データベース・システム 85－7－1
プログラミング－言語・基礎・実践－ 4－7－1
（1991. 10. 4）

パネル討論：永続的プログラム言語とオブジェクト指向データベース

型理論に基づく次世代データモデル実現の可能性

大堀 淳　　沖電気工業（株）関西総合研究所

1 次世代データモデルのためのパラダイム

近年オブジェクト指向データベースが次世代のデータベースシステムとして注目を集めているが、それについての形式的定義は、筆者の知る限り未だ存在しない。最近オブジェクト指向データベースの「満たすべき条件」に関する幾つかの議論があるが、何を持ってオブジェクト指向データベースと呼ぶかは未だに議論の対象であり、何がこれまでと違って本質的に革新的なのかについても十分な説明が与えられているとは言い難いと思われる。新しいデータモデル（オブジェクト指向であれ何であれ）が、関係データベースシステムが経験したような実用化技術の蓄積を享受し、次世代のデータベースシステムの基礎となり得るためには、そのモデルに対して、関係データモデルの記述言語が持っている形式的で整合性のある意味論に対応するような形式的基礎が確立されることが必須であると考える。これが本パネルでの筆者の基本的な立ち場である。

データベースの新しいモデルの可能性は一つとは限らず、次世代データベースのモデルの形式的基礎を与えるためのアプローチもまた複数存在し得る。そのなかで特に重要でかつ有望と考えられるものは、プログラム言語とデータベース両方の概念を統合した理論の構築である。オブジェクト指向データベースは、少なくとも歴史的には [2]、言語システムとして提案されたものであり、オブジェクト指向データモデルがあるとすれば、それはプログラム言語とデータベースの概念を統合したものとなると考えるのが自然である。また、そのような統合されたシステムは、単なるデータ検索システムにとどまらず、次世代の情報処理システムの中核とり得ると期待できる。

この統合の基礎となり得る理論の一つとして、筆者は、プログラム言語の型理論（例えば [4] を参照）が有力であると考える。型理論は、汎用の計算能力と複雑なデータ構造を統一して制御する理論であり、プログラム言語、データベース、オブジェクト指向等の諸機能の統合を目指す次世代データベースの基礎理論として適していると考えらるからである。ここで留意していただきたい点は、本稿では型を、プログラム言語理論の歴史の中で洗練され確立した概念（およびその一般化）として用いるという点である。オブジェクト指向データベースの文献等では、時おり、「型」や「クラス」およびそれらの区別に関する議論を見かけるが、それらの議論が有意義に行なえるためにも、この厳密な態度は重要であると考える。さらに、それら直観的な「型」や「クラス」、「スキーマ」等の概念間の関係を明らかにし、それらを統合する枠組としても、形式的定義に基づく型理論は有用であると信じる。

型理論にはそれ特有の弱点があり、オブジェクト指向データベースや種々の次世代データベースで提唱されているすべての機能を自然に表現する見込みは未だ立っているわけではない。しかし筆者は、近年の型理論の急速な発展を考慮すれ時、型の概念をさらに洗練し拡張することによって、多くの未解決に問題も解決される可能性が十分にあり、型理論に基づくデータベースの実現は一つの有望な研究テーマであると信じる。

本パネルでは、以上の立場から見た永続言語の特徴、永続言語へデータベース機能、およびオブジェクト指向プログラミングの機能を導入する上での課題についての筆者の考えを論じる。

2 型理論からみた永続言語の本質

永続データという用語は、個々のプログラムの実行とは独立に、長期にわたって存在するデータを意味する。しかし型理論から見たデータの永続性の本質は、その生存期間の長さではなく、それらがプログラムの静的環境の外部にあるデータという点にある。（この点で、永続データは例えば、分散環境下の他のシステムのデータなどと似た性質を持つ。）この外部性のため、プログラム言語の静的型システムは、永続データに関する型情報を静的に決定することができず、動的な型のチェックがどうしても必要である。永続データを扱える型システム設計の要点は、静的型システムの種々の性質を損なうことなく、動的型チェックの機構を導入することである。PS－algol[3] をはじめとする永続プログラム言語の研究の最大の貢献は、この機構を組織的に型つきプログラム言語に導入したことでる。それは、永続データを、型情報を含んだ特殊な型を持つデータとして型システムに導入し、永続データを扱う際に必要な動的型チェックを局所化する機構である。このようなデータは、型理論の枠内では dynamic 型を持つデータと呼ばれ、文献 [1] で形式化されている。

以上の諸研究によって明らかになった重要な点の一

つは、型つき言語にとって、データの永続性は他の性質との独立性が高いという洞察である。永続プログラム言語の研究者によって提唱されたスローガン「直交永続性」は、この性質を根拠とするものである。このことは、データベースやオブジェクト指向プログラムの種々の機能が型つき言語において形式化できれば、それらと永続データを取り扱うメカニズムとの統合は比較的容易であることを意味する。

3 永続言語の次世代データモデルへの拡張

以上論じた永続データを取り扱う機能は、プログラム言語のデータ操作能力を外部の永続データへ拡張する機能であり、それ自体ではデータベースシステムに必要な機能を実現することはない。型理論に基づくデータモデル実現のためには、型システムを拡張して、データベースシステムにとって必要な機能を表現可能にする必要がある。このためのに解決しなければならない課題は数多く存在するが、その中で特に重要と考えられる点には以下のものがある。

1. 複合オブジェクト操作システム
   次世代データベースにとっても、関係代数に匹敵するエレガントで強力な操作系が望ましいのは明らかである。オブジェクト同一性の機能を含まない、値としての複合オブジェクトの操作システムを形式的に定義する試みはいくつかあるが、オブジェクト同一性の機能を持つ複合オブジェクト操作系に対する、一般性のあるエレガントな定式化は、未だ成功していないようである。これは、最近話題となっているオブジェクトの「ビュー」の機能を実現する上でも重要な研究課題である。

2. データベースの設計理論
   関係データベースに比べて飛躍的に複雑さが増すと予想される次世代データベースにとっては、より緻密な設計理論が必要になってくることは明らかである。この場合もやはり、オブジェクト同一性の機能を持つ複合オブジェクトに対する実用性のあるしかも数学的にエレガントな設計理論の確立が課題である。

   関係データベースにおいて、エレガントな操作系や設計理論が可能であったのは、それが数学的基礎の基に成り立っていることによることはあきらかであり、以上二つの目的を達成するためには、データモデルの形式的基礎が必須である。

3. 大容量型 (bulk data types) についての理論
   データベースは、型理論では、集合やリスト等の大容量型を持つデータと見なせる。これら大容量型の性質に関する研究は、その重要性にも関わらず、これまであまりなされていない。

4. オブジェクト指向プログラムの機能の導入
   オブジェクト指向プログラミングの諸概念を型システムの中で表現する研究は、現在、型理論の研究でもっとも盛んに研究されているテーマの一つである。これらの研究成果は、次世代データベースシステムにとっても極めて有用である。

5. 実装理論
   データモデルの理論は効率的でかつ信頼性のある情報処理システムを実装するための理論であり、(何らかの意味で) 実装技術に貢献し得ない理論は意味に乏しい。型理論はこの分野でも大きな貢献をする可能性を持っていると信じる。これは、現実からかけはなれた根拠に乏しいこの主張と、受けとられかねないので、筆者の最近の研究であるが、敢えて例をあげることにする。文献 [5] では、レコードのラベルのアドレス情報 (あるいはオブジェクトに対するメソッドのアドレシング) を静的に決定する方法を与えている。この方法は、最新の型推論理論によって可能になったものである。

4 結論

型理論によりるオブジェクト指向プログラミングの諸概念やデータモデルの概念の基礎づけの研究は始まったばかりであり、実現されていない課題は多く存在する。しかし筆者は、型理論が、プログラム言語やオブジェクト指向の考えを統合した次世代のデータモデルを設計するうえでの重要な理論になり得ると信じている。これまで型理論は、データベースの研究者からは注目されることはほとんどなかったが、このパネルを通じて一人でも多くのデータベースの研究者がプログラム言語およびその型理論に関心を持たれることを、またプログラム言語の研究者がデータベースに関心を持たれることを期待したい。

参考文献

[1] M. Abadi *et. al.* Dynamic typing in a statically-typed language. In *Proc. ACM POPL*, 1989. [2] G. Copeland and D. Maier. Making smalltalk a database system. In *Proceedings of ACM SIGMOD*, pp. 316–325. ACM, June 1984. [3] M.P. Atkinson *et. al.* An approach to persistent programming. *Computer Journal*, 26(4), 1983. [4] J.C. Mitchell. Type systems for programming languages. In *Handbook of Theoretical Computer Science*, chapter 8, pp. 365–458. MIT Press/Elsevier, 1990. [5] A Ohori. A compilation method for implicitly typed polymorphic record calculi. In *Proc. ACM POPL*, 1992 (to appear).