解　説

4. オブジェクト指向データベースの応用

4.1 オブジェクト指向データベースと CASE†

川越恭二††　岸　知二†††　鶴岡邦敏††

1. はじめに

ソフトウェアの開発には生産性と品質を同時に向上することが求められている．これをコンピュータを用いて支援するために CASE (Computer Aided Software Engineering) の研究開発および実用化が積極的に進められている[20)~25)]．

CASE の流れは，下流から上流へ，単独から統合へと流れていると言われている[20)]．すなわち，当初，構造化プログラミングを支援するためのエディタやコンパイラに代表される詳細設計や製造作業を支援する開発環境から，データフロー図や状態遷移図の作成を支援したりシミュレーションしたりする設計上流工程を支援する環境へと発展している．さらに，上流から下流までのさまざまなツールを，独立に各ツールに適した場で利用していた状況から，ソフトウェア開発ライフサイクルというマクロな視点から使用するツールの目的や役割を明確にしツールを統合化して使用する段階に達している．

このような CASE の統合化の流れからくる重要な課題が CASE データベースの構築である．CASE で必要な情報管理，CASE で作られる情報の伝達，管理などを目的とする CASE データベースの構築によって CASE の情報統合化が達成される．

CASE とデータベースとの関係は，データベースのスキーマ情報を保管・管理する DD/D（データディレクトリ・辞書）にはじまると考えられる．このスキーマ情報は，データベース応用システム開発段階で作られる CASE データベースで扱うべきデータの一つで，その登録，検索ツールは CASE でもある．最近では，スキーマ情報だけでなくデータベース応用システムで扱う情報の定義情報を一元的に管理する情報資源辞書の標準化が検討されている[29)]．

CASE データベースは情報資源辞書で管理されるような情報定義情報だけでなく，システム開発プロジェクトの管理やシステムのテストに必要な情報，などのシステム開発ライフサイクルすべてに関係した情報を蓄積・管理する．システムが大規模化，複雑化するに従い，システム開発で扱うべき情報は膨大になり，CASE データベースの必要性・重要性はますます高まっている．最近，IBM の AD/Cycle, DEC などでのリポジトリ概念が提唱されているのもその重要性からきている．

一方，オブジェクト指向データベースの最近の研究開発・製品化の動きは加速化している．これまでにすでに多くのオブジェクト指向 DBMS が製品化されている．オブジェクト指向データベースの用途は，現在は，LSI 設計や機械設計などの CAD の分野やマルチメディア文書処理の分野が中心であるが，上記 CASE も重要な応用分野として考えられている．

本解説では，このような CASE データベースの現在の状況についてオブジェクト指向データベースの適用という観点から説明する．この分野の研究開発は現在進行形で進められているため，有効な CASE データベースの構築の成功例はほとんど聞かれない．そこで，CASE データベースへの期待とオブジェクト指向データベースへの期待の説明と有名な研究開発事例を紹介する．

2. では，CASE データベースの内容とオブジェクト指向との関係に触れる．3. では，研究開発事例として日米の代表的プロジェクトである Arcadia と KyotoDB を紹介する．4. では，

† Object-Oriented Database and Computer Aided Software engineering by Kyoji KAWAGOE, Tomoji KISHI and Kunitoshi TSURUOKA (NEC Corporation).

†† 日本電気(株) C&C システム研究所
††† 日本電気(株)ソフトウェア生産技術開発本部

DBMS 側からみたオブジェクト指向 DBMS への課題を説明する.

なお, 本解説は CASE の視点でオブジェクト指向 DB を整理したものであるが, CASE, CAD, DTP (文書作成) などの分野におけるオブジェクト指向 DB (C-OOD[31]) は共通部分も多数存在する. 本特集の CAD, EOA との重複はお許しねがいたい.

2. CASE データベースとオブジェクト指向

2.1 CASE データベース

以下 CASE データベースの役割と要件について, 管理対象, 運用, CASE 環境に分けておのおの CASE の立場から概観する.

2.1.1 管理対象

CASE データベースはさまざまな情報を管理しなければならない. これらの情報の中にはソフトウェア開発プロセスによって生成される最終, 中間成果物, 開発に必要となる各種設計情報, プロジェクトの管理データや品質データなどの管理情報が含まれる. さらに最近では開発に使われる各種ツールやそれを用いた開発の手順を CASE データベースのデータとして管理しようという試みもなされている[14].

これらの CASE データベースで管理使用するデータは一般に構造をもっている. そうした構造の例としては, ソースコードの構文的な構造, ドキュメントの章, 節の構造, ソフトウェアのモジュール構造, プロジェクトの構造等々があげられる. さらにこれらのデータ間には参照関係, 詳細化の関係, 構成の関係, 版の関係など, 多くの関係が定義される.

このように CASE データベースは単に一様なデータを大量に扱うだけでなく, 構造, 形態, 特性の異なる多様で複雑なデータとその間の関連を効率的に管理できることが重要である.

2.1.2 運 用

運用面においても, CASE データベースは従来のデータベースとは, 異なった特性をもつと言われている[2]. たとえば CASE データベースにおけるトランザクションは通常のデータ処理のトランザクションに比較して長く, それが数日に及ぶこともまれではない. こうした特性を踏まえ, 通常のデータベースでは行われないような排他制御の考え方も提案されている[1]. あるいはシステムクラッシュへの対処も従来とは異なったものになると指摘されている[2].

またソフトウェア開発は分散環境でなされることが多い. したがって CASE データベースも分散環境での並行的な使用に対応できなければならない. 分散データベースの問題は必ずしもソフトウェア開発だけの問題ではないが, 先に述べたトランザクションの特性などとも関連して, ソフトウェア開発プロセスに適した分散環境への対応が望まれる.

さらに最近はソフトウェア開発プロセスにおける協調作業を支援しようとする試みもなされている[18]. CASE データベースにもそうした視点からの機能が期待されている[19].

CASE データベース中にはさまざまなデータが蓄積されていくが, それらを削除していくことも考慮しなければならない[5]. たとえば版管理を行う過程では二度と参照されない版も多く作られる. あるいは構成情報などが複雑に変化する状況において, どこからも参照されないようなモジュール情報もできる. こうした不要な情報を検出し, 削除していくための方針が明らかにされることが望まれる.

2.1.3 CASE 環境

CASE 環境は, 単独のツールではなく複数のツールによって構成される. それも単に複数のツールが集まっただけのものではなく, 有機的に結合してソフトウェア開発を効果的に支援することが期待されている. これを CASE 環境の統合 (integration) と呼ぶ[3]. 統合にはデータの統合, コントロールの統合, プレゼンテーション (look & feel) の統合の三つがあると言われており[3], CASE データベースは特に最初の二つと深い関連をもっている.

一般にデータの統合とはデータの一貫性保持, 各種ビューの確保, トレーサビリティの保証などを指し, コントロールの統合はツール間での連動やツールのトリガなどを指す. こうした統合の実現には, 1)CASE データベースの扱うデータ構造や, 2)統合を想定した CASE のアーキテクチャが重要な役割を果たす.

1) CASE データベースのデータ構造をどのように設定するかにより, その CASE 環境の扱え

る問題の範囲が規定される．複数のツールが統合される CASE 環境においては，すべてのツールが共通のデータ構造を通して CASE データベースにアクセスする．一方あるツールは階層的なビューを必要とし，別のツールは関係的なビューを必要とする，というようにそれぞれのツールは独自のビューを必要とすることもある．このようにすべてのツールの要求を満たす十分なデータ構造を提供するのは困難な問題である．

さらに新たなツールが付加されたり，既存のツールが拡張されるなど，CASE 環境は常に変化する．このような追加や変更に対する柔軟性や拡張性を保持しながらどのように統合を実現していくかということも，大きな課題である．

2) CASE 環境のアーキテクチャについてはさまざまな研究が進められている段階にある[12]．図-1 に示したのはオブジェクト管理層を中心にしたアーキテクチャの例である．ここでオブジェクト管理層は，必ずしも従来的な意味でのデータベースそのものに対応するわけではない．むしろオブジェクト管理層はその CASE 環境の必要とする機能を規定するものであり，その実現のために用いられている特定のデータベースなどに依存するものではない．特定のプラットフォーム（ハードウェア，OS，データベースなど）に依存しない仮想的な層であるとも捉えることもできる．したがって CASE データベースの議論をする際には，アーキテクチャの中のどの部分に対応した機能を対象にするのかを明確にする必要がある．

2.2 オブジェクト指向の CASE データベース

先に述べた CASE データベースを構築するためのアプローチをデータベース的に眺め，オブジェクト指向 DBMS を使用したときの効果を説明する．

2.2.1 CASE データベース分類

データベースを構築する際に行われる概念モデルの記述と論理モデルの記述の二つの視点から CASE データベースを分類する．概念モデルとはデータベース対象となった実世界のデータ構造のモデルであり，論理モデルとは DBMS で実動可能なモデルである[35]．概念モデルでは，データベース対象をモデル化する手段として ER データモデルやオブジェクト指向データモデルが使用される．また，論理モデルでは，利用可能な DBMS が存在するオブジェクト指向 DBMS や関係 DBMS が使用されるほか，通常のファイルシステムで専用 DBMS を作ることも可能である．

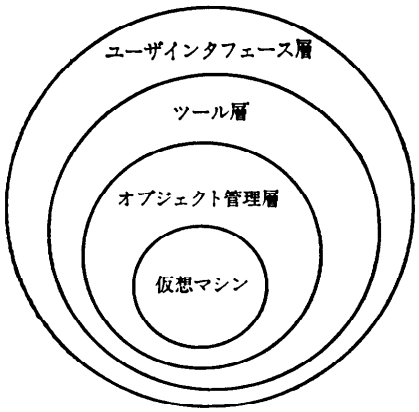

図-1 CASE アーキテクチャの例

この二つの視点から各種研究開発を分類した結果を表-1 に示す．

2.2.2 オブジェクト指向 DB の効果

概念モデルの記述手段として ER モデルを使用するアプローチでは，1. で述べた DD/D や IRDS でも多くの研究が行われてきた．そのポイントはオブジェクト指向との比較よりも関係モデルとの比較で論じられている．一方，オブジェクト指向モデルを使用するアプローチは最近になって行われ，ER モデルに比べてクラス階層によりさまざまなレベルの情報を扱える点と，オブジェクトとしてデータへの操作の定義とデータの定義とが一体化できる点が強調されている．現在，ER モデル自身も拡張されているため，後者の点に関して，オブジェクト指向データモデルの優位性（記述力，記述容易さなど）はまだ具体的には示されていない．しかし，概念的にはオブジェクト指向データモデルは ER モデルよりもモデリング能力

表-1 CASE データベースの研究アプローチ

概念モデル	論理モデル	研究開発例
ER モデル	関係モデル	AD/Cycle
オブジェクト指向モデル	関係モデル	KyotoDB[17], StP[18]
オブジェクト指向モデル	オブジェクト指向モデル	Arcadia/TRW[13]
オブジェクト指向モデル	専用モデル（ファイル）	NSE[9], Arcadia(Cactis)[9]

に富むものであるため，今後は，オブジェクト指向データモデルが採用されていくものと考えられる．

一方，論理モデルについては，数年前の段階では実用的なオブジェクト指向 DBMS がなく，関係 DBMS は容易に利用できたために，実績のある関係 DBMS を使用するアプローチを取らざるをえない研究が多かった[10]．しかし，実用的なオブジェクト指向 DBMS が利用できる現在，論理モデルとしてオブジェクト指向モデルを使用するアプローチが増加するであろう．

CASE 開発者にとってオブジェクト指向 DBMS を使用するメリットとして，筆者の一人は，文献31) で，実際にオブジェクト指向 DBMS を使用した経験に基づいて以下のように指摘している．

- 開発（設計製造）効率

CASE 対象世界の情報モデリングとその実現におけるギャップが少ないため，効率よく開発できる．

- 拡張性

情報構造が自然で理解しやすいため，構造の変更や拡張が容易である．

- 実行性能

CASE のような対話システムでは応答時間が問題となる．処理オーバヘッドを極力減らすために，複雑な構造の変更をともなうような対話処理では，これまでは，通常，専用 DBMS を構築していた．汎用 DBMS は対話処理というよりは作成後の情報管理手段としてのみ使用されてきた．オブジェクト指向 DBMS は対話処理でのデータ管理手段として十分利用できそうである．

3. CASE データベースの研究開発事例

2. で説明した概念モデルでオブジェクト指向を使用している Arcadia と KyotoDB を CASE データベースの日米の代表的な研究開発事例として紹介する．

3.1 Arcadia

Arcadia[15] は Osterweil らによって研究開発がすすめられているソフトウェア開発環境であり，柔軟性と拡張性を備え，かつ統合された CASE 環境を実現することを目的としている．

Arcadia ではプロセスプログラミング[11]の概念を導入して柔軟で拡張容易な CASE 環境を実現しようとしている．このプロセスプログラムはソフトウェアを開発，保守するときのさまざまなプロセスを記述したものである．Arcadia ではこのプロセスプログラムを PPL (Process Programming Language) によって記述する．PPL では CASE 環境中のツールが演算子もしくは関数に，ツールやユーザによって生成されるさまざまなオブジェクトがそのオペランドになる．このプロセスプログラムを修正することにより柔軟性が，新たなプロセスプログラムを記述することにより拡張性が得られると主張されている．

Arcadia のプロトタイプ環境である Arcadia-1 は，プロセスプログラムを実行するインタプリタ，ユーザとのやりとりを行うためのユーザインタフェース管理システム，ソフトウェアオブジェクトにアクセスするためのオブジェクト管理システム，および仮想マシンから構成される固定部分 (Fixed part) とプロセスプログラム，ツール，ソフトウェアオブジェクトなどの可変部分 (Variant part) からなる．

Arcadia-1 では，オブジェクト管理システムが CASE データベースとして重要な役割を果たしている．このオブジェクト管理システムは，概念的に三つのレベルに分類されている．まず，上位レベルではオブジェクトの型，オブジェクト間の関係，オブジェクトの永続性に関する特性の記述などが行われる．そして，中間レベルではオブジェクトの実際のインスタンスや関係が管理される．この中には型の一貫性を保証したり，関係にそって自動的に影響を波及するような機能が含まれる．最後に，下位レベルでは格納領域の管理，並行処理の管理，トランザクションの管理などがなされる．具体的な DBMS としては，上記中間レベルを扱う Cactis[7] が開発されている．

図-2 は Cactis におけるオブジェクト定義の記述例である．ここでは障害票/障害処理票の管理システムにおけるオブジェクトと関係が定義されている．オブジェクト *bug_report* とオブジェクト *fix_report* は関係 *bug_fix* によって関係付けられており，オブジェクト *bug_report* の属性 *is_fixed* は，オブジェクト *fix_report* の属性 *is_fixed* から求められることが示されている．

この Arcadia における CASE データベースは，オブジェクトを自動的にメンテナンスするデータベース機能に特徴がある．しかし，小さな例題で

```
Relationship Class bug_fix
  Transmits
    is_fixed        : boolean To Socket, Default=False;
End Relationship:
Relationship Class module_bug Multi Plug End Relationship;
Relationship Class to_test_result Multi Plug End Relationship;
Relationship Class to_person Multi Plug End Relationship;
Relationship Class work_to_person Multi Socket End Relationship;
Object Class bug_report
  Relationships
    in_module       : module_bug Plug;
    symptom         : to_test_result Plug;
      fixed_by      : big_fix Socket;
    reported_by     : to_person Plug;
    assigned_to     : work_to_person Socket;
  Attributes
    report_words    : text_string;
    is_fixed        : boolean;
  Rules
    is_fixed:=fixed_by.is_fixed;
End Object:
Object Class fix_report
  Relationships
    fixes           : bug_fix Plug;
    fixed_by        : to_person Plug;
    change          : to_delta Plug;
  Attributes
    report_words    : text_string;
  Rules
    fixes.is_fixed:=True;
End Object;
```

図-2 Cactis での定義例

なく，ある程度のデータ量で，あるいは分散した環境でどの程度使えるようにできるかが今後の課題であると考えられる．

3.2 KyotoDB

KyotoDB[17]は松本らによって開発中のソフトウェア開発環境であり，ソフトウェア開発の各段階で生産されるソフトウェア構成要素とその意味的関係を構造的に管理することを目的としている．

KyotoDB はユーザインタフェースを司るユーザインタフェース層，各種ツールを含むツール層，および KyotoDB 核から構成される．KyotoDB 核はソフトウェア開発の諸段階で作られるソフトウェア構成要素をオブジェクトベースとして管理し，それらに対するアクセスのメソッドを提供する．またここでは通常のソフトウェア構成要素だけでなく，上流工程から下流工程への導出関係といった関係もソフトウェア構成要素として管理することができ，影響の波及分析などを可能にしている．

図-3 に KyotoDB 核におけるオブジェクト構造を示す．Data Object はソフトウェア構成要素の基本単位を構造化されたデータとして保持し，それに対するアクセスのメソッドを提供する．Relation Object は Data Object 間の関係をデータとしてもち，それに対するアクセスのメソッドを提供する．Artifact Object は一つの Data Object と，それに関して記述された複数の Relation Object とをもち，これらに対するアクセスのメソッドを提供し，これらの間の一貫性を保持する役割を果たす．View Object は Data Object を可視化したもの（画面イメージなど）であり，画面情報などを扱う．Management Object は Data Object を扱う Unit Workload（プロジェクトの構成員が担当する作業の基本単位）の責任者，納期，コストなどのデータと，それに対する操作を提供する．これらのオブジェクトは関係 DBMS を用いて蓄積管理されている．

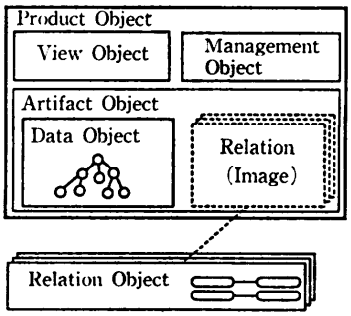

図-3 KyotoDB 核でのオブジェクト構造

現在，KyotoDB は協調作業支援の側面に重点が置かれているように見受けられるが，KyotoDB のオブジェクトの構造のそれ以外の側面での有効性検証が興味深いところである．

これら以外にもオブジェクト指向の CASE データベースが研究開発されている．CASE ツールを結合する手段としてオブジェクト指向 DB を利用したり CASE ツールによって作成された成果物の管理を効率的に行う統合 CASE データベースの開発も盛んであり，今後も多くの研究開発事例が現れるものと思われる．

4. オブジェクト指向 CASE データベース技術の課題

CASE データベースの技術課題については文献2) で多様な情報の表現や版管理，長大トランザクション管理などが指摘されている．今後のオブジェクト指向データベース (DB) が解決すべき課題に関しては，以下のような具体的なデータベース技術課題の解決が必要である．

- 多様な形態をもつ情報の統合管理

2.で述べたように CASE データベースで管理すべき情報は，各種仕様書，図面（構成図，データフロー図，データモデル図など)，データ項目とその属性，プログラム／モジュールの本体（ソース形式／実行形式)，作業進捗データなど多様であり，それらは複雑な相互関連をもつ．

これらを統合的に管理するためには，可変長／長大オブジェクトの格納／操作，クラスの動的生成，クラス間関連の動的設定，などの機能が必要である．特に，CASE ツールによる DB 保守を容易にするためには，項目（インスタンス変数）の追加／削除や項目長の拡大／縮小などのクラス変更操作が，インスタンス格納後でも実行できることが重要である．

これらの機能の一部をすでに実現しているものも多いが，どのようなクラス構造をとるかというクラス構造化技法（オブジェクト指向分析技術）が利用者側にとってはまだ明確ではないため，この面での経験を積む必要がある．

- 図表情報の管理

たとえばデータフロー図エディタでは，ノードとアークからなるデータ構造を対話的に編集する機能が必要である．

オブジェクト指向 DB を用いると，ノードをオブジェクト，アークをインスタンス変数を経由したリンクに対応させて，容易に DB 化が実現できる．またアークに意味を与えたい場合（関係の管理）には，関連メソッドで記述すればよい．この際，オブジェクト指向 DB が提供するデータモデルを，そのまま主記憶上のデータ構造とみなして編集が行える点がポイントである．オブジェクト DB には，(編集/再表示操作にともなう）主記憶上のデータアクセスの高速化，及び主記憶中の仮更新機能（編集完成まで二次記憶更新を遅らせる）が望まれる．特に主記憶に存在するオブジェクトの障害回復技術は未解決の問題である．

- クラスに対する操作の提供

CASE ツールには，データモデル図から DB中のデータ構造を生成するスキーマデザイナなどが含まれる．このため上と類似の理由で，クラス（スキーマ）自体をオブジェクトとして編集できる機能が必要である．

現在のオブジェクト指向 DB で，クラスが単なる型としてみえないものは，この点では不十分である．

今後は，クラスもオブジェクトとして統一的に扱える方式，概念的なオブジェクト形式と，オブジェクトの外部媒体上の表現形態(提示ビュー)との間のなんらかの写像を管理する機能も望まれる．

- 入れ子構造のオブジェクト管理

ソフトウェアモジュールは，それ自身サブモジュール群から構成されるという意味で入れ子構造をもつ．これらの入れ子構造は，オブジェクト指向 DB の複合オブジェクトとして自然に表現できる．ただし，特定の条件による（サブ）モジュールの検索を考慮すると，入れ子構造を対象とする問合せが不可欠である．

現在のオブジェクト指向 DB では，集合オブジェクトを要素とするオブジェクトに対する問合せ表現は，不完全である．非正規関係代数の拡張など，高水準の問合せ（完全な代数系）が重要である．

- 版管理

プログラムの保守に際して，版管理は必須機能である．あるオブジェクト(たとえばモジュール)に対して，特定の版を指定して，その版を構成するすべての子オブジェクト（サブモジュール）を再現できねばならない．また，長大オブジェクト（ソースプログラムなど）の版の保存に際しては，差分管理などにより二次記憶を節約できることが必要である．現在のオブジェクト指向 DB でも，すでに版管理を実装したものがある．ただし，版番号の実現方法，改版の際の処理方法などは，適用分野ごとに多種の方法が考えられる．

現在のオブジェクト指向 DB では，基本的な版管理機能をもつものもあるが，応用分野にマッチした版管理機能を実現するには，版管理のためのデータ構造を利用者がクラスとして定義し，版管

理操作をメソッドとして記述できることが望ましい.

- ビュー管理

ソフトウェア開発，特に近年のプロトタイピング重視の開発過程にあっては，設計仕様の変更が頻繁に発生することが珍しくない．CASE データベースに関しては，これはクラス階層やクラス間関連の変更に対応する．こうした変更時にほかの共同開発者に対する影響を小さくするためには，ある利用者群に対して特定の見方を設定するためのビュー機能が効果的である.

関係データベース (RDB) におけるビューが値のコピーであったのに対し，オブジェクト指向 DB でのビューは元のオブジェクトを直接参照する．このため，ビュー(たとえば結合ビュー)を経由した直接更新や，更新の意味の記述など，RDB では困難であった操作が可能になる[8),32)]．ただし，オブジェクト指向 DBMS におけるビュー機能はまだ研究途中にあり，製品レベルでビュー機能を提供したものはない.

今後，データ構造（オブジェクト間の関連）を自由に組み変えて，仮想的なオブジェクト構造を表現できる機能が実現されることを期待したい.

- 長時間トランザクションの管理

ソフトウェア開発過程で発生する一つの単位業務には，仕様書の作成／改訂，モジュールの作成／修正／テストなどがあり，これらは長時間継続するトランザクションとして実行される．このため，更新中のデータを保持した状態でトランザクションを一時中断する機能，及び中断時点の状態を復元してトランザクションを再開する機能が必要となる[33)]．

すでに長時間トランザクションを提供したとするオブジェクト指向 DB も存在する．また，前に述べた仮更新機能を有する DBMS の場合には，中断／再開機能の実現は比較的容易と思われる.

しかし，入れ子トランザクションの部分的回復／再開，その際の並行制御とリカバリ，版管理との連携（別のトランザクションで同時に開発中のモジュールとのリンクの問題など）など，多くの問題が残されている.

また，共有 DB（リリース用）と私有 DB（テスト用）を用いた分散オブジェクト管理機能の提供も必要となろう.

- オープン性

統合的なリポジトリとして DB を用いるためには，各ツールに対して標準的なアクセスインタフェースに加えて，他システムとの連携のために多様なファイル構造の取り込み機能の提供が必要である[34)]．今後，他 DB／他ファイルとのデータ交換，多言語対応などの機能が充実されるであろう.

以上，CASE からオブジェクト指向 DB 側への要請を中心に述べたが，逆に，オブジェクト指向 DB が活用されるためには CASE 環境側の機能強化も望まれる．ソフトウェア開発工程で得られる情報を個別に DB に入力するのは，利用者の負担が大きすぎる．エディタや言語処理系自身が DB とインタフェースをもち，統合化された CASE データベース利用環境を提供すべきである．オブジェクト指向 DB の活用により，ファイルや二次記憶を意識しない「仮想エディタ」,「仮想コンパイラ」の実現を期待している.

5. おわりに

CASE データベース，特にオブジェクト指向の CASE データベースについて現状と動向を紹介した.

今後，オブジェクト指向データベースが，CASE の一部分として利用されるものと期待できるが，より効果を上げるには，「オブジェクト指向データベースのもつ真のメリットは何か」を十分に議論すべきである.

CASE 開発者にとってはこれまでにない非常に有効な道具であり，オブジェクト指向データベースの利用により開発・保守効率が大幅に向上するのは確実である．この効率アップによって CASE 利用者の望む機能拡張・性能向上などを速やかに CASE に反映できるようになるという間接的な効果はみえている.

しかし，CASE データベースを使用するのは最終的には CASE 利用者である．CASE 利用者にとっての真の効果は CASE 統合化の実現であると考えられるが，オブジェクト指向 DB だからこそ実現できるというものではない．この点で，オブジェクト指向 DB の CASE 利用者へのアピールは弱い．今後，より大きな効果を明確に実証する必要があろう.

参 考 文 献

1) Adams, E. W. et al.: *Object Management in a CASE Environment*, 11th ICSE (1989).
2) Bernstein, P. A.: *Database System Support for Software Engineering—An Extended Abstract—*, 9th ICSE (1987).
3) Chappell, C.: *Real-time CASE: the Integration Battle*, Ovum (1989).
4) Forte, G.: *Rally Round the Repository*, CASE outlook, 1989, No. 4 (1989).
5) Habermann, A. N.: *Automatic Deletion of Obsolete Information*, The Journal of Systems and Software, Vol. 5, No. 2 (May 1985).
6) Honda, M. et al.: *Distributed Object Management Without a Client-Visible Database*, Proc. of Software CAD Database, pp. 57-59 (1989).
7) Hudson, S. E. and King, R.: *The Cactis Project: Database Support for Software Environments*, IEEE Trans. on SE, Vol. 14, No. 6 (June 1988).
8) Kimura, Y. and Tsuruoka, K.: *A View Class Mechanism for Object-Oriented Database Systems*, Proc. DASFAA '91, 1991年4月 (1991).
9) King, R.: *Three Software Engineering Database Research Projects at University of Colorado*, Proc. of Software CAD Database, pp. 70-73 (1989).
10) Muller, R. J.: *Relational DB Support for a SCAD Environment*, Proc. of Software CAD Database, pp. 89-93 (1989).
11) Osterweil, L.: *Software Processes are Software Too*, 9th ICSE (1987).
12) Penedo, M. H. and Riddle, W. E.: *Guest Editors' Introduction Software Engineering Environment Architectures*, IEEE Trans. on SE, Vol. 14, No. 6 (June 1988),
13) Penedo, M. H.: *Abstract for SCAD DB Workshop*, Proc. of Software CAD Database, pp. 109-111 (1989).
14) Penedo, M. H. et al.: *Obéect Management Issues for Software Engineering Environments —Workshop Report—*, SIGPLAN NOTICES, Vol. 24, No. 2 (1989).
15) Taylor, R. N. et al.: *Foundations for the Arcadia Environment Architecture*, SIGSOFT Software Engineering Notes, Vol. 13, No. 5 (Nov. 1988).
16) Wasserman, A. I.: *The Architecture of CASE Environments*, CASE outlook, 1989, No. 2 (1989).
17) 沢田他：ソフトウェアエンジニアリングデータベース *Kyoto DB* 核のプロトタイピング，情報処理学会第 41 回全国大会 (1990).
18) 山下他：CASE 環境における協調活動支援ツールの試作，日本ソフトウェア科学会第7回ソフトウェア研究会 (1990).
19) 石井：グループウェア技術の研究動向，情報処理，Vol. 30, No. 12 (1989).
20) 藤野：*CASE* 環境の概要，情報処理，Vol. 31, No. 8, pp. 1013-1019 (1990).
21) 松本：*CASE* 環境の基礎技術，情報処理，Vol. 31, No. 8, pp. 1020-1027 (1990).
22) 前澤：システムソフトを対象とした *CASE* 環境の現状と動向，情報処理，Vol. 31, No. 8, pp. 1028-1035 (1990).
23) 原田：事務処理ソフトを対象とした *CASE* 環境の現状と動向，情報処理，Vol. 31, No. 8, pp. 1035-1048 (1990).
24) 市川：通信ソフトを対象とした *CASE* 環境の現状と動向，情報処理，Vol. 31, No. 8, pp. 1048-1056 (1990).
25) 山本：制御ソフトを対象とした *CASE* 環境の現状と動向，情報処理，Vol. 31, No. 8, pp. 1057-1067 (1990).
26) 落水：ソフトウェア開発環境の新しい動向，bit, Vol. 21, No. 1 (1989).
27) 岸他：*CASE* 環境構築のためのファイル管理機能，情報処理学会 CASE 環境シンポジウム，1989年3月 (1989).
28) 岸：ソフト開発環境における情報管理について，情報処理データベース・システム研究会，1990 年3月 (1990).
29) 溝口：情報資源辞書システム (*IRDS*)，情報処理，Vol. 29, No. 3, pp. 215-224 (1988).
30) 川越：応用からのオブジェクト指向 *DB*，情報処理学会 DBS 研究会パネル討論資料，1990年7月 (1990).
31) 川越：*CASE* とデータベース，情報処理学会アドバンスト DB シンポジウムパネル討論資料，1990年12月 (1990).
32) 木村，鶴岡：オブジェクト指向データベース操作言語 *Odin/C++* について，電子情報通信学会データ工学研究会，DE 90-26, 1990年12月 (1990).
33) 松本：*CASE* 環境のためのデータベース，アドバンストデータベースシステムシンポジウム，1990年12月，pp. 5-15 (1990).
34) 鶴岡，木村：オブジェクト指向データベース管理システムの物理構造仮想化方式，情報処理学会データベースシステム研究会，73-8, 1989年9月 (1989).
35) 増永：リレーショナルデータベース入門，サイエンス社 (1991).

(平成3年1月30日受付)

川越　恭二（正会員）

1953 年生．1975 年大阪大学工学部電子工学科卒業．1977 年同修士課程修了．同年日本電気(株)入社．以来，中央研究所，C & C システム研究所，ソフトウェア生産技術開発本部で DB，CAD，CASE の研究開発に従事．現在，C &C システム研究所研究課長．1984 年より 1 年間カリフォルニア大学ローレンスバークレー研究所客員研究員．1990 年大阪大学基礎工学部非常勤講師．

岸　　知二（正会員）

1956 年生．1979 年京都大学工学部情報工学科卒業．1981 年同大学院修士課程修了．同年日本電気(株)入社．1986 年より 1 年間カーネギーメロン大学でソフトウェア開発環境の研究に従事．現在ソフトウェア生産技術開発本部主任．ソフトウェア開発環境，構成管理，分析・設計支援に興味を持つ．IEEE，日本認知科学会各会員．

鶴岡　邦敏（正会員）

昭和 28 年生．昭和 51 年東京工業大学理学部情報科学科卒業．同年日本電気(株)入社．中央研究所にてデータベースシステム，オフィスシステム関連の研究開発を経験．現在，同社 C & C システム研究所研究課長．オブジェクト指向 DBMS を中心とした次世代データベースの研究開発に従事．