる工夫がされている[8].

3.5 DOT — ドット記法に基づく拡張項[21]

一つの型に関する属性を特定するにはさまざまな方

法があり．それにともなって型間の継承にもさまざまなパターンが生ずる．たとえば，「学生 IS-A 人間」というIS-A関係と，「人間は組織に所属する」という人間の所属に関する属性が与えられていると仮定しよう．このとき，学生の所属に関する属性として，「学生は大学に所属する」ということが明記されていれば，「大学 IS-A 組織」という関係が継承によって成立すると考えられる(厳密に論ずると問題が生ずる場合がある)．ところが，もし，学生の所属に関する属性が明記されていなければ，学生に関しては，「学生は組織に所属する」という関係が得られるに留まる．

そこでDOT[21]においては，学生の所属に関する属性が明記されていなくても，仮想的にそれを表現するオブジェクト[3](実際は，型か実体である)を導入し，そのオブジェクトをも媒体にし，継承関係を用いて，新たなオブジェクト間のIS-A関係を導くことが可能である．このように，型と実体を区別せず，しかも，仮想的な存在も許すオブジェクト表現を導入することにより，オブジェクト間の継承あるいはIS-A関係を簡潔に記述することを目指してドット表現DOTが提案された．

DOTでは，オブジェクトa[4]の属性pの属性値は，明記されているか否かにかかわらずオブジェクト$a.p$と抽象的に表す．これを，DOT式と呼ぶ．ドット表現は，従来関係データベースなどにおいて属性値を表すのに使用されてきたが，DOTでは，DOT式自体をオブジェクトとして取り扱う．上記の例に対しては，‘学生.所属’という仮想的なオブジェクトの記述が可能であり，それを用いて，学生の所属に関する属性が明記されていなくても，「学生.所属 IS-A 人間.所属」という関係を記述することが可能である．従来の方法では，‘学生.所属’というオブジェクトの表現は無理であった．ここで，オブジェクト間のIS-A関係の継承は

[3]参考文献21)では，「オブジェクト表現」と「オブジェクト」を区別しており，ここでの「オブジェクト」は参考文献21)の「オブジェクト表現」に対応している．

[4]DOTでは型と実体を区別しないために，すべてのオブジェクトは集合で表現される．この意味では$\{a\}$とすべきであるが，以下，集合の要素が一つの場合は読みやすさのために$\{,\}$を省略している．

次の一文で簡潔に表現することができる.

a IS-A b ならば $a.p$ IS-A $b.p$

また. 将来「学生が大学に所属する」ことが明記されれば、その属性値が IS-A 関係で次のように記述される.

大学 IS-A 学生. 所属

さらに

大学 IS-A 学生. 所属.
学生. 所属 IS-A 人間. 所属
(' 学生 IS-A 人間 ' と継承より),
人間. 所属 IS-A 組織

の三つの関係より、IS-A 関係の推移性を用いて ' 大学 IS-A 組織 ' が導出される仕組みになっている.

DOT 表現においては、型と実体の区別をなくしたために、IS-A 関係は、通常の IS-A 関係に加え、集合とその要素の関係の両方を含むような拡張的な関係になっている.

次に、項の表現を例で示すことにする. たとえば、DOT では、' 人 ' を次のように記述することができる.

人 (親→人,
名前→文字列,
年齢→数,
性別 ={ 男性, 女性 })

矢印の方向で IS-A 関係が表され (たとえば、「人. 親 IS-A 人」), = は、双方向の IS-A 関係を表す. 上記の例では、人の性別が男性と女性しかないことより双方向の IS-A 関係が成り立つ. さらに、'A 君 ' という ' 人 ' は次のように記述できる.

A 君: 人 (名前 = 一太郎,
年齢 =24,
性別 = 男性,
親 ← 太郎,
親 ← 花子)

上記の項により表現される IS-A 関係の一部を表したものが図 -2 である. IS-A 関係を表す実線で示したアークと、属性に対応する破線で示したアークがあり、これらをおのおの IS-A リンク, l リンクと呼ぶことにする. オブジェクト a の属性 l の属性値は DOT 式 $a.l$ で表され, その抽象的に表現されたオブジェクトからの

IS-A 関係により属性値の正確な表現が行われている．たとえば，‘A 君’の‘親’属性の属性値をひとまず抽象的に‘A 君. 親’としておき，「太郎 IS-A A 君. 親」といった IS-A リンクを引く．図中の点線で示したアークは，継承によって生じた IS-A 関係を表す．

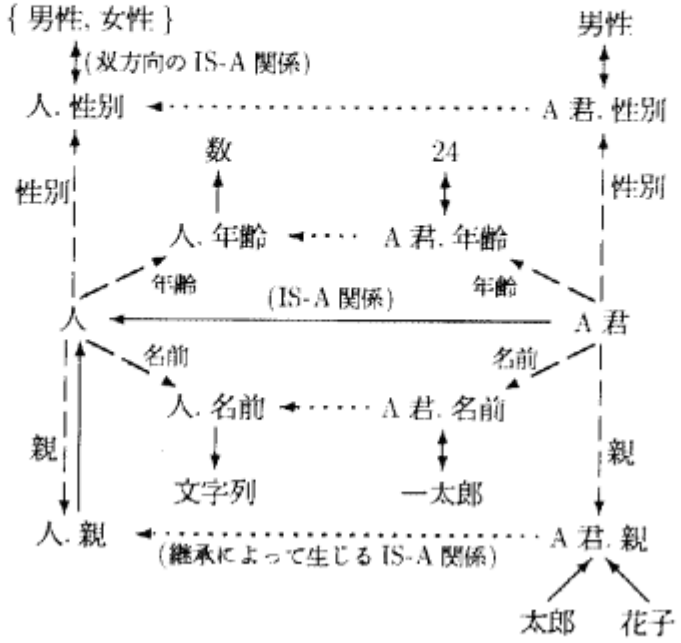

図-2 DOT 項により表現される IS-A 関係（一部）

IS-A 関係は順序関係であり，IS-A 関係を辿ることにより，新たな IS-A 関係が導かれる．**図-2** では，たとえば，‘太郎’から‘A 君. 親’，‘A 君. 親’から‘人. 親’，‘人. 親’から‘人’という IS-A 関係を辿ることにより，「太郎 IS-A 人」という IS-A 関係が導かれる．

また，項表現上の問合せとして，‘A 君’の‘親’が何であるのかを知りたいとき，‘A 君. 親’からの IS-A リンクを辿れば，‘A 君’の‘親’は‘人’であることが分かり，また，何が‘A 君’の‘親’であるのかを知りたいときは，‘A 君. 親’に入る IS-A リンクを逆に辿れば，‘太郎’と‘花子’が‘A 君’の‘親’であることが分かる．

以上の例では示さなかったが，オブジェクト変数を用いた DOT の表現も可能であり，また，**4.2** に例示するようにルールも容易に記すことができる．以上 DOT について例を中心に述べたが，項によって表される意味についての形式的に厳密な議論は参考文献 21) に記述されている．

4. 能動的オブジェクトとしこの演繹データベース

演繹データベース(DDB)がどのような言語で記述されているにせよ，それは事実(ファクト)とルールから構成されており，知識のような一つの(抽象的)実体を表現していると考えることができる．言い換えれば，DDBは知識のあるモジュールを表現していると考えられる．そこでこの章では，前章で述べたDDBの拡張とは異なった立場でオブジェクトをとらえる．つまり，DDB自体を一つのオブジェクトと考え，そのようなDDBの集合の性質を考える．言い換えればDDBのオブジェクト指向パラダイムへの埋め込みである．DDB間には，オブジェクト指向の文脈で考えられる継承関係や上書き機能を自然に導入できる．また各DDBは独立した質問処理能力をもっているので，DDBの集合に対する質問は，オブジェクト間の協調問題解決として考えられる．本章ではまず，DDBを階層構造に組み込み，次に能動的オブジェクトとしての質問処理を考える．この応用としては，(モジュール化された)知識ベース，CAIでの各人の学習の発展過程，あるいは専門家システムでの仮説推論などが考えられる．

4.1 拡張の課題

DDBはそのままではオブジェクトとして定義されていないし，他のDDBとの関係も考えられていないので，いくつかの拡張が必要とされる．

- オブジェクトの表現

DDBを特定するためのオブジェクト識別子や，処理を起動するメソッドを付加する．オブジェクトの単位としては，DDBや述語定義が考えられる．メソッドにもいくつかの指定法が考えられている．

- 階層関係

オブジェクト間の知識の冗長性を減少させるために，それらの間の関係を表現し，動的継承を可能とする．大域的な知識と局所的な知識の切り分けも必要となる．

- 協調問題解決

各オブジェクトに対する問合せを，オブジェクト間のメッセージ・パッシングによって，協調的に実行する．

- その他

局所的な知識の隠蔽，異種言語の共存など，さらに多くの拡張が考えられる．

DDB を管理システムも含めて考えると，同時実行制御，障害回復や永続性管理は，個々の DDB がもっているので，DDB の集合の場合，それらの同期をとることが重要になるが，ここでは触れない．

論理型言語で記述されている知識をモジュール化することは，これまで，論理型言語の研究で同様の研究は数多くなされているが，DDB に関しては少ない．ここでは参考文献 20) に沿って説明する．

4.2 階層構造に組み込んだ演繹データベース

DDB(あるいは述語定義) の集合を階層構造に組み込むためには，個々の DDB をオブジェクトとして識別するためのオブジェクト識別子と，階層構造を表現するためのオブジェクト間の順序関係が必要となる．事実とルールの集合を R とすると

I: オブジェクト識別子の集合

ρ: I から R への関数

$\preceq$: I の半順序関係

の三つ組によって，オブジェクト識別子をもった DDB の集合を階層構造に組み込むことができる．たとえば，**図 -3** はそのような集合を示している．$K_i(1 \leq i \leq 7)$ がオブジェクト識別子で，各リンクの上下関係が DDB 間の関係を示している．

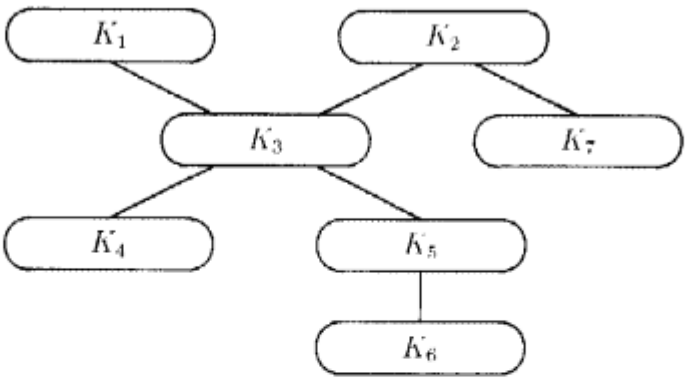

図 -3 DDB の階層構造

この階層構造が何を意味しているかは，たとえば本を考えてみるとわかりやすい．K_i が本の i 章に対応し，その内容が DDB で表されていたとすると，それは章間の知識の継承関係となっている．したがって，K_i に対応する DDB $\rho(K_i)$ の意味は，$S_i = \{K_j | K_j \leq K_i\}$ とすると，$\cup_{K_j \in S_i} \rho(K_j)$ と考えることができる．た

だし，各DDBがサブゴールに対して閉じているかどうか，また，DDBが否定などを含み層状化などの制限を持っていた場合，その制限をいかに保証するかなど，なんらかの(動的)検証機能が必要とされる．メソッド(述語)定義に，いわゆる水平分散の禁止のような制限を導入すれば，上書き機能の導入も同様に自然に行える．

これが**3.**の，オブジェクト識別子をもった拡張項にも適用できることは簡単に分かる．たとえばF-論理におけるルール

人[同世代(X)↠{X}]

人[同世代(X)↠{Y}]⇐ 人:X[親→{Z}],
人:Z[同世代(Z)↠{W}],
人:W[子供→{Y}]

あるいは，DOTにおけるルール

X(同世代←X,
同世代←X.親.同世代.子供)⇐X:人

は，メソッド`同世代'をもった`人'オブジェクトを定義している．継承機能をもったオブジェクト(型)の半順序関係は，(F-論理の場合)束，あるいは(DOTの場合)オブジェクト内で定義されており，ρ(人)が上記ルールを値とするようにできる．さらに関数ρは，新たなオブジェクト識別子によって，項中に現われない新たなオブジェクトを定義することもできる．

4.3 メッセージ・パッシングに基づく質問処理

図-3の例で，各オブジェクトは，前節の意味論にしたがって，必要に応じルールを上位オブジェクトから動的に継承する独立したDDBの質問処理系であるとする．外延データベース(EDB)は，必要に応じて水平分割されて各オブジェクトによって管理されているとする．

K_5のオブジェクトに質問することを考えよう．K_5はK_1, K_2, K_3を継承しているので，その質問処理に必要なルールは，それらの中でその述語を定義している最下位オブジェクトに動的に継承される．たとえば，K_1とK_3で同一述語が定義されていた場合，K_3が，K_1でのその述語の定義を継承する．

再帰質問処理の戦略[19]としてHCT/R[18]のような上昇評価を考えよう．ゴールの束縛情報の伝達によって，探索空間を狭める変換ルールは再帰関係で閉じた

コンポーネント(評価可能)単位に新たに生成される．それは，該当述語を定義する最下位オブジェクトと特化関係をもつオブジェクトとして生成されており，初期節(シーズ)を除けば，同じ束縛パターンの質問に再利用できる．このオブジェクトは，上昇評価のときに，関連したオブジェクトに対して，(EDBの処理などの)コーディネータとしての役割を果たす(図-4のC_1～C_4)．上昇評価はコーディネータ・オブジェクトに初期節を与えることによって起動され，EDBの処理は，各オブジェクトの局所的な処理とコーディネータ処理の二つに分割される．最終的な答えは，最下位にいるオブジェクト(この場合はC_1)に作成される．この質問処理は，オブジェクト間のメッセージ・パッシングによって実行されるが，そのほかに，従来のデータベース管理が行っていた同時実行制御や記憶域管理も各オブジェクトが自律的に行うことができる．

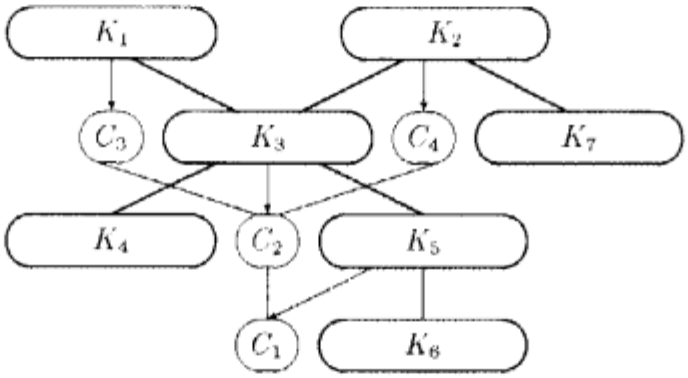

図-4　質問処理のコーディネータ・オブジェクト
(→は特化関係)

このような上昇評価を目指した試作も行われている[20]．また，下降評価のものとしては，並列処理で述語単位のプロセスを動的に生成して実行する例がある[11]．これまで質問処理の最適化は単一のDDBを対象にすることが多く，今後新しいアルゴリズムの開発が期待される．

5. 今後の展望

演繹データベース(DDB)の拡張には，演繹・オブジェクト指向データベース(DOOD)への拡張以外にも多くのアプローチが行われている．この章では，上記以外のアプローチを簡単に紹介し，さらに(DOOD89を含

んだ) 今後の研究動向を概観する.

5.1 その他の拡張

DDB の拡張に関する研究は, この数年急速に広がっており, 以下主要なアプローチを紹介する:

• 拡張項に基づいた論理型言語

論理型言語に集合のグルーピング概念を導入したものに MCC の LDL[24], 複合オブジェクトを扱ったものに INRIA の COL [1], 多値関係を扱ったものに ICOT の CRL[22] がある. さらに, O-論理や F-論理の流れの中で, 構文論的には高階ながら意味論的には一階となっている HiLog [8] がある. LDL, COL, F-論理から HiLog への変換も示されており, 今後の展開が期待される.

• データベース・プログラミング言語

データベース言語とプログラミング言語を統合しようというアプローチとして, データベース・プログラミング言語 (DBPL) がある [12]. これは論理型言語だけでなく, 関数型言語からのアプローチも含んでいる. オブジェクト指向データベース (OODB) の目的の一つに, インピーダンス・ミスマッチの解消があったが, この点は DBPL とまったく一致しており, 上記で紹介した多くの言語も DBPL と考えることもできる.

• 制約パラダイム

論理型言語の拡張として, より複雑な問題を制約として効率的に表現し, 固有の制約評価系によって解決することを目指したものとして制約論理型言語 [16] がある. このパラダイムに基づき, ψ-項などをその一つとして位置づけたり [7], データベース分野に応用しようとするアプローチ [10], [23] もある.

5.2 研究の動向

DDB の研究は, 1977 年のフランスの Toulouse で行われた *The International Symposium on Logic and Data Bases* に始まり, 1986 年の *Workshop on Foundations of Deductive Databases and Logic Programming* で大きな転機を迎えたと考えられる. 同じ 1986 年には, OODB に関し, *The International Workshop on Object-Oriented Database Systems* と *The Symposium on Object-Oriented Programming Systems, Languages and Applications* (OOPSLA), 専門家システムとの統合に関し *The First International Confer-*

ence on Expert Database Systems. といずれも第一回目が始まっており，また翌年の1987年には *Workshop on Database Programming Languages* の第一回目が開催された．このように，1980年代後半は，新しいデータベースに向けてのマルチ・パラダイムの時代の幕開けともいえる．

このような背景の中で，昨年12月に京都で開催された *The First International Conference on Deductive and Object-Oriented Databases* (DOOD89)[15] は，次世代データベースの有力な一つと考えられるDOODに焦点を絞ったもので，新しいデータベースの枠組に向けて，DDBとOODBの両分野からの活発な議論が行われた．「OODBシステム宣言」[4] のほかにも，興味ある発表が多くなされた．今後のデータベースの研究分野の拡大の中で，**2.** で述べたような，DOODという大きな枠組は，さらに位置づけを増してくるように思える．

6. おわりに

演繹・オブジェクト指向データベース (DOOD) の研究は，演繹データベースの拡張の中でも，もっとも新しいものの一つといえるが，この研究は本稿で紹介したように，現在データベース分野で非常に活発に行われてきている．このDOODは，**2.** で説明したように，研究のアプローチの枠組として考えることができ，一意的にその概念を定義するものではない．問題領域，応用分野によってさまざまなDOODが，新しいデータベースとして共存しうると思われる．

謝辞　末筆ながら，草稿に対し貴重なコメントをいただいた本特集の著者の方々およびシャープ(株)塚本昌彦氏，日頃DOODについて有益かつ刺激的な議論をしていただいている，ICOTの「演繹＋オブジェクト指向データベース・ワーキンググループ」(DOO-WG)のメンバーの方々に心より感謝の意を表す．また著者の一人西尾は，日頃御指導いただく京都大学長谷川利治教授および茨木俊秀教授，ならびに大阪大学宮原秀夫教授に深謝の意を表す．

参 考 文 献

1) Abiteboul, S. and Grumbuch, S.: COL: A Logic-Based Language for Complex Objects, *Proc. EDBT*, Venice, pp.271-293 (1988).

2) Aït-Kaci, H.: An Algebraic Semantics Approach to the Effective Resolution of Type Equations, *Theor. Comput. Sci.*, vol.45, pp.293-351 (1986).

3) Aït-Kaci, H. and Nasr, R.: LOGIN: A Logic Programming Language with Built-in Inheritance, *J. Logic Prog.*, vol.3, pp.185-215 (1986).

4) Atkinson, M., Bancilhon, F., DeWitt, D., Dittrich, K., Maier, D. and Zdonik, S.: The Object-Oriented Database System Manifesto, *Proc. DOOD89*, pp.40-57, Kyoto (Dec., 1989).

5) Bancilhon, F. and Khoshahian, S.: A Calculus for Complex Objects, *Proc. ACM PODS*, pp.53-59, Cambridge (1986).

6) Beeri, C., Nasr, R. and Tsur, S.: Embedding ψ-term in a Horn-clause Logic, *Proc. the Third International Conference on Data and Knowledge Bases*, pp.347-359, Jersalem (June, 1988).

7) Beringer, H. and Porcher, F.: A Relevant Schema for Prolog Extensions: CLP (Conceptual Theory), *Proc. ICLP*, pp.131-148, Lisbon (June, 1989).

8) Chen, W., Kifer, M. and Warren, D.S.: HiLog as a Platform for Database Language, *Proc. the Second International Workshop on Database Programming Language*, pp.121-135, Gleneden Beach, Oregon (June, 1989).

9) Chen, W. and Warren, D.S.: Abductive Reasoning with Structured Data, *Proc. the North American Conference on Logic Programming*, pp.851-867, Cleveland (Oct., 1989).

10) Gallaire, H.: Multiple Reasoning Style in Logic Programming, *Proc. FGCS'88*, pp.1089-1099, Tokyo (Nov.28-Dec.2, 1988).

11) Hulin, G.: Parallel Processing of Recursive Queries in Distributed Architectures, *Proc. VLDB*,

pp. 87-96, Amsterdam (Aug., 1989).

12) Hull, R., Morrison, R. and Stemple, D. (ed.): *Proc. the Second International Workshop on Database Programming Language*, Gleneden Beach, Oregon (June, 1989).

13) Kifer, M. and Lausen, G.: F-Logic — A Higher Order Language for Reasoning about Objects, Inheritance, and Schema, *Proc. ACM SIGMOD*, pp.134-146, Portland (June, 1989).

14) Kifer, M. and Wu, J.: A Logic for Object-Oriented Logic Programming (Maier's O-Logic: Revisited), *Proc. ACM PODS*, pp.379-393, Philadelphia (Mar., 1989).

15) Kim, W., Nicolas, J.-M. and Nishio, S. (ed.): *Deductive and Object-Oriented Databases, to be published*, North-Holland (1990).

16) Jaffer, J., Lassez, J.-L. et al: Constraint Logic Programming, *IEEE SLP*, San Francisco (Aug. 31-Sep.4, 1987).

17) Maier, D.: A Logic for Objects, *Proc. the Workshop on Foundations of Deductive Databases and Logic Programming*, pp.6-26, Washington DC (1986).

18) Miyazaki, N., Yokota, K., Haniuda, H. and Itoh, H.: Horn Clause Transformation by Restrictor in Deductive Databases, *J. Inf. Process.*, vol.12, no.3 (1989).

19) 西尾, 楠見: 演繹データベースにおける再帰的な問い合わせの評価法, 情報処理, vol.29, no.3, pp. 240-255 (1988).

20) 高橋, 横田, 横塚, 梶山: 拡張項と階層構造に基づいた演繹データベースの試作, 情報処理学会全国大会, 2C-7, 福岡, 10 月 (1989).

21) 塚本, 西尾, 長谷川: オブジェクト指向の概念を導入した論理データベースのための項表現 DOT, *Proc. Advanced Database System Symposium '89*, Kyoto, (Dec. 1989).

22) Yokota, K.: Deductive Approach for Nested Relations, in *Programming of Future Genera-*

tion Computers II, eds. by Fuchi, K. and Kott, L., North-Holland (1988).

23) 横田: オブジェクトの形式化とスキーム、電子情報通信学会データ工学研究会、*DE88-25*、京都、11月 (1988).

24) Zaniolo, C.: Design and Implementation of a Logic Based Language for Data Intensive Applications, *Proc. ICLP/SLP*, pp.1666-1687, Seattle (1989).