解　説

1.1 次世代データベースシステムとしてのオブジェクト指向データベースシステム†

増　永　良　文††

1. はじめに

世界で最初のデータベース管理システム (Database Management System, 以下 DBMS と略す) は，1963年にアメリカの GE (General Electric) 社が発売した IDS (Integrated Data Store) といわれている．それは，後に CODASYL のネットワークデータモデルとして広く世界に流布することとなった．1968年には，アメリカ IBM 社からハイアラキカルデータベースを管理・運用できる DBMS として IMS が発売されている．1970年には，データベース界で画期的な事が起こった．リレーショナルデータモデルの提案であった[1]．これは，アメリカ IBM 社のサンホゼ (San Jose, カリフォルニア州) 研究所のコッド (Edger F. Codd) の発案によるもので，ネットワークデータモデルやハイアラキカルデータモデルに比較して，(1)モデルの平明さ，(2)高いデータ独立性，(3)非手続き的データ操作言語の提供，(4)分散型データベースへの適合性，(5)ロジックとの親和性などの特長を有し，データベースシステムの流れを大きく変えるものとなった．実際 1970年代に世界の多数の機関が，リレーショナル DBMS の研究・開発に取り組んだ結果，1980年代に入ると商品化され，現在リレーショナル DBMS は広く受け入れられることとなった．

さて，ネットワーク，ハイアラキカル，そしてリレーショナルと発展してきたデータベースシステムには共通した特徴がある．それは，いずれもその応用をビジネス分野においてきたことである．つまり，それらDBMSが管理・運用の対象としたデータベースは，たとえば組織体の人事管理データや会社の売上データ，あるいは在庫データであった．そもそもデータベースは1960年代後半に意識されたソフトウェア危機と深く関連しており，組織体がデータベースを構築する最大の理由の1つは，データを一元管理することにより組織体のソフトウェアの生産性を向上させることであった．したがって上述のリレーショナルデータベースの特長が広く認識されると，ビジネスデータ処理のための DBMS は時代の大きな流れとして，リレーショナル DBMS へと移行したのである．

このような展開の中，1980年代に入ると，DBMS をビジネスデータではなく，CAD/CAM (Computer Aided Design/Manufacturing) データ[2]やマルチメディアデータといった先進的データベース応用に適用する試みが行われるようになった．実はこのような試行が，現在われわれが次世代データベースとして認識するに至ったオブジェクト指向データベースを生む母体となったのである．ただ，1つ注意しておかなければならないことは，当時はリレーショナルデータベースでそのような先進的データベース応用に対処するために，どのように改良するべきかを論じていたのであり，リレーショナルの不適性を暴こうなどとは考えてもいなかった点である．現在でも，たとえば POSTGRES[3] にみるように，そのような先進的データベース応用にもリレーショナルデータベースを拡張すれば事足りるとする主張もある．しかし，そのような先進的データベースにこそオブジェクト指向データベースシステムが適していることが広く認識されるに至っている．これにはさまざまな理由があるが，(1)データの構造が複雑，(2)データの操作も複雑，(3)デザイントランザクションのサポート，(4)マルチメディアデータのサポートといった先進的データベース要

† Object-Oriented Database System: The Next Generation Database System by Yoshifumi MASUNAGA (University of Library and Information Science, Department of Library and Information Science).

†† 図書館情報大学図書館情報学部

求にきわめて自然に，かつ効率よく対処できるのがオブジェクト指向データベースということである．ここで1つきちっと認識しておいて欲しいことは，「ビジネスデータ処理をもオブジェクト指向で」とは主張していない点である．ただ，将来ビジネスデータ処理分野のデータでも特にデータ構造の複雑性の高いところ，換言すればデータ処理要求を満たすためにリレーションのジョイン(join，リレーショナル代数の演算の1つ)を数多くとることを要求するような分野では，オブジェクト指向 DBMS が使われるようになるだろう．

2. 先進的データベース応用とリレーショナルデータベースシステム

2.1 リレーショナルデータベースとその意義

リレーショナルデータベースは，先述のとおり1970年アメリカ IBM サンホゼ研究所のコッド(Edger F. Codd)の提案による．データは，リレーションと呼ばれる表(table)として組織化，管理，表示される．従来のネットワークやハイアラキカルといったデータベースがコンピュータ内のデータアクセス機構，たとえばポインタやリストといったデータ記憶の物理的構造と直結したデータ構造をもっていたのに対し，数学の集合論をモデルの基盤としてもち，徹底的にフォーマルなところがネットワークやハイアラキカルデータベースと根本的に異なるところであった．高いデータ独立性，非手続き的データ操作言語の提供などの上記リレーショナルデータモデルの特長はここに源がある．このような理由から，ネットワークデータベースやハイアラキカルデータベースをリレーショナルデータベースと概念的に区別し，前者を第一世代のデータベース，後者を第二世代のデータベースと言ったりする[4]．

リレーショナルデータベースの意義は，上でも少し述べたが，その特長によりビジネスデータ処理分野でソフトウェアの生産性を向上したことにある．たとえば，リレーショナルデータベースでは，ANSI/X 3/SPARC のいうデータベースの三層スキーマを，第一世代のデータベースシステムに比較してきわめて高度にサポートできる．また正規化理論に基づくリレーションの第三正規形は，情報資源管理に道を開いた．ビジネス分野でのリレーショナルデータベースの果たす役割の大きさは，いくら強調してもし過ぎることはない．ただ，それを非ビジネスデータ処理分野のデータ処理に適用しようとしたときに問題があるというだけのことである．このことについては章を改めて述べる．

2.2 オブジェクト指向プログラミングと人工知能

次世代データベースとしてのオブジェクト指向データベースを語るうえで，オブジェクト指向プログラミングと人工知能研究に言及しておく必要がある．まず，オブジェクト指向プログラミングであるが，ここでは Simula 67 に端を発し Smalltalk-80[5] に代表されるプログラミング言語によるプログラミングを言っている．そこでは，従来型の手続き型のプログラミング(たとえば Fortran, Cobol, PL/I)とは異なり，実世界の対象物，すなわちオブジェクトがそのデータと振る舞いとともにモデル化される．その結果，実世界で生起するさまざまな事象を容易にシミュレーションしたり，またプログラムの生産性を向上させることが可能となった．類似の概念は人工知能の分野にもみられ，たとえばフレーム理論はその典型であろう．

実は，これら2つの概念とオブジェクト指向データベースは深く関わっている．第一に，オブジェクト指向プログラミングではプログラム中で定義したオブジェクトはその実行終了と同時に消滅してしまう．しかし，そのようなオブジェクトを二次記憶装置内に格納しておき，後に必要となったときに再び使用したい．換言すればプログラムで生成したオブジェクトを永続化(persistent)したいという自然な要求がある．一方人工知能では，エキスパートシステムに代表されるが，演繹に使用するデータをさまざまなデータベースからふんだんに取ってきたいという要求がある．データベースでは，プログラミング言語とデータベースをアクセスするためのデータベース言語のいわゆるインピーダンスミスマッチ(impedance mismatch)の問題がある[6]．それはネットワークやリレーショナルデータベースでは避けることができず，データベースの新しいパラダイムが求められていた．結論から言えば，オブジェクト指向データベースなる概念はそれらの共通の問題に解決を与えるものである．このことについては適宜説明を加えたい．

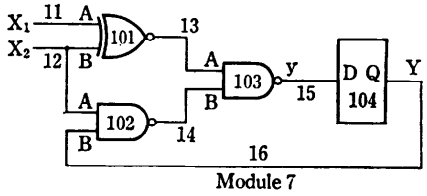
3. 先進的データベース応用と従来型データベースシステム

本章では，CAD/CAM，CASE，マルチメディアデータベース，ネットワーク管理といった先進的データベース応用には，ネットワークデータベースやリレーショナルデータベースといった従来型のデータベースではデータの組織化や管理・運用に問題があり，オブジェクト指向アプローチがそれらの問題点を解決し得ることを具体的に論じてみたい．

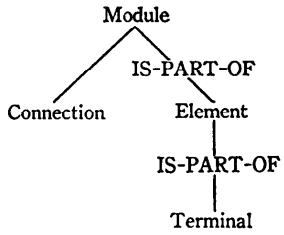
3.1 CAD データベースとオブジェクト指向

1983 年 IBM サンホゼ研究所のハスキン (R. Haskin) らはリレーショナルデータベースを使って論理回路をデータベース化し，管理・運用することを考えた[2]．図-1(a) にそのような回路の一例を示す．アイディアは，まずそのような回路の構造を解析し，データベース化への糸口を摑むことから始まった．結果から言えば，データベース化に当たりまず図-1(b) に示すような構造を認識する．つまり，回路をモジュールと呼び，それは 2 つの部品をもつ．1 つは素子 (element) で，もう 1 つは結線 (connection) である．したがって，モジュールは素子および結線と IS-PART-OF の関係で結ばれる．さらに注意してみれば素子は端子 (terminal) を部品としてもつ．したがって，素子なる概念の下に端子が IS-PART-OF の関係でぶら下がることとなる．そこで図中の各ノードに対応して 1 つずつリレーションを定義する．この例では 4 つのリレーション (スキーマ) モジュール (モジュール識別番号，モジュール名)，素子 (モジュール識別番号，素子識別番号，素子名)，結線 (モジュール識別番号，結線識別番号，結線名)，端子 (素子識別番号，結線識別番号，端子名，入出力) が定義される．図-1(c) に，図-1(a) に示された回路を具体的にインスタンスレベルで表示する．

さて，ここで注意しないといけないことは，まず例のようなきわめて小規模で単純な回路を表現するのにすら，4 つのリレーションと計 22 本のタップルが必要となることである．データベース化の対象となる回路がより規模の大きいものであれば，それを表現するのに必要なタップルの本数はとてつもなく多くなることが予想される．その結



(a) 簡単な論理回路 (モジュール 7) の例



(b) モジュールの構造解析

モジュール

モジュール識別番号	モジュール名
7	Module 7

素子

モジュール識別番号	素子識別番号	素 子 名
7	101	2 inp. XNOR
7	102	2 inp. NAND
7	103	2 inp. NAND
7	104	Latch

結線

モジュール識別番号	結線識別番号	結線名
7	11	X_1
7	12	X_2
7	13	
7	14	
7	15	y
7	16	Y

端子

素子識別番号	結線識別番号	端子名	入出力
101	11	A	I
101	12	B	I
101	13	XNOR (A, B)	O
102	12	A	I
102	16	B	I
102	14	NAND (A, B)	O
103	13	A	I
103	14	B	I
103	15	NAND (A, B)	O
104	15	D	I
104	16	Q	O

(c) モジュール 7 のリレーショナルデータベース表現

図-1 論理回路モジュールのデータベース化

果発生するきわめて重要な問題は，1つの回路を取り扱うのに原則的に上の4つのリレーションの結合 (join) をとらねばならないことである．よく知られているように，リレーショナルデータベースを使用して一番効率の落ちる点はリレーションの結合をとることにある．もちろん，リレーショナル DBMS にはこのような効率低下を避けるためのさまざまな最適化技法が組み込まれているが，結合演算は本質的に時間がかかりシステムの効率を著しく低下させる原因となっている．したがって最初からリレーションの結合演算を前提としたモデル化しかできないようなデータベース化は，問題があるといっても過言ではない．実際，上のリレーションの属性 (attribute) に着目すれば，リレーション素子の属性モジュール識別番号はリレーションモジュールの外部キー (foreign key) であり，図-1(b)の IS-PART-OF の関係が外部キーの関係で表現されていることに気づく．このことは，取りも直さず回路の検索や更新を行おうとすれば，一般に常に上記リレーションを外部キーを頼りに結合しなければならないということである．したがって，時間のかかる処理を余儀なくされている．二番目に注意しないといけないことは，データベースユーザには図-1(a)が提示されるのではなく，図-1(c)が提示されるということである．換言すれば，ユーザは図-1(c)に示されたリレーションの集合から，ユーザが処理の対象とした回路を的確に想像し，検索や設計の変更をリレーショナルデータ操作言語を使って行わなければならない．このことは対象としている回路が小規模なときはよいが，回路が大規模になるにつれて相当に複雑な操作となり，究極的には人間の能力の限界を越えてしまうことは想像に難くない．実際に図-1(a)で素子101と素子103を交換することを考えてみよう．それを実現するために図-1(c)で行うことは，リレーション端子の6本のタップルの結線識別番号値をしかるべき値に更新するということである．

さて，ここで読者によく考えてもらいたいことは，この回路をデータベース化するにあたりリレーショナルデータベースを使う必然性があったのかということである．確かにリレーショナルデータベースで上述のようにモデル化すればデータベース化は可能である．実際に，1980年の初め，ハスキンら[2]やストンブレーカーら[7]が考えたことは，リレーショナルデータベースシステムをどのように拡張すれば，上記のような構造をもったデータ，換言すれば一本のタップルでは表現できないようなデータ（これを以下複合データ (complex data) と呼ぶ）をモデル化できるかであった．しかし，筆者がここで声を大にして主張したいことはモデル化できるかできないかではなく，リレーショナルデータベースを用いて複合データのモデル化を行うのが適切か不適切かである．さらに考慮しないといけないことは，このような複合データに対する操作 (operation) の表現である．上例のように，素子の交換という CAD ではきわめて単純と思われる操作を実現するのに，リレーションのタップルを多数更新しないといけないという対応関係は自然ではない．

結論からいえば，上述のような複合データとその操作は，オブジェクト指向データベースを用いれば一発できわめて自然に表現できてしまう．このことは，次章でオブジェクト指向データベースの最も基本的な概念と定義を与えてから示したい．

3.2 マルチメディアデータベースとオブジェクト指向

従来の文字・数値データに加えて，テキスト，図，静止像，動画，音といったさまざまなデータを統合して1つのデータベースとし，それに統一的なインタフェースを介してデータの検索や生成・削除・更新を行えるデータベースシステムをマルチメディアデータベースシステムという．さて，マルチメディアデータベースを構築するにはテキスト，画像，音といったメディア（すなわち実世界記述のためのさまざまな記号系）の異質性 (heterogeneity) を克服した，異メディア間データの統合を考えなければならない．より具体的には，たとえばテキスト（これは文字列データ）と画像（これはビット列データ）をどう統合したらよいのであろうか．増永のオブジェクト参照法[8]によればテキストも画像も全てオブジェクトとみなすことにより統合が可能となる．マルチメディアデータベースを構築する考え方は幾つかあるが，オブジェクト指向アプローチがデータモデル論に立脚した最も本格的なものと言えよう[9]．

3.3 その他の先進的データベース応用とオブジェクト指向

非ビジネスデータ処理分野で対象となるデータベース応用は，上記の CAD やマルチメディアデータベース構築とその管理・運用だけではない．他の例としては，CASE (Computer Aided Software Engineering)，ネットワーク（電話網，通信網，鉄道網，道路網，交通網，回路網など）システムの管理・運用やシミュレーション，組織体の意思決定システムのためのデータベース構築，CAD や CASE における共同作業支援 (CSCW : Computer Supported Cooperative Work)，あるいは演繹データベースの実現などをあげることができる．

CASE[10] では，上流工程や下流工程のモデル化をオブジェクト指向で行おうとするアプローチがある．一方ソフトウェア工学では，上流工程から下流工程への変換をフォーマルに記述することが大事であるが，それが未解決であるがゆえに CASE のオブジェクト指向データベース化が十分に行えないという指摘もある[11]．

ネットワークシステムのモデル化にあたっては，リレーショナルデータベースを使ってのモデル化に対して，皆一様に直感的に疑問を感じたといっても過言でない．これは，すでに図-1 で遭遇したのとまったく同様の問題点をそこにみてしまうからにほかならない．つまり，ネットワークをリレーショナルデータベースで表現すれば，必然的にリレーションの結合演算処理を前提としなければならず，リレーショナル DBMS の泣き所を避けては通れないということである．このような場合も，CAD のところで示したと同じ理由でオブジェクト指向データベースアプローチが問題を解決してくれる．

意思決定のための MIS (Management Information System) では複雑で多様なデータを組織化，管理・運用することを要求される一方，さまざまな条件下で的確な決定を下していくことが要求されよう．そこではデータと手続きが一体化していることが望ましく，オブジェクト指向データベースを中核としたシステム構成が適していると考えられる．

CAD や CASE における CSCW では，個々のオブジェクトがそれぞれに行動のパターンをもち，それらのオブジェクトが集まり，全体としてある目的を持って振る舞うという系を的確にモデル化することが必要である．そのようなシステム構築の核となるのがオブジェクト指向 DBMS ではないかとも期待されている[12]．

次世代データベースシステムは，演繹能力をもつのが望ましいことは言を待つまい．ただ，演繹オブジェクト指向データベースシステムを構築するためには，新しいオブジェクト指向データモデル論を必要とするという立場[13]や，そうではなくエキスパートシステム (Expert System : ES) をオブジェクト指向データベースシステムのバックエンドに置くという考え方もあろう．現在どちらの考え方が有効なのか結論を出せるような状態ではないが，現存するオブジェクト指向 DBMS と ES の結合例 (Neuron Data 社の Nexpert Object) をみるかぎりにおいては，オブジェクト指向と演繹性との相性はきわめて良いように感じられる．

4. オブジェクト指向データベースシステム

4.1 オブジェクト指向データベースシステムとは

世界で最初のオブジェクト指向 DBMS は，オレゴン大学院センターで研究・開発された GemStone[8] である．それは，1984 年に "Making Smalltalk a Database System" という題の論文として発表された．論文の題が如実に示すように，GemStone はオブジェクト指向プログラミング言語 Smalltalk-80 のオブジェクト概念をデータベース化することを目的としたシステムであった．換言すれば，ユーザが Smalltalk-80 プログラム中で定義・生成するオブジェクトをプログラムの実行終了後も二次記憶中に格納しておき，必要とあらばいつでも取り出して使える仕掛を作り上げたということである．さらに別の表現をすれば，オブジェクトを永続化 (persistent) でき，それを管理・運用できるシステムを作り上げたということである．

ここで，もう少し具体的にオブジェクト指向 DBMS のイメージを伝えるために，最近のオブジェクト指向 DBMS 製品の 1 つである ONTOS[14] を取り上げ，オブジェクトの生成や利用の様子をみてみよう．まずオブジェクト指向 DBMS では一般的なことであるが，オブジェクトを生成する

にあたりそのオブジェクトが属する永続的なクラスを定義する．ここでは，前章で議論した CAD の論理回路をオブジェクト指向データベースのもとでデータベース化する．**図-2** に図-1 のモジュールのオブジェクト指向表現を示す．また，下にクラス Module の定義を示す．

```
class Module : public Object {
private :
  int priv_moduleNum ;
  char* priv_moduleName ;
  Set* priv_elementParts ;
  Set* priv_connectionParts ;
public :
  Module(int moduleNum,char* moduleName) ;
  Module(APL*) ;

  /accessors*/
  Set* elementParts() {return priv_elementParts ;} ;
  void elementExchange(int elementNum 1, int
    elementNum 2) ;
  ...
} ;
```

なお，ONTOS では，全てのクラスの最上位に位置するクラス Object がオブジェクトの永続性を用意するので，クラス Object のサブクラスとして定義されるクラスのオブジェクトは皆永続的になる．また，最近のオブジェクト指向 DBMS のほとんどがそうであるように，ONTOS もクラスの定義はオブジェクト指向プログラミング言語 C++[15] の構文に従っている．

C++ プログラム中で，モジュール番号が 7 で名前がモジュール 7 のモジュールを生成し，変数 currentModule に割り当てるには，プログラム中で次の文 (statement) を書けばよい．

```
Module *currentModule=new Module(7,
  "Module7") ;
```

なお，オブジェクト指向データベースでは，オブジェクトが生成されるごとに DBMS の責任でそのオブジェクトにオブジェクト識別子 (object identifier) を与える．これにより，DBMS がオブジェクトの同一性 (object identity)[16] を決定できる．この結果，オブジェクトはその値がどのように変化しようとも，その識別子により唯一に識別される．これは，リレーショナルデータベースにおいてリレーションのタップルの唯一性を主キー (primary key) の値で保証するのとは根本から異なる．その分，オブジェクト指向データベースでは実世界のモデル化が自然になる．たとえば，形も重さも同一の釘を多数生成したとしよう．リレーショナルデータベースではその釘一本一本に異なる釘番号を付与しないと格納できないが，オブジェクト指向データベースではその必要はない．

次に，モジュール 7 の素子 (番号) 101 と 103 を交換するためには次の文を書けばよい．

```
currentModule→elementExchange(101, 103) ;
```

次に，ここでもう 1 つのオブジェクト指向データベースの特長に言及する．それは，リレーショナルデータベースで問題となったプログラミング言語とデータベース言語のインピーダンス不整合 (impedance mismatch)[6] の現象が，オブジェクト指向データベースでは相当に改善されていることである．つまり，たとえばリレーショナルデータベースで埋め込み型 SQL 親プログラミング言語の一つである PLI/SQL で応用プログラムを書くと，データベースアクセス部では "EXEC SQL" なる SQL 先頭子 (prefix) で始まり "; "(SQL 終了子) で終わるデータベースアクセス部分をプリコンパイラで処理すべく明確に区別しなければならない[17]．しかし，オブジェクト指向アプローチでは，そもそも Smalltalk-80 や C++ といったオブジェクト指向プログラミング言語のオブジェクトを，その定義構文もそのままにデータベース化しただけである．したがって，オブジェクトの永続化の仕方を工夫すれば，応用

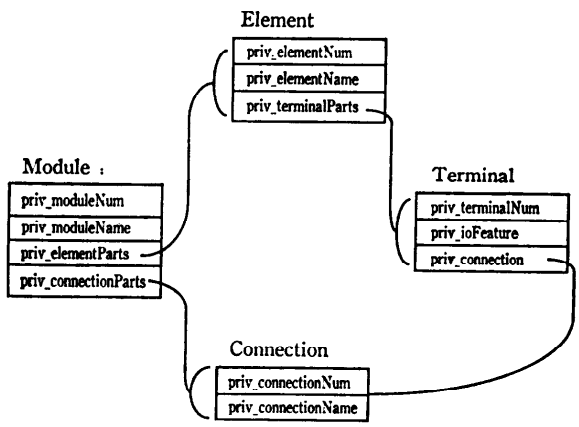


図-2 図-1 のモジュールのオブジェクト指向による表現

プログラマは，プログラムのオブジェクト（つまり非永続的なオブジェクト）を扱っているのか，それともデータベースのオブジェクト（つまり永続的なオブジェクト）を扱っているのかを意識することなくプログラミングできる．

4.2 オブジェクト指向 DBMS の研究・開発の状況

GemStone を始まりとしたオブジェクト指向 DBMS の研究・開発は，1980 年代の中ごろから他の機関でも始まり，1980 年末から 1990 年にかけて十指に余るシステムをみることができるようになった．アメリカではボストン地区，シリコンバレー，テキサスを中心としてシステムが開発されてきた．ヨーロッパでは，フランスで研究・開発活動が活発である．日本でも，現在まで 4 つほどのシステムが開発あるいは研究・開発されている．ここで，以下にそのようなシステムを列挙する（順不同）．米国で開発されたシステム：GemStone (Servio Corporation), Ontos(Ontologic Inc.), Object Store (Object Design Inc.), VERSANT (Versant Object Technology), Objectivity/DB (Objectivity, Inc.), ITASCA (ITASCA Systems Inc.), Statice (Symbolics), TIGRE (Tigre Object Systems, Inc.) (以上商用)；ORION(MCC), IRIS(Hewlett-Packard Laboratories), ODE(Bell Laboratories), Open OODB (Texas Instruments Incorporated), ヨーロッパで開発されたシステム：O_2 (GIP Altair, 仏)，日本では：Jasmine[18]（富士通研究所），OMEGA[19]（図書館情報大学），Odin[20]（日本電気），Obase（千里国際情報事業財団）がある．関連して田中[21]や吉川ら[22]による調査報告がある．

1 つ注意をしておけば，GemStone や ORION[23] といった初期のシステムはオブジェクト指向プログラミング言語 Smalltalk-80 のオブジェクトを永続化するという指針で開発されたが，近年商品化された ONTOS, ObjectStore, Objectivity/DB, VERSANT といったシステムはいずれもがオブジェクト指向プログラミング言語C++のオブジェクトを永続化している．これは Smalltalk-80 はオブジェクト概念での完結性，プロトタイピングの迅速性といった点で勝るのに対し，C++は言語Cのスーパセット (super set) でこれまでのソフトウェアの蓄積を生かせることを最大の理由としている．しかし，プログラミング言語の教育期間を考えると Smalltalk-80 に利があり，非プログラマ (non-programer) にとっては Smalltalk-80 に基づいたシステムのほうが使いやすいのではないかという巻き返し論もある．また，Smalltalk-80 とC++のオブジェクト概念はあまりにも違いすぎるため（極論すればC++のそれは，クラスとかの概念は入れてはいるが，単なるデータ型にすぎない．）オブジェクト指向 DBMS は多言語をサポートすべきであるとの考え方がある[19),24)]．

4.3 オブジェクト指向 DBMS のパフォーマンス

オブジェクト指向 DBMS はパフォーマンスが良いと言われるが，二，三の観点からそれをみてみたい．

まずユーザのソフトウェアの生産性の観点からみてみると，ObjectStore を発売している Object Design Inc. から，UNIX ファイルのデータをアクセスする 134 行の C プログラムが，データを ObjectStore の元でオブジェクト指向データベース化し，CプログラムをC++プログラムに変換することにより 22 行に縮小されたという具体例が提示されている．この結果，プログラムの可読性，保守性，再利用性などが格段に向上することが期待される．

次に，データの格納構造がリレーショナルのそれと根本的に異なり，その結果オブジェクト指向 DBMS ではデータベースの検索や，複合データの挿入，削除，更新で速度の向上がはかれる．具体的には O_2[24] ではウイスコンシン大学で開発されたディスクマネジャ WiSS (the Wisconsin Storage System) を使用した例が報告されている．また，オブジェクト指向 DBMS ベンダ (vender) は，各社独自の効率の良い格納構造を社運を賭けて開発しているのが現状である．それらに共通した考え方は，主記憶内のデータ構造（つまりプログラムでのデータ構造）と二次記憶内のデータ構造を同じにすることにより複合データのアクセスを徹底的に速くしようとするところにある．（このようなアーキテクチャで構築されたオブジェクト指向 DBMS を第二世代のオブジェクト指向 DBMS といったりする．）

オブジェクト指向 DBMS アーキテクチャのもう 1 つの特徴はオブジェクトバッファの導入である．オブジェクト指向データベースでは先述のと

おり，CAD に代表される複合データを処理の対象とするので，関連するデータは一括して主記憶中にあってほしい．そのためオブジェクト指向 DBMS では，オブジェクトバッファをもつのが常套手段となっている．システムによってはこのバッファサイズより大きなオブジェクトの生成は許さないほどである（たとえば VERSANT Object Technology 社の VERSANT）．この状況は，当面の作業に必要な複合データをサーバマシーンからチェックアウト (check out) し，クライアント側に置き，作業を終了した後サーバに返す (check in) という作業形態を考えるとなんら不自然ではない．逆にいえば CAD のようなエンジニアリングデータベース環境では，主記憶中での複合データのアクセス速度が重要な要因となると考えてよい．

さて，ここで最近報告されたオブジェクト指向 DBMS のパフォーマンスのベンチマークテストの結果を紹介する．このテストはアメリカ Sun Micro systems 社のカテル (R. G. G. Cattell) らにより報告されたもので[25]，データを複合オブジェクトとし，評価に使ったデータベースシステムは5つのオブジェクト指向 DBMS (GemStone, ONTOS, ObjectStore, VERSANT, Objectivity/DB)，2つのリレーショナル DBMS (Sybase, INGRES)，そして Sun Microsystems 社の ISAM (Indexed Sequential Access Method) ファイルパッケージ NetISAM であった．ベンチマークのために構築したデータベースは2万個の部品(レコード)からなる．1部品から3本の結線が出てほかの部品とランダムにつながる(したがって計6万本の結線がある)．さらに結線の状態は，できるだけ具体的な基盤の配線状況を反映するよう，2万個の部品の 90 パーセント（したがって1万8千個）は自分に割り当てられた1から2万までの部品番号を i としたとき，$i-100$ から $i+99$ の部品(つまり自分の番号の前後合わせて1パーセント内）にランダムに結線されるような局所性を想定している．部品と結線データの総計は約4メガバイトになるが，それらは最初はサーバにあり，必要に応じてクライアントに転送されてくる．ベンチマークは，Sun 3/260（8メガバイトの主記憶で約5メガバイトのキャッシュ）のもと，①部品番号をランダムに与えてその部品のデータを検索する (lookup)，②オブジェクト間を結線に従ってトラバースする（渡り歩く，traversal)，③オブジェクトを挿入する (insert) の3種類である．このテストをおのおの10回繰り返し，第一回目のテストをコールド (cold) な状態，第二回目以降，結果が定常的になった状態をウオーム (warm) と言っている．エンジニアリングデータベースでは，ウオームな状態での複合部品のトラバースが大事である．**図-3** にカテルらのベンチマークの結果を示す．オブジェクト指向 DBMS はリレーショナル DBMS に比べて，平均して数十倍，対象によっては数百倍程，高速である．

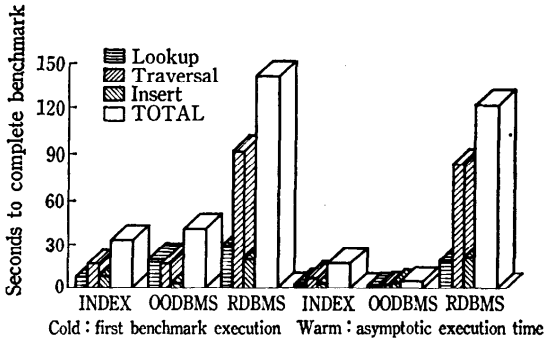


図-3 カテルら[25]のエンジニアリングデータベースベンチマークテストの一結果

5. オブジェクト指向データベースに適応するために

オブジェクト指向データベースは，従来型のデータベースとはそのパラダイムが異なり，リレーショナルデータベースを第二世代のデータベースとするなら第三世代のデータベースと呼んでよいであろう．しかし，パラダイムが変わると，実世界のデータ構造の認識の仕方そのものに大きな変化がおこることに注意しないといけない．その根本は，オブジェクト概念におけるデータと手続きのカプセル化であろう．コンピュータデータ処理の初期はアルゴリズム中心（すなわちプログラム中心）であったが，データベースの発展とともにデータ中心となり，今その2つの考え方がオブジェクト指向のもとで融合したと考えることもできる．

このような中，一番認識を変えないといけない部分はデータベースの構築，つまりデータベース化の対象となった実世界のデータ構造の認識の仕方であろう．従来はまず実世界を実体-関連モデ

ルで表現し，それをリレーショナルデータベースに変換し，正規化などのチューニングをする．応用プログラムは別途開発する，といったシナリオであった[26]．しかし，オブジェクト指向では単にオブジェクト指向データベース概念のみならず，オブジェクト指向プログラミングを含む広範なオブジェクト指向概念の理解が要求される．換言すれば，必要なクラスの定義が完了した時点で同時に応用プログラムもでき上がっていると言えよう[27),28]．

関連して，オブジェクト指向 DBMS のクラスライブラリ (class library) の整備が大事である．オブジェクト指向では，継承 (inheritance) という概念とメカニズムを最大限利用することにより，コードの再利用性，最小化などを達成できると考えられている．それを可能にする鍵は，オブジェクト指向 DBMS 自体が，あらかじめどれほど豊かなクラス階層をクラスライブラリとして用意しておき，ユーザの利用に供しうるか，ということである．一部のオブジェクト指向 DBMS のライブラリは貧弱であると聞く．ユーザの期待を裏切らないよう対処すべきである．

ライブラリの整備に加えて，オブジェクト指向データベースの標準化も大きな課題である．Smalltalk-80 あるいは C++ といった大きな流れは不変であろうとも，オブジェクトの永続化の方法，質問言語 (query language)，プログラミング言語インタフェース，エンドユーザインタフェースなどさまざまな点でシステムごとにまちまちなのが現状である．ANSI(アメリカ国家規格協会)，OMG (Object Management Group)，CFI (CAD Framework Initiative)，OSF (Open Software Foundation) などで標準化の作業が行われているが，現在のところ標準がいつ，どのような形ででき上がるか明確なシナリオはでき上がっていない[22]．

このような観点からも，これからできるだけ早い時期にさまざまなオブジェクト指向データベースの応用を開発してみて経験を積み，利点や問題点を的確に認知することが今われわれに課せられている大事な課題の１つであろう．

6. おわりに

オブジェクト指向データベースシステムは次世代データベースシステムとして CAD，CASE，マルチメディアといった先進的データベース応用のための中核をなす DBMS として機能してゆくであろう．現在，オブジェクト指向プログラミング言語 Object-oriented COBOL も出現したという．それをプログラミング言語インタフェースとしてもつオブジェクト指向データベースシステムが出現すれば，オブジェクト指向データベースシステムはビジネスデータ処理分野のデータベースシステムとしての役割も果たすことになろう．ネットワークデータベースは 1960 年代初頭に誕生し今なお広く使われているが，リレーショナルデータベースにその役割を譲り今終焉しようとしている．そのリレーショナルデータベースは 1980 年代に入り実用化された．ビジネスデータ処理でのリレーショナルアプローチがこれからもますます重要性を帯びてくることに疑いはない．しかし，このような中，1990 年に入りオブジェクト指向 DBMS がベータテスト (beta test) を終え続々商品化されている．その出現はわれわれのこれまでのデータベース観を根底から変える新しいデータベースパラダイムを提供している．上でみたようにパフォーマンスにも目をみはるものがある．21 世紀に向けた次世代データベースシステムをオブジェクト指向データベースシステム抜きに語ることは決してできない．

謝辞　これまでオブジェクト指向データベースシステムについて議論してくださったさまざまな方々に深甚の謝意を表する．

参考文献

1) [Codd 70] Codd, E. F.: A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Vol. 13, No. 6, pp. 377-387 (1970).
2) [HaLo 82] Haskin, R. and Lorie, R.: On Extending the Functions of a Relational Database System, Proceedings of the ACM SIGMOD Conference on Management of Data, pp. 207-212 (1982).
3) [StRH 90] Stonebraker, M., Rowe, L. A. and Hirohama, M.: The Implementation of POSTGRES, IEEE Transactions on Knowledge and Data Engineering, Vol. 2, No. 1, pp. 125-142 (Mar. 1990).
4) [Comm 90] The Committee for Advanced DBMS Function: Third-Generation Data Base System Manifesto, Memo. No. UCB/ERL M 90/28, Electronics Research Laboratory, UC Berkeley (9 Apr. 1990).

5) [GoRo 89] Goldberg, A. and Robson, D.: Smalltalk-80: The Languages, Addison-Wesley, Massachusetts (1989).
6) [CoMa 84] Copeland, G. and Maier, D.: Making Smalltalk a Database System, Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 316-325 (1984).
7) [StRG 83] Stonebraker, M., Rubenstein, B. and Guttman, A.: Application of Abstract Data Types and Abstract Indices to CAD Data Bases, Proceedings of Engineering Design Applications, Database Week, San Jose -May 23-26, 1983, pp. 107-113 (1983).
8) [Masu 87] 増永良文：マルチメディアデータベース総論，情報処理，Vol. 28, No. 6, pp. 671-684 (1987).
9) [Masu 90] 増永良文：オブジェクト指向マルチメディアデータベース，システム／制御／情報学会会誌，Vol. 34, No. 8, pp. 469-478 (1990).
10) [IPSJ 90] 情報処理，CASE 環境特集号，Vol. 31, No. 8 (1990).
11) [Matu 90] 松本吉弘：CASE 環境のためのデータベース，情報処理学会データベース・システム研究会，1990 年アドバンスト・データベース・シンポジウム (1990 年 12 月).
12) [IPSJ 90 a] 情報処理学会，第 80 回データベース・システム研究会・第 73 回人工知能研究会・合同研究会，パネル討論会 "オブジェクト指向データベースと分散人工知能" (1990 年 11 月).
13) [YoNi 90] 横田一正，西尾章治郎：演繹・オブジェクト指向データベース，情報処理，Vol. 31, No. 2, pp. 234-243 (1990).
14) [Onto 89] Ontologic Inc.: Ontos Object Database Developer's Guide, 132 p. (Oct. 1990).
15) [Stro 86] Stroustrup, B.: The C++ Programming Language, 328 p., Addison-Wesley Pub. (1986).
16) [KhCo 86] Khoshafian, S. and Copeland, G.: Object Identity, Proceedings of the First OOPSLA Conference, pp. 406-416 (1986).
17) [JSA 90] 日本規格協会：データベース言語 SQL, JIS X 3005 (1990).
18) [AIM* 90] Aoshima, M., Izumida, Y., Makinouchi, A., Suzuki, F. and Yamane, Y.: The C-based Database Programming Language Jasmine/C, Proceedings of the 16th International Conference on Very Large Databases, pp. 539-551 (1990).
19) [Masu 91] Masunaga, Y.: Design Issues of OMEGA: An Object-Oriented Multimedia Database Management System, Journal of Information Processing, Vol. 14, No. 1, pp. 60-74 (Mar. 1991).
20) [KiTs 90] 木村 裕，鶴岡邦敏：オブジェクト指向データベース言語 Odin/C++ について，電子情報通信学会データ工学研究会報告，DE 90-26 (1990 年12 月).
21) [Tana 90] 田中 淳：データベース活用新時代へ－オブジェクト指向データベースの進撃始まる，日経 AI 別冊 1990 夏号，pp. 56-73 (1990).
22) [YoST 90] 吉川正俊，下條真司，田中克己：オブジェクト指向データベースシステムの最近の研究・開発動向，電子情報通信学会データ工学研究会報告，DE 90-30 (1990 年 12 月).
23) [KBC *89] Kim, W., Ballou, N., Chou, H.-T., Garza, J. F. and Woelk, D.: Features of the ORION Object-Oriented Database System, in (28), pp. 251-282 (1989).
24) [Deux 90] Deux, O.: The Story of O_2, IEEE Transactions on Knowledge and Data Engineering, Vol. 2, No. 1, pp. 91-108 (1990) (日経 AI 別冊，1990 秋号，pp. 138-165 に邦訳あり).
25) [CaSk 90] Cattell, R. G. G. and Skeen, J.: Engineering Database Benchmark, Technical Report, 27 p. Sun Microsystems (Apr. 1990).
26) [Masu 90 a] 増永良文：リレーショナルデータベースの基礎，212 p., オーム社 (1990).
27) [Saka 90] 酒井博敬：オブジェクト指向入門，オーム社 (1990).
28) [Booc 91] Booch, G.: Object-Oriented Design with Applications, 580 p., Benjamin/Cummings Pub. (1991).
29) [KiLo 89] Kim, W. and Lochovsky, F. H. (eds.): Object-Oriented Concepts, Databases, and Applications (book), 602 p., ACM Press (1989).
30) [ZdMa 90] Zdonik, S. B. and Maier, D. (eds.): Readings in Object-Oriented Database Systems (book), 629 p., Morgan Kaufman Pub. (1990).
31) [Kim 90] Kim, W.: Introduction to Object-Oriented Databases (book), 234 p., MIT Press (1990).

(平成 3 年 1 月 22 日受付)

増永 良文 (正会員)

昭和 16 年生．昭和 45 年東北大学大学院工学研究科博士課程電気及び通信工学専攻修了．工学博士．同年東北大学電気通信研究所助手．昭和 58 年図書館情報大学図書館情報学部助教授．昭和 61 年より同大学教授．その間，昭和 50～52 年国際応用システム解析研究所（略称 IIASA，オーストリア）研究員．昭和 57～58 年 IBM サンホゼ研究所（米国カリフォルニア州）客員研究員．分散型データベース，マルチメディアデータベース，オブジェクト指向データベース等の研究開発に従事．著書「リレーショナルデータベースの基礎―データモデル編―」(オーム社) など．平成 3 年度より情報処理学会データベース・システム研究会主査．電子情報通信学会，ACM 各会員．