TECHNOLOGY REVIEW

UNISYS 技報

通巻 33

1992年5月発刊 Vol. 12 No.1

特集：データベース技術

日本ユニシス

ISSN 0914-9996

生産性向上と高度な情報処理の実現に寄与する技術としてデータベース技術は発展してきた．現在，メインフレームを共用する時代からネットワークを通してワークステーションを共用する時代に変わりつつある．新しい情報処理環境の実現に向けてオブジェクト指向技術が期待を集めている．データベース・システムの発展を見る視点には，その理論的展開や実装の開発技術等がありえる．原　潔の**次世代データベースの展望**は，データベースの応用の視点からその技術を見直し，次世代のデータベース・システムの姿を探り，次世代データベースの展望を行っている．

データベース技術に対する標準化はリレーショナル・データベース言語SQLを中心に進められてきた．最近では，実世界の複雑なデータ構造を取り扱うためのオブジェクト指向データモデルの研究が活発に行われ，標準化作業にも，オブジェクト指向の概念が影響を与えている．中山佐保子は，**データベース関連標準化におけるデータベース技術——SQLとオブジェクト指向データベースに求められる機能**の中で，リレーショナル・データベースとオブジェクト指向データベースの二つの標準化の流れに着目し，データベースに寄せられている機能要求に対し，どのように標準化が進められていくのかを考察している．

米国ユニシスでは，1990年10月にUnisys Architecture(UA)を発表し，ユニシスが1990年代に提供するソフトウェア・プロダクトの開発方針と機能を提示した．藤島忠篤は，**ユニシスのデータベース・アーキテクチャ——Unisys Architectureにおけるインフォメーション管理サービス**の中で，UAの概要，UAのサービスの一つであるインフォメーション管理サービス（IMS）の役割を述べるとともに，ユニシスのデータベース分野での方向性を紹介している．

XTPAでは，ディスクとRLPを共用する計算機システムをチャネル結合した多重プロセッサ構成をクロズリ・カプルド・システムと呼ぶ．この多重プロセッサ構成下で汎用データベースの共用を可能とした実現技術がXTC-UDSである．阪口喜好の**XTC-UDSの概要——密結合から疎結合，そして睦結合へ**は，クロズリ・カプルド・システムの特徴を明らかにし，データベース共用を実現するための技術的課題について述べ，XTC-UDSでどのように解決したかを解説している．

UNISYSシリーズ2200・1100の汎用データベース管理システムUDSは，複数の異なるデータモデルを共存の形で管理し，大規模データベース・アプリケーションを効率よく処理・支援する．下田隆夫・小林雅浩の**UNISYSシリーズ2200・1100 UDSロック制御**は，UDSのロック制御の特徴，とくに，大規模システム上の処理効率向上化技術，異なるデータモデルのデータベースのロック管理，UDSの新技術となるXTC-UDS下のRLPを使用したロック制御の解説をしている．

UNISYSシリーズ2200・1100の汎用データベース管理システムUDSの特徴の一つに大規模ページ・バッファがある．大田静枝・行成敦の**UDSコントロールのバッファ管理——バッファ管理の変遷と今後の課題**は，UDSのバッファ管理の特徴と，各使用者環境でその効力を十分に発揮できる調整方法について述べ，さらに，より知的なバッファ管理となるための改良点を考察している．

RDAは，異なるコンピュータシステム間で，データベースの相互運用を実現するためのOSI応用層の規格である．坂谷祥史・島村隆一は，**OSI RDA概略——そのモデルとサービス**の中で，RDA基本標準の内容を，モデルおよびサービスの両面から論じ，併せてINE'91におけるデモンストレーションの概要を紹介している．

IDESは，システム開発の生産性と品質向上を目的とする開発支援システムである．板倉教の**システム開発環境IDESのリポジトリ**は，IDESの基盤となっているリポジトリに焦点をあて，そのデータモデルの考え方，機能，実現およびリポジトリに保存されるIDESデータの構造について説明している．

特集「データベース技術」の発刊によせて

内　藤　　聰

コンピュータ45年余の歴史の中で，データ共用のためにプログラムからデータを分離させたデータベース技術は30年を数える．この間，ハードウェアの技術革新により記憶容量(とくにディスク等の外部記憶装置)の大規模化と処理の高速化が進み，同時に，ユーザニーズは高度化・多様化してきた．これらの変化に対しデータベース技術は研究開発と実践を重ね，CODASYL DBTG提案によるネットワーク型のデータベースシステムやE. F. Codd提案による関係モデル等により実践的にも理論的にも飛躍的な進歩を遂げたが，今なお多くの新しい要求や課題をその応用分野から課せられ，さらなる発展に挑んでいる．たとえば，従来のデータベースシステムの持つデータの共用，データ独立性，データ一貫性，データ呼び出し制御（データ保護）等の機能に加えて，とくに技術分野からの要請として，複雑なデータ構造を表現し処理するためのデータモデルや，知識や手続きの付加機能，あるいは多くの人々がアクセスできる分散環境等が要求されている．これらの要求に対しオブジェクト指向データベース等に見られるように，データモデルの拡張やデータと手続きの一体化等データベースの枠組の拡大を図るとも言える研究開発が積極的に行われている．これらの研究，標準化の中心としてACM, IEEE, VLDB (超大規模データベース) 国際会議，日本では情報処理学会，電気通信学会，さらに最近ではSQLの標準化を中心としてANSI, ISO, 日本規格協会，が活発に活動している．

今コンピュータシステムは社会生活の一部に組み込まれ，なくてはならぬ重要な役割を担うとともに，その利用の広がりの中でさらに高度な対応を求められている．いわく，大量のバックログを抱えるシステム開発の生産性向上であり，システムの効率と信頼性のさらなる向上であり，エンドユーザにやさしいシステムであり，データの所在場所やコンピュータ機種を気にしない自由で容易なデータ蓄積・更新・検索であり，大規模トランザクション処理における無停止運転であり，さらには画像情報や音声情報等のマルチメディアデータ処理等である．このような要求に対し，コンピュータシステムは機能向上が著しいPCやワークステーションと役割分担をする，いわゆる分散システム化やオープンシステム化をめざしている．このとき汎用機は，インフォメーションネットワークの中でインフォメーションハブという情報の核として超高速大容量データベースサーバの役割を果たす．

90年代に求められる新しいシステム像は統合分散であると言われている．それは，物理的な分散と論理的な統合を意味し，システム全体としての性能，信頼性を向上させ，また柔軟なシステム構成を可能にする．コンピュータシステムが抱える課題を解決するための方策である．ユニシスはこの統合分散の世界を先取りしたUAを発表し提供を開始した．統合分散の世界でもデータベース技術は重要な役割を担う．これまでデータベース技術はプログラムの生産性向

上をめざし，プログラムからデータを分離し，階層型，ネットワーク型，そして関係モデル等を生んだ．また最近ではプログラムの開発，運用，保守に関する総ての情報を一元的に格納するいわゆるリポジトリが注目されている．統合分散の世界ではさらに分散型データベースシステムを実現させなければならない．データの共用と集中制御を目的とするデータベースシステムは新たに分散にともなう技術課題，たとえばデータの分担方法とアクセス方法や，分散を利用者から不可視化する方法等を従来のデータ一貫性等とともに解決・提供する必要がある．

このように，コンピュータシステムが求めるものはさらに高度になっており，データベース技術は従来からの課題とともにこれらの要求に着実に応え，関連ハードウェアや関連ソフトウェアの進歩，各分野での標準化の進展と共にさらなる発展が求められている．

本号では，30年の歴史の中で培われたデータベース技術の基本となる技法やデータベースを取り巻く現状，そして今後の展望をも含め汎用的な利用におけるデータベース技術の周辺を探る．データベース技術の面白さを感じて戴ければ幸である．

（システム技術本部 データベースソフトウェア部 部長）

次世代データベースの展望

Prospects for the Next-generation Database

原　　潔

要　約　大規模なデータを統合・管理し共用することにより，生産性向上と高度な情報処理の実現に寄与する技術としてデータベース技術は発展してきた．その過程で，データ独立性，データモデル機能，同時実行制御機能，回復機能，高水準のデータベース・インタフェースと問い合わせ最適化技法等が実現されている．現在，メインフレームを共用する時代からネットワークを通してワークステーションを共用する時代に変りつつある．新しい情報処理環境の実現に向けてオブジェクト指向技術が期待を集めている．ビジネスデータ処理領域におけるデータベース技術の成功をエンジニアリング領域やCASE領域等，非ビジネスデータ処理領域に拡大するためには従来とは異なったデータベース技術が必要になってきている．データの意味の素直な表現，設計（ロング）トランザクション機能，バージョン機能等である．

応用の拡大という視点からデータベース技術を見直し，現技術の成果を継承し新しい応用の世界を開くものとして，オブジェクト指向データベースが次世代データベースとして位置付けられる．

Abstract　Continued efforts to integrate, manage and share large-volume data have so far kept database technology developing as one of the methods which contribute to higher productivity and more advanced information processing. Now available are assured data independency, data modeling facility, controlled concurrency, recovery capability, high-level database interfaces and query optimization and others. The current trend is seen is that the age of shared mainframes is changing to that of shared workstations through networking. Attention has also been drawn to object-oriented database technology with the aim of implementing a new information processing environment.

New database technology different from traditional approaches has been considered necessary to extend successful database technology in business data processing over to the area of non-business data processing which includes a wide variety of technical, engineering and CASE applications. For such applications, required are plain semantic data representations, long-vector design transactions and version facility.

This paper makes a review of database technology from a viewpoint of the intended expansion of applications, and positions the object-orientd database as a next-generation one which paves the way toward a new world of applications while taking in present-day technological achievements.

1. はじめに

高度情報化社会といわれる現在，「メインフレーム」を共用する時代からネットワークを通して「ワークステーション」を共用する時代に急速に変わりつつある．ワークステーション技術を中心とする革新に伴い，コンピュータの使用環境やその処理形態は大きく変化してきている（ダウンサイジング/ライトサイジング）．コンピュータ利用者にとって最も望ましい情報処理環境はどのようなものになるのか，いま望ましい情報処理環境の実現に向けて新しいパラダイムとして，オブジェクト指向技術が期待

を集めている．

そのような中でデータベースの応用分野も，ビジネスデータ処理分野から非ビジネスデータ処理分野へと今までの枠を越えた広がりを見せてきており，従来とは異なったデータベース技術が必要とされている．その一つがオブジェクト指向データベースであり，拡張リレーショナル・データベースである．とくに1989年12月のDOOD '89国際会議にて発表された「オブジェクト指向データベース・システム宣言」[1]や，それに対して1990年4月に発表された反オブジェクト指向データベース宣言といわれる「第三世代データベース・システム宣言」[2]等を契機に次世代データベース・システム像をめぐる論議が活発化している．実世界の情報をより忠実に表現できるものとして，非常に期待されている．

データベース・システムの発展を見る視点には，その理論的展開や実装の開発技術等がありえるが，本稿ではデータベースの応用の視点からその技術を見直し，次世代のデータベース・システムの姿を探り，次世代データベースの展望を行う．

2. データベース

2.1 データ独立性

複数の異なる利用目的に対して「データの補給基地」としてデータを共有するという必要から生じたデータ管理の考え方がデータベースであり，特定の目的のためのデータ管理であるファイルとの大きな相違点になっている．データベース (Database) という用語が一般的に使われ始めたのは1970年に入ってからであるが，一部では1960年前半から使われていた．その語源は明らかではないが，ベース (base) は軍事用語で補給基地を意味しており，データベースとはそこへ要求を出すとどんなデータでもすぐに供給してくれる「データの補給基地」のことである，という語源説が流布している．ここでは狭義に，コンピュータ内に蓄積されたデータ群が組織体の共有資源となっており，その中のデータが統合されているとき，そのデータ群をデータベースということにする．

データの共有や統合というと，大規模データを多数の利用者により同時にアクセスすることを想定するが，個人用データベース (personal database) であっても，同じデータに対して異なる目的での利用があれば状況は同じである．共有という概念には再利用という概念も含まれている．個人用データベースの発展もカード型，スプレッドシート型からリレーショナルのような本格的なデータベースの利用に変遷してきていることは，大型コンピュータにおけるファイルからデータベースの変遷と同じ背景によると見られる．

データベースはそこに蓄積されるデータ資源の共有化が最大のトピックスである．データベースはさまざまな応用プログラムからデータを切り離して，それを共有資源として統合管理するものであり，そのことによりデータの変化，プログラムの変化から互いを独立にすることによってソフトウェアの生産性を高めようというのが目的である．この性質をデータ独立性といい，データベースの基本的な概念となっている．データベースの歴史は，1963年に米国GE社（ジェネラル・エレクトリック社）が発表したIDSに発し，その後1969年にCODASYL (Conference on Data Systems

Languages：データシステム言語協議会）の DBTG（Data Base Task Group）が制定したネットワーク型データベースと呼ばれるデータベースに始まる．また 1968 年に IBM 社が開発した IMS というデータベース管理システムで採用した階層型データベースがある．これら二つのモデルは，安定したデータの編成技法であるポインタやリンクを使用することによって大規模データの統合管理を実現している．CODASYL によるデータベース共通言語体系[3] の制定はその後のデータベース技術の大きな進展の原動力となった．この言語体系はデータを定義するための言語（データ定義言語：DDL）とデータを操作するための言語（データ操作言語：DML）を明確に分けている．その後，データ独立性を達成するためには三層のスキーマ構造をとるのが望ましいと，ANSI/X3/SPARC が 1978 年に報告した[4]．

この三層のスキーマ構造を図 1 に示す．実世界をモデル化したものが概念スキーマである．利用者のために外部スキーマが与えられ，概念スキーマを実装するために内部スキーマが与えられる．この結果，内部スキーマに変更があっても，内部スキーマと概念スキーマ間の写像を変えるだけで概念スキーマの変更をする必要がない（物理的データ独立性）．一方，実世界に変更があり，概念スキーマが変わっても，外部スキーマと概念スキーマ間の写像を変えるだけで外部スキーマを不変にできる（論理的データ独立性）．

CODASYL にはスキーマ/サブスキーマの機能があるが，この機能ではデータ独立性はまだ不十分である．1970 年に Codd により発表されたリレーショナルモデル[5] では，アクセス方法から独立したデータの表現手段としてリレーションの概念を導入し，その上に仮想データとして定義されるビューにより，この三層のスキーマをより良く

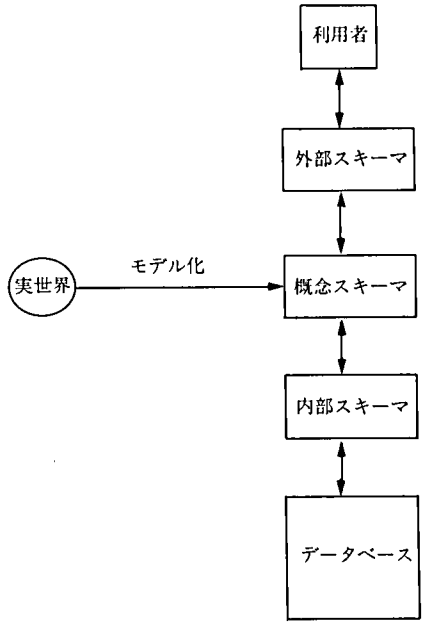


図 1　ANSI/X3/SPARC の三層スキーマ構造

Fig. 1　Three level schema structure of ANSI/X3/SPARC

支援できている．

2.2 データモデル

データの統合・共有という観点から，複数の利用目的からのデータ独立性に加え，共有データの表現手段やその操作法，データの重複の排除やデータ間の無矛盾性の保証等が重要になってくる．

データとは実世界（real world）をなんらかの方法で認識し，抽象化・記号化したものである．それらのデータを蓄積した総体がデータベースである．データベースを構築するために，対象世界をなんらかの方法で認識し，抽象化・記号化し，記述するための手段がデータモデル機能である．このモデル機能は，対象のデータの構文的・意味的構造やデータとデータ間の構文的・意味的構造を記述するための手段を提供する．データモデル機能により記述された結果をデータモデルという．データモデル機能の概念を図2に示す．

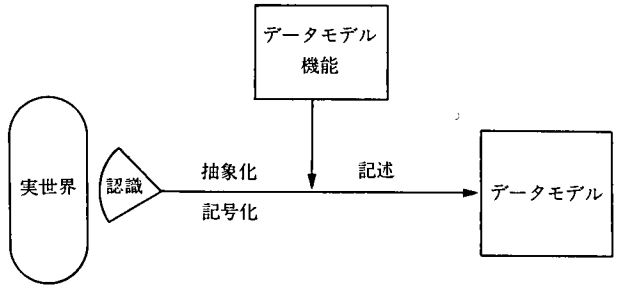


図 2　データモデル機能

Fig. 2　Data modeling facility

データモデルは図3に示すように三つの要素に関する記述からなる[6]．三層スキーマ構造における概念スキーマがこのデータモデルに相当する．

このモデル機能の相違によりデータベースは区別され，ネットワーク型，リレーショナル型等と呼ばれる．主な相違はデータ間の関係の表現の方法にある．

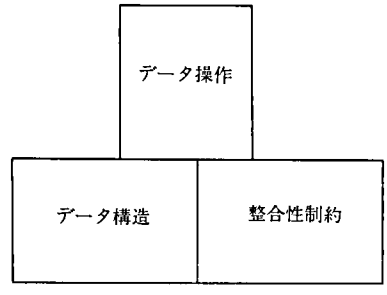


図 3　データモデルの三要素

Fig. 3　Three components of data model

2.3 データの共用

統合され共有資源となっているデータベースを，いろいろな目的で多数の利用者が

自由に利用するためには，利用者とデータベースとの間にあってその橋渡しをするシステムが必要になる．それをデータベース管理システム（DataBase Management System, DBMS）という．

コンピュータはいまや大量のデータを効率よく統合管理し，運用するための情報処理機械となっている．この技術を支えてきたのがデータベース技術である．データベースという概念は，データベース管理システムというソフトウェア製品によって具体的な姿で実現されてきた．以下，このデータベースとデータベース管理システムを合わせて，広義にデータベースということにする．

データベース管理システムの機能を，ここでは，1)ファイル処理の機能，2)データの共有のための機能，3)データの多目的利用のための機能に分類して考える．

1) ファイル処理の機能……情報をディスク上に蓄え，そこから望む情報を取り出すことができる記憶装置としての機能である．データを永続的（Persistet）に管理するとともに，データの基本操作であるデータの問い合わせと更新（追加，削除，変更）を行う．大容量のデータの蓄積管理の技術やその高速アクセス手法を含む．

2) データの共有のための機能……対象とする世界を常に忠実に反映することができ，データベース利用者に正しいデータを提供する機能である．対象世界をモデル化するためのデータモデル機能や，データ処理の同時実行制御やデータベース更新時の障害回復機能を行うトランザクション管理機能を含む．

3) データの多目的利用のための機能……共用されたデータは，その利用者にとって都合の良い方法で利用できなくてはならない．利用者の範囲を拡大するための

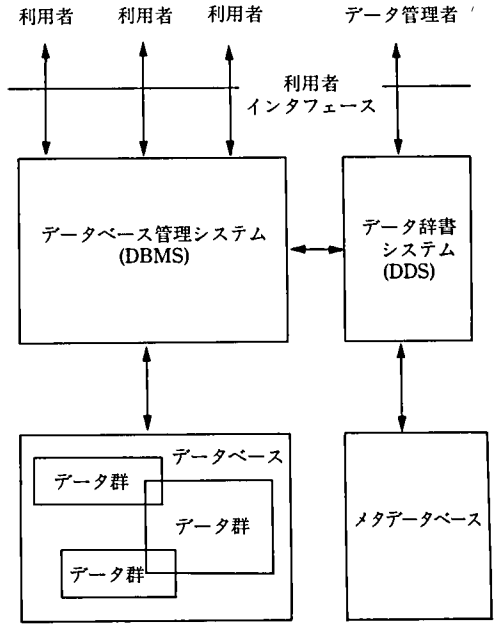


図 4　データベースとデータベース管理システム

Fig. 4　Database and DBMS

条件として，データの物理属性を利用者から隠すための三層スキーマの機能，ソフトウェアの生産性を上げるためのデータ独立性の実現，データベース中のデータをより効果的に利用・管理するためのメタデータ管理の技術がある．図4にデータベースとデータベース管理システムの概念を示す．

2.4 データベース管理システムの利用者

データベースの技術は，大型コンピュータによる大規模データの集中管理ということを目指して発展してきたといえる．ビジネスデータ処理の分野においてその傾向は顕著であり，成功を納めてきている．このような世界においては，データベース利用者も役割分担をしており，それに応じた機能をデータベース管理システムは提供してきた（図5）．

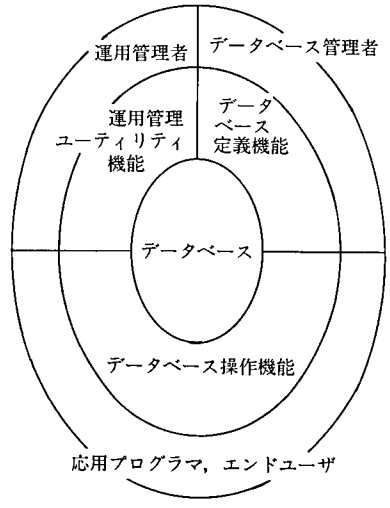


図 5 データベース管理システムの利用者

Fig. 5 Users of DBMS

一つはデータベースの定義に関する仕事で，この役割を担う人達をデータベース管理者と呼び，データベース定義機能を提供している．もう一つは，データを処理したりそのための応用プログラムを準備する仕事で，この役割を担う人達を応用プログラマとかエンドユーザと呼び，データベース操作機能やエンドユーザ機能を提供している．さらに実際にデータベース・システムを運用する運用管理者と呼ばれる人達のために各種のユーティリティを提供している．

3. データベースの応用環境

3.1 応用環境の変化

1960年代後半に出現したデータベースは，その後多くの実務の場で必要性，有効性が実証されてきている．そしていまや情報処理業務において不可欠なものとなっており，システムの分析・設計からシステムの運用，あるいは付加価値情報の提供という利用局面に至るまでのあらゆる範囲に対して影響を与えている．このようなデータベースの性格は情報システムの構築，運用の方法を変えていく大きな原動力ともなって

いる．

データベースは，もともと大型コンピュータによるデータの集中管理という発想の中で発展してきており，とくに銀行の勘定系システム，座席予約システムあるいは伝票処理や在庫管理，人事管理等のビジネスデータ処理分野において成功を納めてきている．この分野での応用は，比較的単純な構造の大量データに対し少量のデータを検索し，(繰返し)単純な計算を行い，その結果更新されたデータを再度データベースに格納するという特徴がある．しかし単位時間当たりのトランザクション数は多く，100件/秒以上にものぼる．

データベース管理システムが現れた当時のコンピュータ環境はメインフレームの共用の時代であり，主メモリは十分には大きくなく，できるだけ小さいメモリ量で効率良くデータベース操作を行うことが重要であった．小さいバッファでも動作可能であり，バッファの内容の頻繁な変更が可能なシステムとしてのデータベース管理システムが追及された．ビジネスデータ処理はこのようなシステムアーキテクチャで十分間に合う応用である．数学的基盤がしっかりしているリレーショナル・データベースは，その簡明さ・自由度の高さ・操作言語の高い表現能力等からとくに事務データ処理分野において大きな成功をおさめてきている．

最近のワークステーション技術を中心とするコンピュータ環境の変化は，ワークステーションをネットワークを通して共有する時代に変化してきている．大型コンピュータによる集中処理からワークステーションによる分散処理への変化によるデータベースの分散化，パーソナル・データベースの利用拡大が進んでいる．そしてウィンドウやビットマップ・ディスプレイに見られる高度な対話機能を持つワークステーションの登場により，従来のようなデータベース利用の形態は変化してきている．利用者の要求としても，情報システムにおけるデータベース利用の高度化があり，コンピュータ寄りのデータモデルではなく，実世界の情報をより忠実に，直接的に表現できるデータモデルが必要とされている．

データベースのビジネス処理分野における成功を，ビジネス処理分野以外へも広げていこうという要請が強くなってきている．設計支援データベース (CAD/CAM) やCASE等では複雑な構造のデータを扱う必要がある．製品のように一つの物が他の物(部品)から構成されるような構造を持つ対象の良いデータ表現とその操作が必要になっている．互いに関連する構造のデータの操作は比較的複雑な処理を必要とする．しかし，そのような対象世界のモデル化を行う道具としては，リレーショナルモデルのような単なる表形式ではデータの意味表現能力が不十分であり，また間接的にすぎる．もともと複雑な構造を持った情報を無理に平坦な表形式に変換しなければならないといった不便さがある．

また，オフィス情報システム等においては，データベースの応用の高機能化の要求やマルチメディアデータ（図形，画像，音声）の取扱い等が必要とされている．ネットワーク型やリレーショナルではデータの表現がレコード指向・値指向なので，対象世界の持つデータの意味(たとえば，営業職もSE職も従業員である等といったis a関係やメモリは多くの素子から構成されているといったpart of関係等）をそのデータ構造に十分に反映できない，という弱点がある．

さらに図形や画像，音声のデータは，ビジネス処理でのデータに比べてそのサイズは極めて大きく，処理操作の抽象度も高いという特徴がある．また，設計という仕事では一つのトランザクションが何日にもわたることが普通である．しかも同じデータに対して異なる版を処理する必要がある．

このような非ビジネス処理分野での応用は，複雑な構造のデータに対し複雑な処理を行うという特徴がある．単位時間当たりのトランザクション数はそれほど多くはないが，一つのトランザクションの処理時間は極めて長い．

ネットワークを通してワークステーションを共有するという情報処理環境への変化の中で，データベース管理者，応用プログラマ，エンドユーザといったこれまでの利用者の役割分担も変化しはじめている．従来の三つの役割を，各個人が自分ですべて行う環境が選べる必要がある．また，ワークステーションの持つ高度な対話機能を有効に使用した新しい利用者インタフェースが期待されている．さらに，新しい応用へのコンピュータ適用がデータベース/知識ベース/プログラミング言語の融合を必要としはじめている．

3.2 次世代データベースへの期待

ビジネスデータ処理分野でのデータベース技術の成果は，次のようなものである．

1) 三層スキーマ構造によるデータ独立性の達成
2) データモデル機能による共有データの表現手段と共通操作法の実現
3) データベース管理システムによるトランザクション管理
4) 非手続き的問い合わせ言語と問い合せ最適化機構

応用環境の変化でこれらの成果にも新たな要求が高まってきている．「データ独立性」の概念は，プログラムの部品化というプログラムパラダイムの変化に対応して手続きとの一体化を含めた再定義が必要となっている．「共有データの表現手段」においては，複雑な構造体をより直接的に表現する必要が出てきている．「トランザクション管理」においては，ロングトランザクションの取扱いが問題になっている．また，「非手続き的問い合わせ言語」では，ワークステーションの持つ高度な対話機能を有効に使用した新しい利用者インタフェースが期待されている．

応用環境の変化に対応し，データベースに対する要求に応えるためには，現在のデータベース技術の成果を包含し，次のような要求に応えていく必要がある．

1) 対象領域としてCAD/CAM，CASE，オフィス情報システム等，非ビジネス領域を含むこと
2) 分散環境に適合したものであること
3) データモデル機能として自然に対象を表現でき，複雑な構造体を直接扱えること，そしてデータの意味を十分にデータ構造に反映できること
4) マルチメディア情報が取り扱えること
5) 利用者インタフェースが十分に高度であること

そして既存のデータ資源を有効に利用するために，それは既存のデータベース・システムと共存でき，緩やかな移行が可能であることが望ましい．

4. 次世代データベースの候補

4.1 オブジェクト指向データベース・システム

次世代データベース像をめぐる論議を活発化したきっかけにDOOD '89国際会議で発表された「オブジェクト指向データベース・システム宣言」[1]がある．この宣言の目的は，オブジェクト指向データベースを定義することではなく，共通のデータモデルや理論的基盤が確立しないうちにシステム開発だけが先行している状況に対し，オブジェクト指向データベース・システムの要件をまとめ，オブジェクト指向データベース・システムとは何かということをこれから議論していくための重要な叩き台を呈示することにある．

この宣言では，あるシステムがオブジェクト指向データベースとみなせるために備えていなければならない要件を三つに分類している．

必須条件：あるシステムがオブジェクト指向データベースと呼ばれるために絶対に兼ね備えていなければならない諸要件

付加条件：必須条件ではないが，システムをオブジェクト指向データベースとしてより良いものにするために付け加えることができるいくつかの要件

選択条件：データベースの設計者が，その項目については同じくらい妥当と思われるいくつかの選択肢を持ち，その中から自由に選べるもの

この他に，必須条件または付加条件としてまだ結論が出ていない要件がある．これらをまとめると表1のようになる．

表1　オブジェクト指向データベース・システム宣言の各条件
Table 1　Conditions of object oriented database system manifesto

必須条件	
複合オブジェクト	計算完全性
オブジェクト識別性	拡張可能性
カプセル化	永続性
型/クラス	二次記憶管理
型階層/クラス階層（継承）	並行処理制御
再定義	障害回復
多重定義	アドホックな問い合わせ処理機構
遅延束縛	

付加条件	選択条件
多重継承	プログラミングパラダイム
型検査と型推論	表現システム
分散	型システム
設計トランザクション	統一性
バージョン	

必須条件または付加条件
ビュー定義と導出データ
データベース管理ユーティリティ
整合性制約
スキーマ進化機能

オブジェクト指向データベースの重要な基本概念は，次の三つと考えられる．

1)　実世界の実体の直接表現

オブジェクト識別性，複合オブジェクト（part of関係）

2) 汎化関係の直接表現
クラス階層（is a 関係）と継承
3) データと手続きの一体化
カプセル化

永続性，二次記憶管理，並行処理制御，障害回復，アドホックな問い合わせ処理機構等はデータベース・システムとしては当然の機能である．

分散，設計トランザクション，バージョンは新しい応用のためには重要な概念であるが，オブジェクト指向とは直接には関係ない．

ビュー定義と導出データ，データベース管理ユーティリティ，整合性制約等が未検討項目に分類されているが，これらは必須条件と考えるべきである．

上記1)，2)はネットワークモデルやリレーショナルには無い効用を応用にもたらす．たとえばCADデータベース・システムにおいては，設計対象は複数の部品を内包した複合オブジェクトである．複合オブジェクトやその部品オブジェクトを，どちらも独立した対象として取り扱うのが普通である．リレーショナル・データベースでは，このような対象を表現するには複数の表に情報を分けて持つ．このため複合オブジェクトの取扱いには複数の表の結合処理が必要であり，処理効率や処理のわかりづらさといった問題がある．また，部品に一つずつ異なる識別子を属性値として利用者が与えることは困難である．このためのオブジェクトの存在自体が，その性質を表す属性値とは独立にできるオブジェクト識別性は有効である．更新の波及が生じる場合にもリレーショナルに比べてその処理が簡素化できるという利点がある．

3)はオブジェクト指向データベースの大きな特徴である．オブジェクトの振舞いをオブジェクト自身と一緒に管理するということである．データベース・システムでは，データを応用プログラムからできるだけ独立して管理しようとしてきた．データ独立性がデータベース技術の基底にある．データ独立であるべきなのは応用プログラムであって，抽象データ型でいうような手続きのことではない．この機能により，開発生産性が高まりデータベース・システムの品質も上がることが期待できるが，メソッドという概念で応用プログラムをどこまで管理できるかがその成功の鍵を握っている．現在のところカプセル化を十分に支援しているオブジェクト指向データベース・システムは存在しないようである．

4.2 拡張リレーショナル・データベース・システム

オブジェクト指向データベース・システム宣言に対して，反オブジェクト指向データベース・システム宣言として出されたのが，「第三世代データベース・システム宣言」[2] である．

この宣言では，「レコード」指向により大規模データ管理を築いた階層型/CODASYL型データベース・システムを第一世代，「集合」指向あるいは「値」指向で非手続き的データ操作言語と高度なデータ独立性を提供したリレーショナル・データベース・システムを第二世代のデータベース・システムとして位置付けている．さらに，これを踏まえて次に来るべき第三世代のデータベース・システムが備えるべき条件をこの宣言で提案している．

表2にそれらの条件を示す．また，これらの条件を満たす次世代データベースは，

表2　第三世代データベース・システム宣言の各条件
Table 2　Conditions of the third generation database system manifesto

豊富なオブジェクト構造とルールを扱う機能
・型システムの支援 ・継承機能 ・関数定義機能とカプセル化機能 ・主キーの代替としての識別子の支援 ・トリガーや完全性制約を実現するための独立なルール機能
第二世代データベース機能を包含できること
・非手続的/高水準な問い合わせ機能によるプログラム中のデータベース・アクセス ・集合の外延的記述/内包的記述機能 ・更新可能なビュー機能 ・データモデルの効率向上のための機能からの独立性
他システムからの開放性
・複数の高水準言語からのアクセス機能 ・永続的プログラミン言語支援 ・SQL の支援 ・問い合わせ/検索結果のデータ集合がクライアント・サーバ間の転送単位

現状のオブジェクト指向 DBMS では不十分であり，それよりも既存のリレーショナル DBMS の拡張というアプローチで達成できるという立場をとっている．

この宣言でもオブジェクト指向の良いところは十分に認めており，その実現がリレーショナル・データベースの拡張でも十分にできることを強調している．特徴的な指摘はオブジェクト識別子の選択的な利用とルール機能の必要性を強調している点である．

4.3 標　準　化

現在 ISO や JIS で標準化が進められている活動は次の通りである．

- データベース言語 NDL
- データベース言語 SQL
- 情報資源辞書システム
- データ管理参照モデル
- 遠隔データアクセス

次世代データベースとして新しいデータモデルを基にした標準化作業はまだないが，リレーショナル・データベース言語である SQL の拡張版である SQL2 や SQL3 においてオブジェクト指向の取り込みが意図されている．オブジェクト指向技術のうち重要なもの(たとえばクラス階層と継承，トリガー機能等)は，リレーショナルモデルの拡張として取り込めるという立場である．

1989 年 1 月に ANSI の X3/SPARC/DBSSG のもとに OODBTG (Object Oriented DataBase Task Group) が設立された．その目的は，OODB 技術の開発と利用をさらに進めるため，

- OODB の共通の参照モデルの定義，
- OODB の標準化はどの側面で可能か，また有効かの評定，
- OODB 分野における ANSI/X3 の標準化活動に対する勧告を行う最終レポートの取りまとめ

を行うことである．OODBTGの最終目標は標準化そのものではなく，標準化に向けての調査研究であり，具体的には，オブジェクト指向分野の用語の定義と文献調査，商用や研究用プロトタイプOODBMSの調査，OODBMS参照モデルの作成等を行った．

OODBTGの活動期間は2年間で，上記のような活動結果を最終レポート[7]として取りまとめ，1991年にDBSSGを通してX3/SPARCに提出し，SPARCやX3テクニカルコミティからの質問やコメントに答えた後，1991年7月に解散した．この最終レポートが標準化に及ぼす影響は大きいと予想される．将来，ISOの標準化がもし行われるならば，その米国案の土台となることが予想される．

この他にOMG（Object Management Group）という米国のベンダ/ユーザからなる民間の任意団体があり，オープン性を意図したOODBの共通化の活動が進められている．

5. 次世代データベース・システムの展望

ネットワークを通してワークステーションを共有するコンピュータの利用環境の変化の時代において，これまで築いてきた大規模データの統合・管理，共用の技術にさらに新しい技術が必要になってきている．分散環境に適合するための機能，高度な対話機能に適合したデータベース・インタフェース，メタデータの管理機能等である．また，ビジネス領域から非ビジネス領域への利用の拡大に対応する技術が期待されている．設計トランザクション(あるいはロングトランザクション)，バージョン管理等である．

応用プログラムやデータベースに共通する問題は，対象世界をどう認識できるかということである．オブジェクト指向というパラダイムは，対象世界の認識の手段として「オブジェクト」という新しい概念を提案している．対象の構造的側面だけでなく，その振舞いも一緒に捕らえるというところにその新しさがある．

データベース技術はいろいろな形で発展してきているが，本質的な発展はそのデータモデル機能による．データベースの歴史はデータモデル機能の発展で見ていくのが良い．「レコード」指向により大規模データの管理を築いた「ネットワークモデル」を第一世代，「集合（リレーション)」指向あるいは「値」指向により高いデータ独立性を達成した「リレーショナルモデル」を第二世代とみる．したがって次世代は第三世代であり，それは「オブジェクト」指向により，対象世界の表現能力を拡大する世代である．

現在データベース技術に要求されている機能は，リレーショナルの上にも実現できるものが多い．しかし，新しい応用から必要とされているものは，新しいデータモデルの上で実現するのが適当である．

オブジェクト指向データベースの出現の背景には，オブジェクト指向プログラミングにおけるプログラム実行時の情報保持のメカニズムに対する要求がある．オブジェクト指向概念が与えるデータ表現方法やプログラミング機構を，ディスク上に存在する情報に作用させるものとしてのオブジェクト指向のパラダイムを持つデータベースである．

次世代データベースとして，「3.2節の次世代データベースへの期待」を踏まえてオブジェクト指向データモデルを基礎とした，次のような機能を既存の機能に加えて提供するものを期待する．

1)　データモデル機能

実体を素直に直接表現できる機能で，意味を十分に取り込めるための機能拡張

- オブジェクト，オブジェクト識別子，カプセル化
- 複合オブジェクト
- クラス，クラス階層，継承

2)　データベース管理システムの機能

新しい応用の要請に応えるための機能拡張

- 設計（ロング）トランザクション
- バージョン管理
- スキーマ進化
- 分散環境適合性（クライアントサーバ）

3)　データベースアクセス・インタフェース

利用者インタフェースをより高度にするための機能拡張

- メッセージパッシング
- インピーダンスミスマッチの解消

次世代データベースは既存のデータベース・システムの改善を担うというよりは，新しい応用分野にデータベース技術を提供するものといえる．ビジネスデータ処理においてはリレーショナルが定着しており，SQLが分散環境における共通インタフェースになっている既存の応用領域ではリレーショナルが，そしてCADやCASE，オフィス情報システム等の非ビジネスデータ処理分野にオブジェクト指向データベースが適用されるとみるべきであろう．リレーショナル・データベースに対しては，応用プログラムの利用者インタフェースをオブジェクト指向で実現するということを考えることになろう（図6）．

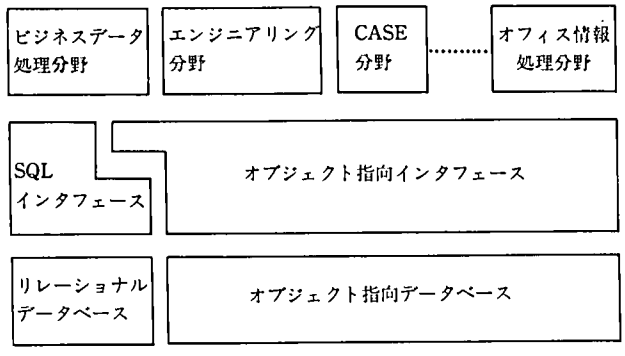


図 6　新しい応用分野とデータベース

Fig. 6　New application domain and database

6. おわりに

次世代データベース論議がさかんである．これは，情報処理環境の急激な変化により，新しい応用分野の実現性が高まってきたからであると思われる．伝統的なビジネスデータ処理分野では，着実にリレーショナルの定着が進んでいる．次世代データベースとしてオブジェクト指向データベースに期待するところは大きいが，それを有効に利用するためには，今から先行的な適用研究を行っていく必要がある．データベースは組織体の貴重な共有資源であり，その運用については堅実であるべきである．データベースの応用分野の拡大と利用の高度化を目指して，弊社でも次世代データベースに向けて積極的に取り組んでいる．

参考文献

[1] M. Atkinson, et al., The Object-Oriented Database Manifesto, Proc. DOOD '89, pp. 40～57, Kyoto, Dec. 1989.

[2] The Committee for Advanced DBMS Function, Third-Generation Data Base System Manifesto, Memo. No. UCB/ERL M90/28, Electronics Research Laboratory, UC Berkeley, 9 April 1990.

[3] Codasyl DBTG, Data Base Task Group Report to the CODASYL Programming Language Committee, Oct., 1969.

[4] ANSI/X3/SPARC, The ANSI/X3/SPARC DBMS Framework, Report of the Study Group on Database Management Systems, Information Systems, 3, 3, pp. 173～191, 1978.

[5] E. F. Codd, A Relational Model of Data for Large Shared Data Banks, Comm. ACM, 13, 6, pp. 377～387, 1970.

[6] E. F. Codd, Relational Database: A Practical Foundation for Productivity, Comm. ACM, 25, 2, pp. 109～117, 1982.

[7] ASC/X3/DBSSG/OODBTG, OODBTG Final Report, 17-Sept., 1991.

執筆者紹介 原　　潔 (Kiyoshi Hara)

昭和44年京都大学理学部数学科卒業．45年日本ユニシス(株)入社．データベース分野の開発・保守に従事．現在，知識システム部部長．情報処理学会/情報規格調査会，日本規格協会/情報資源スキーマ調査研究委員会各委員．東京理科大学理工学部非常勤講師．著書「標準SQLプログラミング」(啓学出版，1990)．

データベース関連標準化におけるデータベース技術 ——SQLとオブジェクト指向データベースに求められる機能

Database Management Technology for Database Standardization ——Functional Requirements for SQL and the Object-oriented Database

中山佐保子

要約 データベース技術に対する標準化は，データベース言語SQLを中心に進められてきた．しかし，SQLはもともとがリレーショナル・データベースのために開発された質問言語である．そのために複雑なデータ構造を扱うことを得意とはしていない．しかし，最近ではデータベース利用者は，実世界の複雑なデータ構造をそのまま取り扱える機能をデータベースに対して要求している．SQLを始めとする各標準化団体は，この要求を仕様に取り入れようと開発を進めている．その一方，利用者の要求を満たそうとリレーショナル・データベースとはまるで異なる概念を持つオブジェクト指向データベースも現われている．

本稿では，リレーショナル・データベースとオブジェクト指向データベースの二つの標準化の流れに着目し，データベースに寄せられている機能要求に対し，どのように標準化を進めていくのかを考察する．

Abstract Standardization efforts for database management technology have been continued with a major interest focused on the database language SQL. SQL, however, is originally meant as a query language for the relational database, thus, proving to be not very powerful in dealing with complex data structure. In recent years, an increasing number of database users have been anxious for functionality which allows complicated data structure in the real world to be handled as it is. With no exception of SQL, individual standardization groups have been pushing to incorporate such requirements into their own specifications. Meanwhile, the object-oriented database with its concept totally different from that of the relational database has come into being so user needs can be fulfilled.

This paper discusses some ways in which to standardize functional requirements considered necessary for database management by addressing two different moves toward the standardization of the relational database and the object-oriented database.

1. はじめに

データベース技術に対する論議は，1970年代のリレーショナル・データベースの出現により一つの大きな山場を迎えた．従来の階層型やネットワーク型のデータベースでは考えられなかった高度なデータ独立性を誇るリレーショナル・データベースは，さまざまな論議を醸し出し，その結果データベース関連の標準化としては初めてのリレーショナル・データベース言語SQLの規格制定作業が始まった．

今日では，システム設計の中核をなすデータベースに対する標準化は，以外な程に遅れていたのである．

1980年代に入り，情報の多用化に伴いシステム設計の手法も従来の処理中心型から，データ中心型へと移っていった．データ中心型のシステム設計を行う上で多くの

利用者がリレーショナル・データベースの必要性を認め，各コンピュータメーカが競ってリレーショナル・データベースを発表したのである．利用者がデータベースに求めたものは，単なるデータの管理ではなく，財産ともいえるすべての情報の管理であった．この要求に応じて，すでに規格制定されたデータベース言語SQLを中心にさまざまなデータベース技術の標準化が活発化してきたのである．

その後，データベースに対する要求はとどまるところを知らず，現在では実世界での複雑なデータ構造を取り扱うためのオブジェクト指向データモデルの研究が活発に行われ，このモデルを実用化したオブジェクト指向データベースも発表されている．標準化作業にも，オブジェクト指向の概念は影響を与えている．

第2章で，現在のデータベース技術に対する標準化作業を紹介する．第3章では，データベース言語SQLの標準化に至る背景とその機能，そしてSQLに対して挙げられている機能要求のうち興味深い項目について述べる．第4章では，オブジェクト指向についての標準化の流れとその内容について述べる．

2. データベース技術標準化の現状

まず，はじめに，現在のデータベース関連の標準化作業にはどのような作業が存在しているかを紹介することにより，標準化作業の現状に触れる．

日本における標準化活動は，大きく分けて，日本国内の標準化作業を行う活動と国際標準を行う活動の2種類がある．

日本国内向けの標準化活動は，国際規格（ISO/the International Organization for Standardization）に対応した日本国内規格（JIS/Japanese Industrial Standards）を作成することを主たる目的とし，日本規格協会のデータ管理調査研究委員会の中で行われている．この委員会には3種類のワーキンググループ(WG)がある．WG 1はデータベース言語SQL 2のJIS素案の作成を行っている．SQL 2は，現在のSQL言語規格に対して，ほぼ完全に上位互換性を保って拡張されたデータベース言語である．

WG 2は，遠隔データベースアクセス（RDA/Remote Database Access）のJIS化作業を行っている．遠隔データベースアクセスとは，異機種のシステム間でデータベースを相互にアクセスすることを意味している．これは，遠隔からデータベースをアクセスするためのサービスとプロトコルを規定することで，適用業務とデータベースシステムとの相互接続を可能にするものである．

WG 3は，情報資源辞書システム（IRDS/Information Resource Dictionary System），およびデータ管理参照モデル（RMDM/Reference Model of Management）に関するJIS化作業を行っている．ここでいう標準化を行うべきIRDSとは，ISOにより次のように定義されている．

「企業がその活動において，資源として重要な情報を格納・管理する場合に利用するシステムソフトウェア製品が情報資源辞書システムである．」

また，データ管理参照モデルは，データベース関連の標準を作成するにあたり，準拠すべき枠組みを規定したものである．

国際標準のための活動は，情報処理学会情報規格調査会SC 21専門委員会WG 3で行われており，その活動は，ISO/IEC JTC 1/SC 21/WG 3に対応する．ここでは，デ

ータベース言語，情報資源辞書システム，データ管理参照モデル，遠隔データベースアクセスの4種類の標準化項目が挙げられている．これらの標準化項目については，先に述べたJISのWG 1からWG 3の中で日本固有の事情も考慮し，日本国内向けに標準化作業が行われているわけである．また，実装ツールとしては，データベース言語SQLが想定されている[1]．

日本だけではなく，世界各国においても，ISOを中心として同じような標準化作業が行われている．その中でもISOに対して多くの提案を行ってきた代表的な団体に，アメリカ国家規格協会（ANSI/American National Standard Institute）がある．現在，ANSIでは，SQLを用いての文章データベースの標準化作業も進められている．文章データベースについては，国際規格原案JISとして，日本国内においても開発が進められている．

標準化作業の開発は，こうして見る限りではSQLを中核において行われているようであるが，その一方では，オブジェクト指向データベースのための標準化活動も活発に行われている．

代表的な活動には，ANSIのOODBTG (Object-Oriented Database Task Group)や，Unisys, Ontologic, DEC, AT & T, Canon, VERSANT, SUN, HP等，オブジェクト指向に注目する多数の企業により組織されたOMG (Object Management Group)がある．

OODBTGはANSI/ASC/X 3/SPARC/DBSSGのもとで，1989年に設立された．このグループの目的は，オブジェクト指向データベースの標準化のための調査研究を行うことにある．1991年9月には，ANSI/ASC/X 3/SPARCに最終報告書が提出されている．内容としては，オブジェクト情報管理のための標準化への要求，オブジェクト・データ管理の参照モデル（Object Data Management Reference Model）の定義，製品として発表されているオブジェクト・データ管理システムの調査等である．ANSIは，この報告書を受けて標準化作業の方向を決定する予定である[2]．

OMGは，ANSIとは別に，各会社で必要とするオブジェクト指向データベースの枠組みを作成している．このグループの活動は，ANSIと直接に関係ないとはいえ，その報告書の内容が製品としてのオブジェクト指向データベースに大きく影響することは確かであろうと予測される．

以上述べたように，現在データベースの標準化では，日本国内，また国際的に見てもSQLとオブジェクト指向データベースの二つの大きな主流に別れている．

3. SQL

3.1 標準化に至る経緯

SQLは，リレーショナル・データベースのために標準化されたデータベース言語である．リレーショナル・データベースは，1970年にE. F. Coddによって発表された論文[3]を基に実現されたデータベース管理システムである．この論文は，数学における関係（リレーション）の集まりによって実世界を表現するリレーショナル・モデルを提案したものであった．リレーションを扱うことで，リレーショナル・モデルはそれまでのハイアラキカル・モデルやネットワーク・モデルでは不十分であったデータの

独立性を高めることに成功したのである．ただし，論文はデータモデルについて述べられており，データベース管理システムに対するものではなかった．論文が発表された後，リレーショナル・モデルを基に多くのデータベース開発者が，今日リレーショナル・データベースと呼ばれているデータベース管理システムのアーキテクチャを開発したのである．

初めてリレーショナル・モデルを具現化したデータベース管理システムは1970年後半に誕生している．一つはカリフォルニア大学バークレー校のグループが開発したUNIX*上で稼働するINGRES，そしてもう一つはIBM社サンノゼ研究所が開発したSystem-Rである．この二つのデータベース管理システムには，それぞれ関係論理を用いた非手続き的なデータ操作言語が提供されている．このうち，System-Rのために開発された言語がSQLの前身である．その後，INGRES，System-Rを皮切りに多くのコンピュータ・メーカがリレーショナル・データベースの製品化に向けて開発を始めた．各データベースごとに非手続き的なデータ操作言語も提供され，リレーショナル・データベースが市場に氾濫するにつれて，リレーショナル・データベース言語の早期標準化が求められるようになった．

リレーショナル・データベースに関する標準化作業は1981年にANSI/X 3/SPARCのタスクグループにより，リレーショナル・データベース言語の標準化についての最終報告書が提案されたことから始まった．ANSIはこの報告書を受けてデータベース言語SQLの標準化を開始するのである．

ANSIでは，1986年10月にデータベース言語SQLをアメリカ国家規格（ANSI X 3. 135-1986）および連邦情報処理規格（FIPS/Federal Information Processing Standard）として規格制定した．このSQLはISOでは1987年にISO 9075として制定，日本では，1987年11月にJIS X 3005として制定されている．これが，データベース関連における世界で初めての規格化であった．1986年に制定されたSQLでは，スキーマ定義，ビュー，権限の定義，データ操作，モジュール言語等，既存の処理系が持っている機能のみが規格化されている．これは，SQLの機能の充実よりも先に適用業務プログラムの可搬性を確保するための早期標準化を目的としたためである．1989年には，ISOによってSQL補遺1と呼ばれる規格の拡張がなされた．これは，整合性に対する拡張機能であり，その内容は次に示す通りである．

1） 満たされなければならない表間の参照制約（参照整合性定義）
2） 表の行に適用される検査制約（検査句）
3） 表の中に列の値を指定しない行が挿入される時にとられる値（DEFAULT句）

これらの整合性制約は，データベースにデータを格納する上で，リレーショナル・データベースが持つべき必要最低限の制約といえよう．さらに，現在のJIS規格（JIS X 3005-1990）では，SQL補遺1の整合性拡張機能に加えて，データ型の拡張である漢字型を導入し，JIS SQLとして制定している．以下，本稿において，このSQLとSQL補遺1を含めて標準SQLと呼ぶことにする[4]．

* UNIXオペレーティング・システムは，UNIX System Laboratories, Inc. が開発しライセンスしている．

3.2 データベース言語 SQL 2

標準 SQL を規格化するための審議が始まった 1985 年には，この SQL に対する拡張機能の開発がすでに始まっていた．最初の拡張機能の制定内容が，前節で触れた SQL 補遺 1 である．その後，拡張機能に対する開発コードを SQL 2 とし，今日に至っている．SQL 2 の JIS，ISO の規格制定時期は 1992 年が予定されている．

SQL 2 は，実用面での機能を取り入れることに重点が置かれて開発が進められた．そのために，非常に多くの機能が取り入れられ，その仕様は標準 SQL の倍以上にもなる．さらに，ISO，JIS のデータベース関連の標準化作業は，今日，データベース言語に SQL を採用することを念頭に作業が進められている．このことを考慮に入れた開発であることも，仕様が増えた一つの理由であろう．

SQL 2 と標準 SQL との主な技術的変更内容を，以下に簡単に触れる．仕様の各詳細は，現在公開レビュー中である SQL 2 の JIS 素案を参照されたい．

1) データ型の拡張……漢字列型(JIS では導入済)，日時型，時間隔型，ビット列型の追加．

2) 値式の拡張……CURRENT_TIME, CURRENT_TIMESTAMP 等の日時に関する関数の導入や問合せ式中に CASE 文を指定することで条件による分岐に従った動作が可能となった．

3) データ型間の変換……CAST 演算を用いることで，プログラム言語で支援していない SQL のデータ型をプログラム言語のデータ型に SQL 中で変換することができる．

4) 文字集合，照合順番の追加……スキーマ中での文字集合定義が可能，また照合順番が利用者によって定義可能．

5) 整合性制約機能の拡張……参照制約に違反した場合の動作指定，複数の表間の値の検査を行う表明機能の追加，定義域に対する検査制約定義の指定が可能．

6) 情報スキーマの定義……利用者に参照可能なデータベースの構造と内容についての情報を，その利用者が利用可能なようにスキーマ中に作成する．

7) 動的 SQL……文字列として与えられた SQL 文を直接実行することが可能．

8) SQL の直接起動……対話型問合せ，ネットワークを介しての SQL 言語プログラムといった処理系に対する規定の明確化．

9) 診断機能……誤り発生時に返される標準誤りコードの規格化，および詳細な誤りについての情報を得るための診断修得文の追加．

10) 遠隔データベースで必要とされる機能の提供．

11) 集合演算の拡張……UNION (和)，EXCEPT (差)，INTERSECT (積) の各演算が直接指定可能．また，外結合，自然結合，結合条件の指定が可能となった．

12) トランザクションの一貫性水準の提供……トランザクション状態を読出し専用，更新可能と指定することができ，トランザクションのスケジュールやロックに関するより細かな管理機能を提供．

13) データ操作……問合せ式の結果を実行中に一時表として定義可能．また FETCH の拡張により，カーソルの移動位置を行の最初，最後，次行，前行，絶体位置と指定が可能．

14) スキーマ操作機能……スキーマ定義を削除,変更するためのスキーマ操作言語の追加.

15) ホスト言語……埋め込み構文に Ada, C, MUMPS を追加.

16) モジュール言語……すべての SQL 文をモジュールの手続きから呼び出すことが可能[5],[6].

3.3 SQLへの要求

リレーショナル・データベースが発表されて以来, リレーショナル・データベースに対する期待とその限界について多くの論文が発表されたり, 利用者からの批判もあがっている. その中には, リレーショナル・データベース自身に対する批判と SQL の規格からなる問題の両者が含まれる. ここでは, 標準 SQL に対して求められている機能を中心に, リレーショナル・データベースおよび SQL 2 では, その機能が実現可能かどうかを考える.

標準 SQL に対する批判の一つに整合性制約の甘さがある. この点は, SQL 2 でかなり強化されていると言えよう. SQL 2 での整合性制約には, 参照整合性制約, 検査制約定義, DEFAULT 句, 表明がある.

参照整合性制約は, SQL 2 では, SQL 補遺 1 で追加された主キーと外部キーに加え, 参照制約を破った場合の動作指定ができる. 次の例では, 部門表に存在する行を削除すると, その行に対応する社員表の行が削除されるという動作が行われる.

```
部門（部門番号，部門名）
社員（社員番号，社員名，給与，所属部門番号）
FOREIGN KEY （所属部門番号）
  REFERENCES 部門 ON DELETE CASCADE
```

また, 部門表に存在する行を更新すると, その行に対応する社員表の行が更新されるように指定することも可能である. CASCADE に代わり, SET NULL を指定すれば, 参照整合性制約に違反すると, 所属部門番号には NULL 値が設定され, SET DEFAULT を指定すれば, 既定値が設定される.

検査制約定義は, 列の値の範囲を検査する制約の定義である. 列の値が検査制約に違反すると誤りが返され, その値はデータベースに反映されない. 次に示す例は, 給与の値が 10 万円から 40 万円までしか取りえない場合の検査制約定義である. 仮りにある社員の給与に 100 万円を割り当てようとしても検査性制約により違反が返されるのである.

```
CREATE TABLE 社員
  (社員番号      CHAR (4)  PRIMARY KEY,
   社員名        NCHAR (10),
   給与          DECIMAL (6),
   所属部門番号  CHAR (4)  REFERENCES 部門,
   CHECK
     (給与 BETWEEN 100000 AND 400000))
```

DEFAULT 句は, 既定値として入れる値を指定する. 通常の定数以外に, 日時値関数, USER, SYSTEM USER, NULL を指定できる. 表明の機能を使用することで,

検査制約を複数の表にまたがって定義することも可能である.

第2番目に抽象データ型を支援していないという指摘がある.

データとそれに対する手続きを一体化した抽象データ型は，もともとプログラミング言語から誕生した考え方である.従来のプロセス中心と呼ばれるシステム設計では，データが手続きに従属する形でプログラミングが行われていた．データが手続きごとに存在するために，データの論理構造に変更が生じると，それに伴って膨大な量の手続きの変更が必要となった．けれども，抽象データ型を取り入れることで，手続きはデータに付随することになり，データに対しては，その決められた手続き以外ではアクセスできないことになる．これは，情報隠蔽ともカプセル化とも呼ばれる考え方であるが，これをデータベースに取り入れることで，複雑なデータ構造も利用者が簡単に利用できるようにしようというものである．ただし，抽象データ型はデータと手続きが独立して存在することは許されない.

ところで，リレーショナル・データベースの非常に強力な利点は何であったかという問題に立ち返ると，それはデータ独立にある．リレーショナル・データベースの高度なデータ独立性はデータをすべて2次元の表形式で捕え，データと手続きを分離している．そして，データの操作言語であるSQLは，どのようにデータを得るのかという方法は問わずに，何が欲しいのかを指定するだけの非手続き性を特徴としているのである．これにより，リレーショナル・データベースには，一切のデータに対する振る舞いは含まれずにデータの静的な部分だけが格納されている．データに対する振る舞いをSQLで拡張し，リレーショナル・データベースに適用したら，高度なデータ独立性，非手続き性が保証されるとは限らないだろう.

第3番目に，汎化を支援していないという指摘がある．汎化を支援するということは，表に対して副表を設けるということであるが，もともとリレーショナル・モデルでは副表という概念は扱っていない．副表とは，一つの表に対して部分集合にあたる表を考えるという意味である.

ここで社員と営業社員についての関係をリレーショナル・モデルで表現してみる.

社員（社員番号，社員名，給与，住所）

営業社員（社員番号，営業成績）

営業社員は社員という集合の部分集合であるため，営業社員表は社員表の副表と考えられる．営業社員表に存在する営業社員一人の情報をとると，その人についての情報は必ず社員表にも存在する．つまり営業社員表の社員番号に存在する値は，必ず社員表の社員番号に存在しているはずである．二つの表にあるそれぞれの行の社員番号が等しければ，情報が二つの表に分離していてもそれぞれの行は同一人物の情報を表しているという意味になる．ただし，同一人物であることを保証するのはあくまでも利用者である．リレーショナル・データベースが主キー，参照キーとともにその値を指定することを利用者にまかせている以上，これは仕方がないことである.

表と副表に関係するデータベース操作を考える．たとえば，ある営業社員が退職した場合は，情報は営業社員表から削除するのではなく，親表である社員表からも削除する必要がある．逆に一人の社員が営業として入社した場合に，社員表に社員の情報を追加するだけではなく，営業社員表にも追加する必要がある.

さらに，汎化には属性の継承の概念がある．属性の継承とは，親表の属性が副表に引き継がれるということである．例では，社員表の属性はすべて営業社員表へ引き継がれることを指す．ただし，社員表の社員番号を営業社員表へ継承させることにより，営業社員表には同じ意味を持つ同じ属性が2種類存在してしまう．この場合のように主キーを継承させるには，何らかの制約が必要になるであろう．当然，継承もデータベース操作に大きく影響してくる．営業社員に対して一律10%の昇給を行う場合，給与は社員表から継承されているはずなので，営業社員表の給与に対して更新を行えばよいということになる．

リレーショナル・データベースで汎化を支援するには，表と副表の結合が必要になるために，新たに汎化のための参照整合性の保証も必要になるだろう．

第4番目に再帰的な問い合わせを支援する必要性が挙げられている．

再帰的な問い合わせは部品展開を必要とする適用業務から強く要求される．SQLでは部品展開のような親子関係を表現するには，次のようにモデル化する．

部品（部品番号，部品記述，値段）

部品要素（親部品番号，子部品番号）

“現在，使用している部品の親部品はどれか”といった問い合わせをSQLで行うには，部品要素表の子部品番号と部品表の部品番号で結合操作を行い，その結果から親部品番号を得るという手順が必要である．ある部品の親部品の親部品というように問い合わせが複雑化してくれば，その分だけ利用者が行う結合操作は増加する．しかし，SQLが定義域属性の考え方を導入すれば，比較的楽にこの問い合わせを行うことができるであろう．

```
CREATE  TABLE  部品
            (部品番号  CAHR (4)   PRIMARY  KEY,
             部品記述  NCHAR (10),
             値段      DECIMAL (6),
             親部品    部品,
             子部品    部品
            )
```

従来のデータ型に表名を与えることで，部品表に対して親部品を持つという表間の関連を示すことになる．関連をスキーマ上で定義できれば，その定義に従い，今まで利用者が行っていた結合操作をデータベース管理システムにまかせてしまうことができる．ただし，これはSQL 2の仕様には含まれていない考え方である．

標準SQL，SQL 2ともにデータベース言語とはいえ，リレーショナル・データベースを想定して規格化されていることに違いはない．今まで述べた拡張をリレーショナル・データベースに施した場合，そのデータベースは果たしてリレーショナル・データベースと呼べるのであろうか．逆に考えればリレーショナル・データベースの特徴を残したままでこれらの拡張を行うには，SQLに対してかなり大きな仕様の変更が必要になると考えられる．

4. オブジェクト指向データベース

4.1 標準化への背景

1980年に入るとリレーショナル・データベースは商用データベースとして，広く普及し始めた．その裏では，リレーショナル・データベースは事務処理には適するが，複雑なデータ構造は，2次元の表に変換することでそのデータの持つ意味を表現することができないといった批判が聞かれるようになった．このような批判が表面化したのは，コンピュータの扱う処理が事務処理にとどまらず，図形，画像，音声も情報として取り入れたい，定型業務だけではなく，実世界の情報そのものを扱いたい等と多くの期待がコンピュータ処理に向けられたためである．実世界の意味を表現する機能を持つデータモデルの研究は1970年後半からすでに始まっていたが，これらのモデルは意味データモデルと呼ばれた．意味データモデルは，レコード，表，行といった用語を一切使用せず，データベースが表現する実世界をすべてエンティティ（実体）と関連で表そうとするモデルである．このモデルでは，汎化，集約化等の実世界を表現するために必要な論理データ構造の抽象化手段が豊富に提供されている．ただし，意味データモデルと呼ばれるモデルは論理データ構造の構造体の部分のみに着目し，操作については支援していない[7]．そこで，注目されたモデルがオブジェクト指向モデルである．

オブジェクト指向でいうオブジェクトには，従来のようにデータ構造の抽象化だけではなく，そのデータ構造に対する操作の抽象化が含まれている．実世界に存在する実体は，自分がどのように動くのかという振る舞いを知っているはずである．オブジェクトとは，実世界に存在する実体のことであり，その中には従来のようなデータ構造の抽象化だけではなく，自分がどのように振る舞うかといった操作も抽象化されているのである．これにより，利用者にとって必要なことは，利用者が欲しい情報は何かということだけで，その情報を得るための操作は必要ではなくなるのである．

現在の情報化社会では，処理を必要とする情報が非常に複雑なデータ構造を持っている場合がある．データベースの分野だけではなく，プログラミング言語，システム設計の分野でも複雑な情報の処理には同じような問題意識を持っている．オブジェクト指向がデータベースの分野のみならず，それぞれの分野から注目されたのは偶然ではないであろう．オブジェクト指向データベースはオブジェクト指向プログラミング言語の考え方に大きく影響を受けて出現した．当初，データベースの分野におけるオブジェクト指向の議論は，プログラミング言語として必要とされている機能と交錯し，理解に苦しむものがあった．これは，オブジェクト指向がリレーショナル・モデルのように一つの確固たる理論の上に基づいているわけではないことに原因があるようである．

1989年12月に京都で第1回演繹・オブジェクト指向データベース国際会議が開催された．その中で6名の研究者によって発表された「オブジェクト指向データベースシステム宣言（The Object-Oriented Database System Manifesto)」は大きな反響を呼び，その後，この論文に対するさまざまな議論がなされた．その中でも有名な論文に，1990年4月にストーンバークレー他5名の研究者によって発表された「第3世代データベース宣言（Third-Generation Data Base System Manifesto)」がある．

前者がオブジェクト指向データベースに備わっていなければならない条件を列挙しているのに対して，後者は次世代データベースが持つべき機能についてリレーショナル・データベースを拡張する形で論じている．オブジェクト指向データベースは，この両論文に代表されるように，従来のデータベースとはまるで異なる新しいデータベースとして捕えようとする考えと，リレーショナル・データベースを拡張する，すなわちSQLに対して拡張を行おうとする考えの2種類がある．

現在，オブジェクト指向データベースも商品として出回り始めているが，オブジェクト指向データベースについての理解に統一が取れていないことも事実である．このために，OODBTGやOMGを代表とする標準化の動きが出てきているのである[8][9]．

4.2 OMGにおけるオブジェクト・モデル

オブジェクト指向データベースについて，その機能やメカニズムを詳細に規定した規格書はまだ存在しない．ただし，開発を進めていく上での枠組みは，OMGが作成している．ここでは，OMGが報告しているオブジェクト・モデルを中心に，第3章で述べているSQLの問題をオブジェクト指向ではどのように取り扱えるのか，を考察する．

OMGのオブジェクトとは適用業務中の対象物であり，共通の特性を持つオブジェクトの集まりを型と呼んでいる．上位の型をスーパタイプと呼び，スーパタイプからある共通の特性でさらに類型化したオブジェクトの集まりをスーパタイプのサブタイプと呼んでいる．スーパタイプの特性はすべてサブタイプに継承される．適用業務中で利用者が使用するオブジェクトを類型化することとは別に，OMGではシステムが提供する型の階層構造をオブジェクト・モデルとして定義している．

以下，本稿においてオブジェクト・モデルは図1に示すOMGが提供するモデルを指すものとする．

```
TYPE u_object
      subtype : Object
              : characteristic

TYPE Object
      supertype : u_object
        subtype : Thing
                : Literal
                : Collection

TYPE Thing
      supertype : Object

TYPE Literal
      supertype : Object

TYPE Collection
      supertype : Object

TYPE characteristic
      supertype : u_object
      subtype   : Operation
                : property

TYPE Operation
      supertype : characteristic

TYPE property
      supertype : characteristic
      subtypc   : Attribute
                : Relationship

TYPE Attribute
      supertype : property

TYPE Relationship
      supertype : property
```

図1 オブジェクト・モデルの提供する型

Fig. 1 Predefined types in the object model

図1では，オブジェクト・モデルの型階層構造をデータベース記述言語の構文に似せて表現している．英大文字で始まる型にのみ，直接，オブジェクトのインスタンス（実現値）や特性が含まれ，英小文字で始まる型には直接，存在するオブジェクトはない．

u_object 型は，型の階層構造の根にあたる．これが，通常オブジェクト指向が表現しようとしているオブジェクトを意味する．この型は Object と characteristic から構成される．Object は，Thing, Literal, Collection から構成される．Literal は文字や数のようにシステムによって定義されている型を示す．Collection は，オブジェクトの集合を表す型である．Thing は，従来，われわれがデータベース設計における実体と呼んでいたものに近い概念であり，オブジェクト識別子を持つ．オブジェクト識別子とは，Thing が発生した時にシステムによって作成される識別子である．この識別子がどのようなものかは，利用者が知ることはできない．オブジェクト識別子を参照できるのはシステムだけであり，システムはこの識別子を使用してオブジェクトの管理を行う．オブジェクト識別子が存在することで，持っている値がすべて同じである二つのオブジェクトも別オブジェクトであるとシステムが認識できる．オブジェクト・モデルでは Thing が削除されても，そのオブジェクト識別子はシステムにより管理されている．システムはオブジェクトが発生し，消滅したという事実もこのオブジェクト識別子で判断できるのである．

characteristic は，オブジェクトの振る舞いを行う Operation とオブジェクトの状態を示す property から成る．オブジェクトは，外部から何らかのメッセージを受け取った時にどのような振る舞いをすればよいのかをこの Operation により理解している．このために，利用者はオブジェクトに対する操作を知らなくても必要とするオブジェクトの値を得ることができる．property は，そのオブジェクトの属性名やオブジェクト間にはどのような関連が存在しているかといった特徴を持っている．characteristic はオブジェクト識別子を持たない．しかし，Thing の特徴は，この characteristic により理解できる．

ここで，前章で述べた SQL に要求されている機能について，抽象データ型，汎化，再帰的問い合わせ，整合性制約の順にオブジェクト指向ではどのように対処できるかをオブジェクト・モデルを用いて考察してみる．

SQL で支援ができない抽象データ型は，オブジェクト・モデルでは，characteristic が存在するために容易に実現できる．静的なデータ構造は property に持ち，振る舞いの部分は Operation で表現している．データは，characteristic によりカプセル化されるために，論理データ構造の構造体に利用者が直接触れることなく，正しい情報を利用者に返すことができる．

汎化について考える．OMG では，スーパタイプとサブタイプにより汎化と継承を支援している．前章の例を適応する．

スーパタイプ……社員（社員番号，社員名，給与，住所）

サブタイプ　……営業社員（営業成績）

リレーショナル・モデルでは，同一人物であれ，2表に情報を分離し，それぞれの表の識別子の値を用いて同一人物であることを利用者が保証する．オブジェクト・モデルでは，オブジェクト識別子を用いて，同一人物であることをシステムが保証する．さらに

属性が上位の型から下位の型へ引き継がれるという特徴のために，サブタイプとスーパタイプに対して利用者が結合操作を行う必要がない．表と副表によって汎化を支援する場合には，参照整合性制約の保証が必要であったが，オブジェクト・モデルでは，整合性はすでに保証されていると考えられる．

再帰的問い合わせでは，定義域属性の概念が必要ではないかと述べた．オブジェクト・モデルでは，型に定義域属性の考え方を用いている．このためにSQLでいうデータ型の部分に，オブジェクト間の関連を定義することが可能になる．オブジェクト・モデルのpropertyは二つのサブタイプを持つ．片方のサブタイプであるAttributeは従来のデータ項目や属性名と呼ばれるものと等しいが，もう片方のRelationshipが定義域属性にあたる．

以下の例はOMGの報告書で示されている例である．

```
type  PROFESSOR
      relationships:
            advisees: {STUDENT}  <——>>  STUDENT. thesis_advisor
type  STUDENT
      relationships:
            thesis_advisor: {PROFESSOR}  <<——>PROPESSOR. advisees
```

教授（PROFESSOR）が指導する（advisees）相手は学生（STUDENT）である．教授は学生に指導するという関連をもっている．例において矢印は逆関係とその発生頻度を示している．この意味は教授が指導する学生は多数存在し，学生から教授へは指導者（thesis_advisor）という関連が存在しているということである．つまりPROFESSORの持つ属性adviseesの型はSTUDENTであり，これはadviseesがSTUDENTへの関連であることを示しているのである．オブジェクト操作言語についてOMGでは何の規定もしていないが，スキーマ上に定義するためのモデルが確立していれば，操作言語もそれに伴って開発されると考えられる．たとえば，教授が指導している学生に対する情報が必要な時は，利用者がオブジェクト間で結合操作を行わなくても，オブジェクトの持つ関連をたどるだけで容易に得ることができるはずである．前章の部品展開の例で示したような再帰的問い合わせについては，定義域属性が存在することで次のように表現できる．

```
type  PART
      relationships:
            parent_part: {PART}  <<——>>  PART. child_part
            child_part: {PART}  <<——>>  PART. parent_part
```

OMGでは，整合性制約については明確には規定していない．しかし，オブジェクト識別子によるオブジェクト間の航行巡回は参照整合性を保証するであろうし，豊かな型の概念は整合性制約を十分に保証できるであろう．

今後，OMGでは，次の段階のオブジェクト・モデルについて開発を進めていく方針である．その中にはオブジェクト定義言語，オブジェクト操作言語，オブジェクトの世代管理といった項目も挙げられているようである．オブジェクト指向データベースの持つべき機能はこれによって，さらに明確になっていくであろう[10][11]．

5. おわりに

現在, ISO では SQL 2 の拡張である SQL 3 の開発が進められている. この SQL 3 では, SQL が不得手としている分野も含めての要求項目が各国から寄せられている. 副表と汎化の支援, 再帰的問い合わせが可能となるような結合操作の支援, 抽象データ型の支援等は, 真っ先に SQL 3 への提案として挙げられている. SQL 3 が現在の SQL の形式をとどめるものか, それともオブジェクト指向データベースの操作言語として採用され, SQL とはまるで異なる形式をとるものなのかは, 明らかではない. ただ, 言えることは, データベースに対する利用者の要求は, リレーショナル・データベースであれ, オブジェクト指向データベースであれ, 似たような機能なのかもしれないということである.

弊社においても, ネットワーク・モデル, リレーショナル・モデル, セマンティック・モデルを基に開発した多種多様なデータベースを市場に提供している. 各データベースがどのような理論に基づいて, 何をめざして開発されたか, また標準化されようとする流れに対してどのような位置付けにいるのかを理解することは, 複雑化するデータベース技術の中で, 今後より一層必要となるであろう.

参考文献

[1] (財)日本規格協会, 「カラーディジタル画像システムの標準化に関する調査研究 (データ管理調査研究) 報告書」 平成 3 年 3 月.

[2] Accredited Standards Committee X 3, INFORMATION PROCESSING SYSTEMS DBSSG/OODBTG Final Report, 17-September-1991.

[3] E. F. Codd, A Relational Model of Data for Large Shared Data Banks : Communications of the ACM, Vol. 13, No. 6, Jue 1970 P. 377.

[4] 日本工業規格, データベース言語 SQL, JIS X 3005-1990.

[5] 日本工業規格(改正素案), データベース言語 SQL 2, X 3005-199 X.

[6] DIS 9075 : 199 X, Database Language SQL-April 8, 1991, ISO/IEC JTC 1/SC 21 N 5739.

[7] M. Hammer, and D. Mcleod, Database Description with SDM : A Semantic Database Model, ACM Trans. on Database Syst., Vol. 6, No. 3, pp. 7～92 Sep. 1981

[8] M. Atkinson, F. Bancilhon, D. Dewitt, K. Dittrich, D. Maier, S. Zdonik, The Object-Oriented Database System Manifesto, Proceedings of the First International Conference on Deductive and Object-Oriented Database (DOOD' 89), pp. 40～57 (Dec. 1989).

[9] The Committee for Advanced DBMS Function : Third-Generation System Manifesto, Memorandum NO. UCB/ERL M 90/28, University of California at Berkeley (Apr. 1990).

[10] Object Managment Group (OMG) Object Model Task Force (OMTF) draft 0.9 Septmber 3, 1991 OMG Document Number 91.9.1.

[11] Won Kim : Introduction to Object-Oriented Database, The MIT press (ISBN 0-262-11124-1).

執筆者紹介　中　山　佐保子 (Saoko Nakayama)

昭和 60 年東京電機大学理工学部数理学科卒業. 同年日本ユニシス(株)入社. 人材開発本部システム教育課を経て, 現在, システムプロダクト本部ソフトウェア 4 部基本ソフトウェア 4 課にて A シリーズデータベース管理システムの開発, 保守業務に従事. データ管理調査研究委員会 WG 1/SQL 2 委員. 情報資源スキーマ調査研究委員会　オブザーバー.

ユニシスのデータベース・アーキテクチャ
——Unisys Architectureにおけるインフォメーション管理サービス

Unisys Database Architecture
——Information Management Services Provided by the Unisys Architecture

藤島忠篤

要約 米国ユニシス社は，1990年10月にUnisys Architecture（UA）を発表し，ユニシスが1990年代に提供するソフトウェア・プロダクトの開発方針と機能を提示した．

UAで提示される機能は，五つの機能群（サービス）に体系化されており，その一つであるインフォメーション管理サービス（IMS）は，データベース管理およびリポジトリに関する機能と他のサービスとのインタフェースを規定している．また，これらの機能を実現するデータモデルには，オブジェクト指向モデルが有効であり，従来から提供しているリレーショナルモデルやネットワークモデルと共存して実現させていくことを宣言している．

本稿は，UAの概要とIMSの役割を述べて，ユニシスのデータベース分野での方向性を紹介する．

Abstract In October 1990, U. S. Unisys announced its Unisys Architecture concepts, making clear the strategy of its software products development, including the functionality of its future products planned for release in the 1990s.

The functionality provided or to be provided by UA is divided into five different groups of functions (services). Information management services (IMS), one of the five, defines the functionality related to database management and repository in addition to interfaces to other services. Placing stress on the effectiveness of an object-oriented model as a database model where those functions are (or are to be) implemented, U. S. Unisys has claimed that object-oriented models would become available in co-existence with relational database models and network models so far released to the market.

This paper is intended to show the direction in which U. S. Unisys plans to move in the field of database by giving a brief description of UA and giving a clear picture of IMS's roles.

1. はじめに

米国ユニシス社（以下ユニシス）は，1990年10月にUnisys Architecture（UA）を発表し[1]，90年代に求められる新しい情報処理システム像を提示した．UAで述べる情報処理システム像は，これからの企業情報システムの開発および実行の基盤となるコンピュータ環境のあるべき姿と，ユニシスが提供するソフトウェア・プロダクトの開発方針を明確にすることで示されている．

今後，ユニシスから提供するソフトウェア・プロダクトは，UAの開発方針に従った機能を具備していくことになる．UAで規定する機能群は，サービス*と呼ばれ，アプリケーションを支援する五つのサービスに体系化されている（図1）．

* UAでは，サービスという用語を幅広く使用する．UAで規定する機能そのものをサービス（service）と表現し，体系化された機能群もしくはその部分集合をもサービス（services, service set）と呼ぶ．

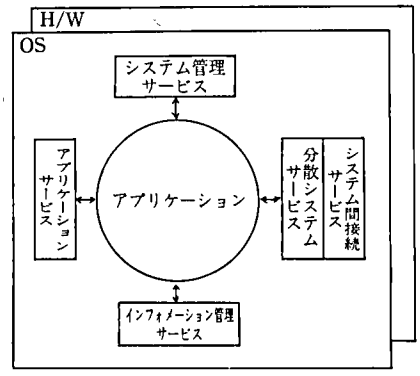


図 1　アプリケーションを支援する五つの機能群

Fig. 1　UA-5 services related with applications

本稿は，UA の概要を簡単に説明し（2章），UA のサービスの一つであるインフォメーション管理サービスが果たす役割を述べる（3章）．インフォメーション管理サービスは，企業情報システムのあらゆる情報資源を効率よく蓄積し，効果的に検索・利用するための機能と他のサービスとのインタフェースを規定している．つまり，インフォメーション管理サービスは，基本的にデータベース管理およびリポジトリに関する機能を集約すると言える．しかし，従来のように単一環境下でのみ有効な範囲での実現を行うのであれば，UA が持つ目的を満たすことができない．UA では，各サービスで規定する機能をメインフレームやワークステーション等のハードウェア環境に共通して提供することを基本としており，ハードウェア規模やユーザの利用状況に順応した情報処理環境とユニシスが持つ固有の技術を活かした最も効果的な情報処理環境構築の実現を図っている．

2. UA の概要

2.1 UA の背景

90 年代の企業情報システムのあり方についての模索は 80 年代より続けられ，80 年代の後半にはコンピュータ・メーカの各社より，各々の特徴を備えたアプリケーション・アーキテクチャが提示された．これらは，メーカ固有のソフトウェア技術の固定化にともなうアプリケーション資産の分散化と急激に増大するユーザのアプリケーション開発要求への対応を背景としている．

一方，OSI (Open Systems Interconnection) を中心とした標準化の動きは，日本においても INTAP (INteroperability Technology Association for information Processing, JAPAN) による三度の公開実験の成功でネットワーク技術の浸透とオープン化への拍車がかかり，公表された実装規約ベースでの製品提供のバックボーンとなっている．

しかし，現実に提案されている OSI でのサービス*とプロトコルの提供のみでは，大

* OSI でのサービスは，UA のサービスとは違った意味合いを持っている．OSI の場合，直上位層もしくは直下位層との間で機能のインタフェースを規定する意味が強い．

規模化，複雑化する企業情報システムが必要とする課題をすべて満たすとは言いきれない．このことは，アプリケーション・アーキテクチャを具体化するコンセプトあるいはフレームワークの提言が順次行われ，改善に向けたアプローチがなされていることからも，うかがうことができる．

ユニシスは，企業が適材適所で採用していくコンピュータ資源を有効に活かした情報処理環境を「インフォメーション・ネットワーク」と定義し，このインフォメーション・ネットワークが備えるべき要件の実現に向けてのユニシス・アーキテクチャ(UA) を提示することで，企業情報システムにおける課題への対応を図っていく方向性を示した．

インフォメーション・ネットワークの基本的要件は，以下の通りである．

1) 相互運用性……分散環境，異機種環境におけるシステム間の相互運用性および相互接続性が高いこと．
2) 共通性……エンドユーザからの操作が共通化され，利用しやすい環境が構築できること．また，プログラミング・インタフェースが統一され，高い開発効率とプログラムの移植性が保証されていること．
3) 結合化……分散された複雑な情報資源を，統合的に管理し，運用が可能なこと．また，全社にまたがる情報資源の共有化が可能なこと．
4) 透過性……エンドユーザは，分散された情報資源の物理的な存在場所を意識することなく，利用が可能なこと．

2.2 UAのねらい

UAは，企業情報システムを構築・実行・運用していく上で，柔軟性に富みかつ最も効率よい環境を可能にすることをねらいとしている．UAで規定されるサービスが，前述のインフォメーション・ネットワークの要件を満足するように定められ，また，インフォメーション・ネットワークを構成するコンピュータ資源（プラットフォーム）の役割を明確にすることで，以下に示すような融通性・拡張性のある基盤構築を可能としている．

1) 異機種間の相互運用性の向上……プラットフォームが備えるべき機能とインタフェースに一貫性を持たせて，異なるプラットフォーム間での円滑な連携処理を可能とする．このため国際標準を基本とした分散処理と資源の共用機能を採用し，異なるベンダとの負荷の少ない相互運用性，相互接続性を高める．
2) ユーザ・インタフェースの共通化……X/Open*で提唱される共通アプリケーション環境（CAE: Common Application Environment）に基づいたオープン・システム標準仕様を採用し，アプリケーション・プログラムの移植性，接続性の向上を図る．
3) システム管理の統合化……複雑なインフォメーション・ネットワーク全体の情報資源を統合的に，かつ集中的に管理・運用することを可能にする．
4) 分散環境の透過性……プラットフォーム間のインタフェースは，アプリケーション・プログラミング・インタフェース（API: Application Programming Interface) と独立に規定され，物理的な配置を意識することなく，エンドユーザがイン

* X/Openは，X/Open社の登録商標である．

フォメーション・ネットワーク上の資源を利用することを可能にする.

5) 開発生産性の向上……システム開発のライフサイクル (SDLC: System Development Life Cycle) の全工程に対応した CASE/4GL (Computer Aided Software Engineering/4th Generation Language) 等の開発支援ツールの適用およびリポジトリによる管理情報の共用化と有効利用を可能とする.

6) 高付加価値機能の提供……ユニシス固有の CASE/4GL, 高速トランザクション処理機能, 無停止連続運転機能等の先進的な技術を採用し, 個々のプラットフォームの利用価値を高める.

7) ユーザ資産の継承……すでに蓄積されているユーザのアプリケーション・システムを新しい環境でも有効に活かすため, プラットフォーム固有の既存機能を継続する. このことで, 各プラットフォームの性能を最大限に活用しながら, 既存アプリケーション・システム資産の継承を可能にしていく.

2.3 UA の特徴

UA は, ユーザにとって効果のあるインフォメーション・ネットワークの構築を可能にする以下の特徴を持っている.

1) 三つのプラットフォーム……ネットワークを構成するコンピュータ資源の役割を位置づける.

2) 五つのサービス……各プラットフォームが備えるべき機能を規定し, 分類する.

3) 三つのサービス・クラス……各サービスで規定される機能の適用目的を明確にする.

2.3.1 三つのプラットフォーム

UA は, 情報システムの基盤を実現するコンピュータ資源 (プラットフォーム) を三つのグループに分割し, インフォメーション・ネットワーク内での役割と位置づけを明示した (図 2).

これは, 部門間に拡がっていくサーバとワークステーションの必要性を認識するとともに, これらを統合するメインフレームの重要性がますます高まることを提起するものである. UA では, 企業全体にまたがる情報システムを統括する役割を持つメインフレームを「インフォメーション・ハブ」と呼び, その持つべき属性を明確にした.

1) インフォメーション・ハブ……インフォメーション・ネットワークの中心となり, 大規模サーバとして全体レベルのシステムの要ともなる. 基幹業務に必要な処理能力を保持し, 各部門業務間で必要となる横断的なメインフレーム機能の役割を担う. インフォメーション・ハブが備えるべき要件として, 次の八つの特性を規定する.

① 大規模トランザクション処理
② 無停止連続運転
③ 先進的なデータベース・システム
④ 限りないシステムの成長性
⑤ オープンな相互接続と相互運用性
⑥ 高生産性アプリケーション開発と実行環境
⑦ システムの無人運転/統合管理

⑧ 高度なセキュリティ

2) サーバ……部門業務を可能にするローカルエリアネットワーク（LAN）を母体とした情報システムのサーバとなる．エンドユーザが共用する部門単位でのデータベースやアプリケーションに必要な機能を提供するとともに，他部門サーバやインフォメーション・ハブとのコミュニケーションを行う．

3) ワークステーション……エンドユーザに対し，情報処理機能を提供する．サーバやインフォメーション・ハブへのアクセスを可能とするが，どのサーバをアクセスしているのか等のインフォメーション・ネットワーク全体の構成を意識する必要はない．

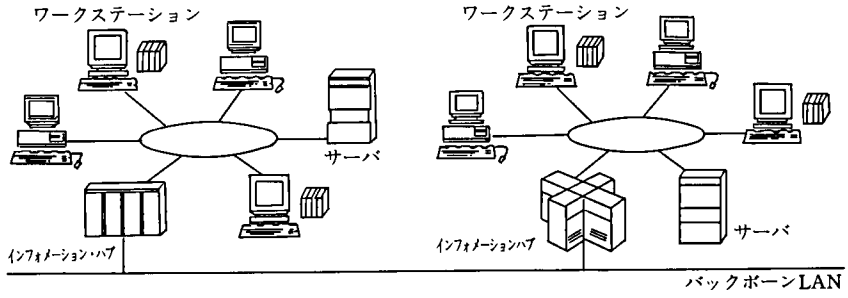


図 2 インフォメーション・ネットワークを構成する3種類のプラットフォーム
Fig. 2 UA-3 tiers platform in information network

2.3.2 五つのサービス

UAでは，情報処理システムの構築および運用環境に必要な機能を明確にし，五つのサービス（機能群）に体系化を行った（図3）．また，このサービスを各プラットフォ

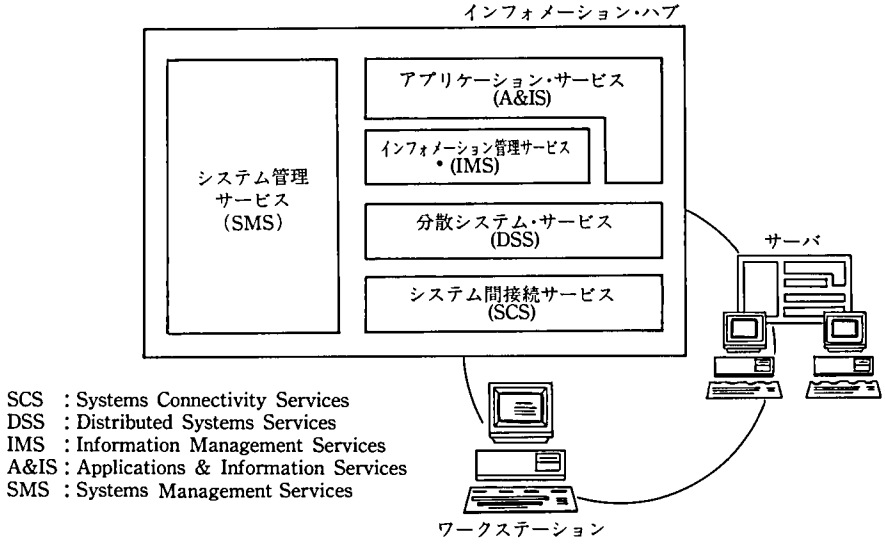


図 3 各プラットフォーム上で展開する五つのサービス
Fig. 3 UA-5 services on each platform

ームに共通に提供することでインフォメーション・ネットワーク全体の相互運用性，開発環境の柔軟性，アプリケーション資産の移植性を高めている．

1)　システム間接続サービス（SCS)……インフォメーション・ハブ，サーバ，ワークステーションの各プラットフォーム間の接続を行う．異機種システムを含むマルチベンダ環境でのインフォメーション・ネットワークの構成を可能とする．装置および各プラットフォーム間との接続に関する各種通信規約を整理統合し，ユニシスが提供する機能を規定することで他ネットワークとのインタフェースについても柔軟な対応が可能となる（図4)．

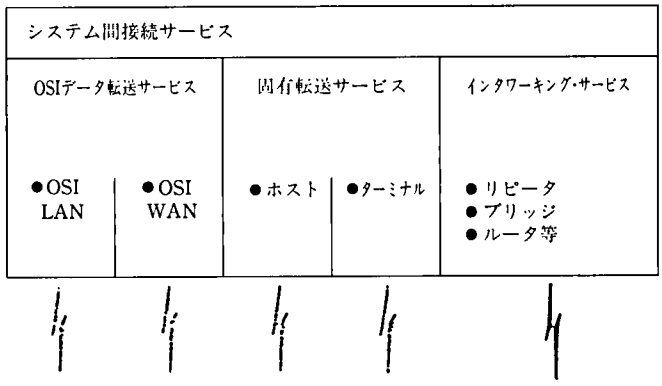


図 4　システム間接続サービスの機能

Fig. 4　UA-Systems connectivity services

2)　分散システム・サービス（DSS)……インフォメーション・ネットワークを構成する各プラットフォームに分散する資源の共用を可能とする．リモート・ファイル・アクセス機能，リモート・プリント機能，リモート・プログラム実行機能，プロセス間通信機能，ステーション間メール機能等の分散資源処理機能を規定し，アプリケーション・レベルでの相互運用性を実現する（図5)．

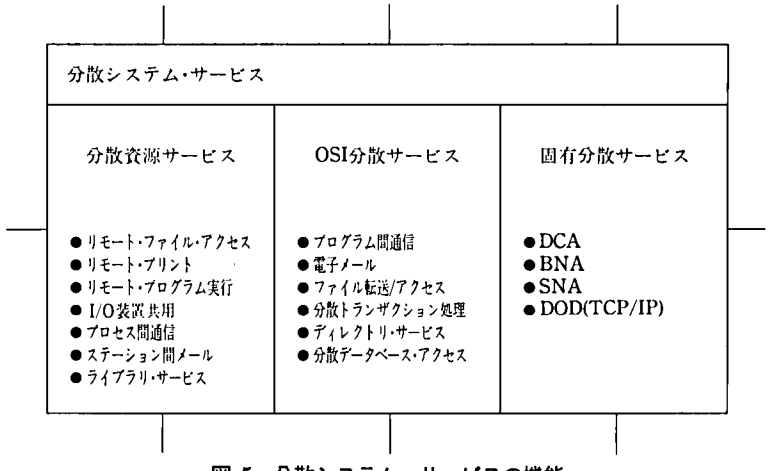


図 5　分散システム・サービスの機能

Fig. 5　UA-Distributed systems services

SCS および DSS が，異機種システムも包含した形でのネットワーキング分野の要件に対応する機能を提供していくことになる（図6）．

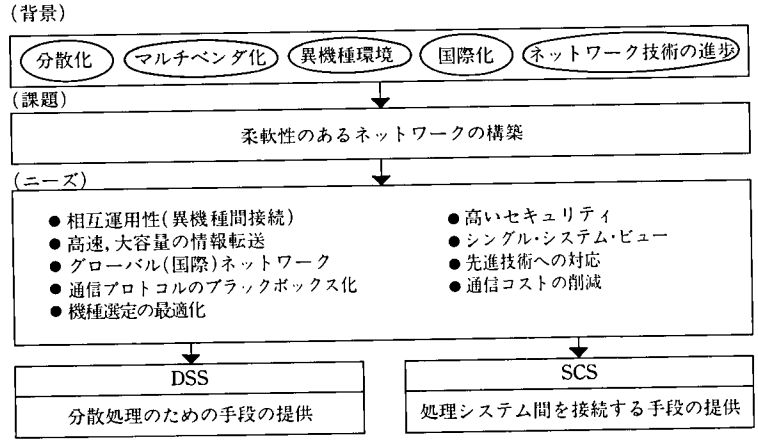


図 6　ネットワーキング分野をとりまく背景とニーズ

Fig. 6　UA-Conceptual requirements in networking area

SCS で規定する機能範囲は，OSI 基本参照モデルでは下位層（トランスポート層，ネットワーク層，データリンク層，物理層）に対応している．

一方，DSS が OSI の上位層（応用層，プレゼンテーション層，セション層）および CCITT (International Telegraph and Telephone Consultative Committee) で標準制定されるプログラム間通信機能，電子メール機能，ファイル転送/アクセス機能，分散トランザクション機能，分散データベース・アクセス機能等を実現する．

3)　インフォメーション管理サービス (IMS)……企業情報システムの財産となるデータベースとリポジトリに関する機能を含む（3章で改めて述べる）．

IMS は，次に述べるアプリケーション・サービスと連携してユーザ・アプリケーション分野における要件に対応する機能を提供していくことになる（図7）．

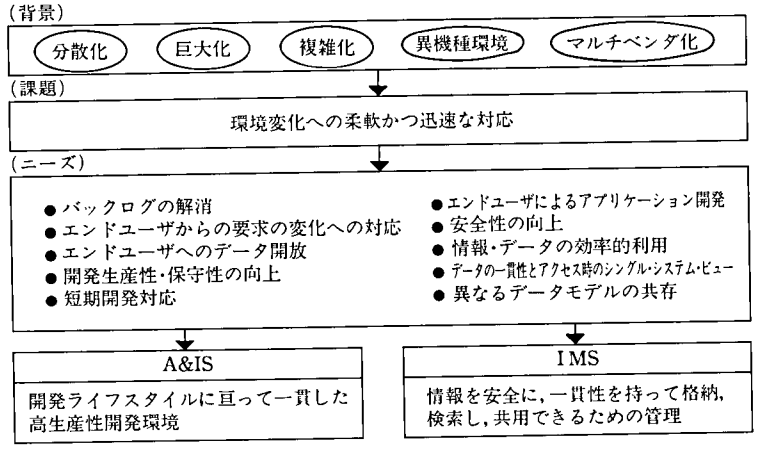


図 7　アプリケーション分野をとりまく背景とニーズ

Fig. 7　UA-Conceptual requirements in application area

4) アプリケーション・サービス (A&IS)……企業情報システムを開発・実行するための共通の操作環境，プログラミング・インタフェースを可能にし，システム開発ライフサイクルに一貫した開発方法論の支援と開発支援ツールの提供を行う(図8)．エンドユーザ，プログラマ，システム設計者等のインフォメーション・ネットワークの利用者に対する基本的なインタフェースを規定しており，以下のサービスに分類できる．

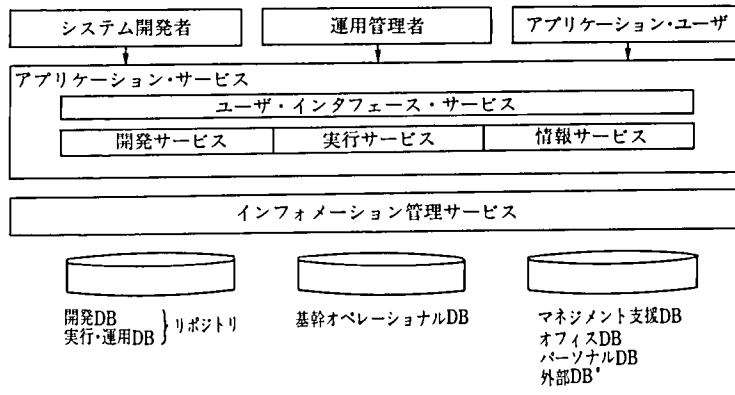


図 8　アプリケーション・サービスの全体概念図
Fig. 8　UA-A&IS service model

① ユーザ・インタフェース・サービス

画面イメージの統一，キーボードやマウス入力の共通化によるエンドユーザとの対話性の向上およびユーザ・インタフェースの規則を述べるスタイル・ガイドを含む（図9）．

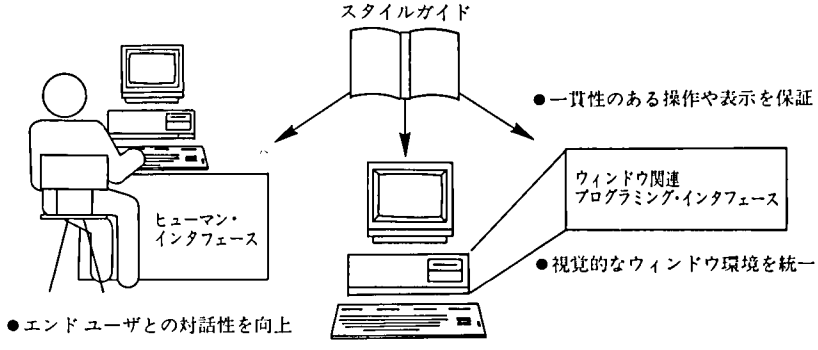


図 9　A&IS ユーザ・インタフェース・サービスの機能
Fig. 9　A&IS-End user interface services

② 開発サービス

企業分析，システム分析，設計，プログラミング，テストおよび保守へと流れるシステム開発ライフサイクルの各工程に対して，統合された開発方法論の支援とユニシス固有の CASE/4GL および国際規格化された高水準プログラミ

ング言語仕様を含む（図 10）.

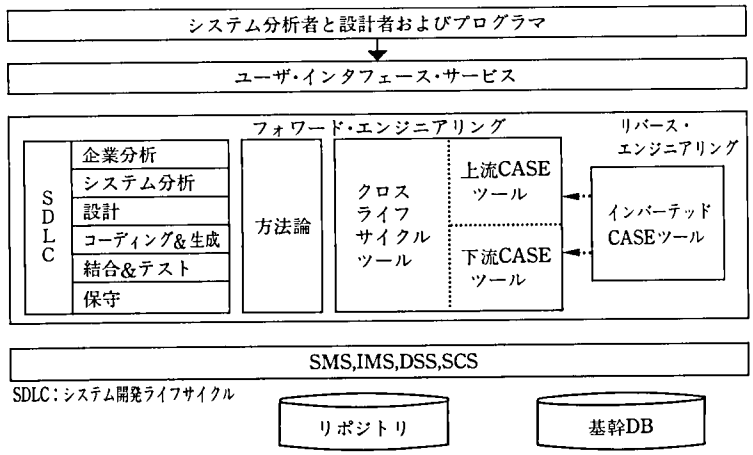


図 10　A&IS 開発サービスの機能

Fig. 10　A&IS-Application programming environment services

③　実行サービス

アプリケーション・システムの実行を支援するための共通のインタフェースを含む．つまり，UA の他のサービス（SMS，IMS，DSS）の機能を利用するためのプログラミング・インタフェースおよびエンドユーザ・インタフェースを提供する（図 11）.

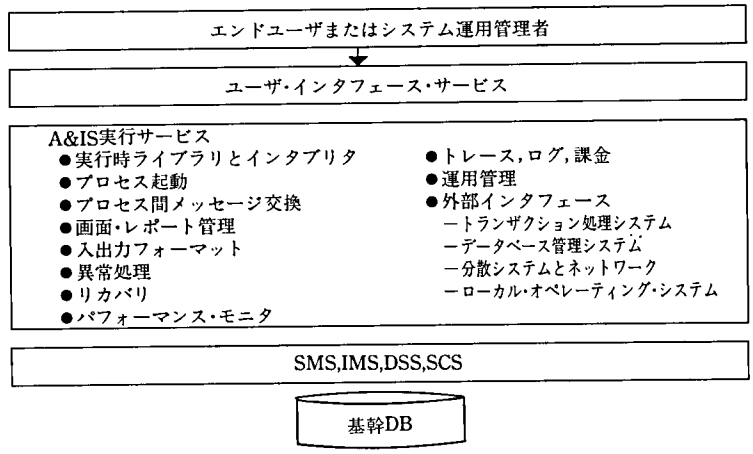


図 11　A&IS 実行サービスの機能

Fig. 11　A&IS-Application execution environment services

④　情報サービス

情報検索，知識ベース，音声応答等の企業内の基幹情報システムと連携して，エンドユーザ・コンピューティングに関わる機能を含む（図 12）.

5)　システム管理サービス (SMS)……インフォメーション・ネットワーク資源全体の一元的な運用，維持，管理を可能とする．インフォメーション・ネットワーク

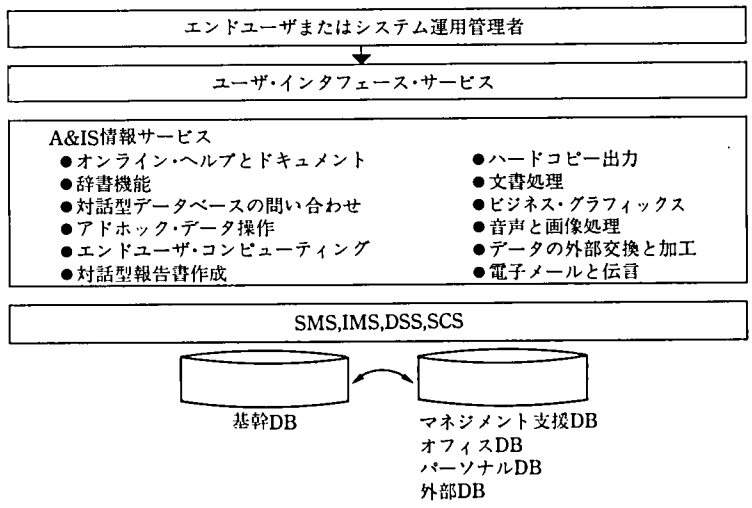


図 12　A&IS 情報サービスの機能

Fig. 12　A&IS-Application information environment services

構成の導入・変更管理，障害管理，キャパシティ管理，セキュリティ管理等を規定し，UA の他の四つのサービス (SCS，DSS，A&IS，IMS) で規定される機能群を統合的に管理する機能を実現する (図 13)．

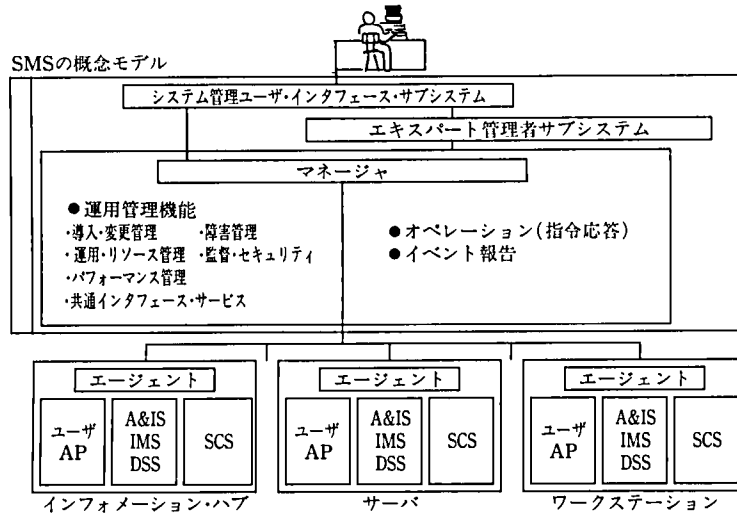


図 13　システム管理サービスの機能

Fig. 13　UA-Systems management services

2.3.3　三つのサービス・クラス

UA で規定される各サービスは，オープン・システムを目標とする標準化の動きを全面的に採用することで，インフォメーション・ネットワークの要件を実現しようとしている．しかし，各プラットフォーム固有の性能を適切に活用するための機能や長年蓄積されてきた情報資源を有効に活かすための機能も重要なものと認識している．

インフォメーション・ネットワーク上での適用目的によって，採用基準となる仕様

が異なっており，三つのクラスに分類される．

1) オープン・クラス……X/Open を基準としたオープン・システムを実現可能とするためのサービスを含む．インフォメーション・ネットワーク全体を最大限に活用するための重要な基本機能を満たす．

2) プレミアム・クラス……各プラットフォームで共通に規定されるユニシス固有のサービスおよび既存ユーザ資産の継承のためのサービスを含む．
各プラットフォームの性能および特長が効率よく活かされる．

3) 共存クラス……オープン・システムで規定されないが，他社システムとの相互接続性を可能にするためのサービスを含む．マルチ・ベンダ環境でのインフォメーション・ネットワークの構築上の柔軟性を高める．

以上の三つのクラスは，各プラットフォーム共通に規定される．

ユーザの条件あるいは目的に応じて，選択したり，共存させることで，最適なアプリケーションの開発・実行環境を構築することができる（図 14）．

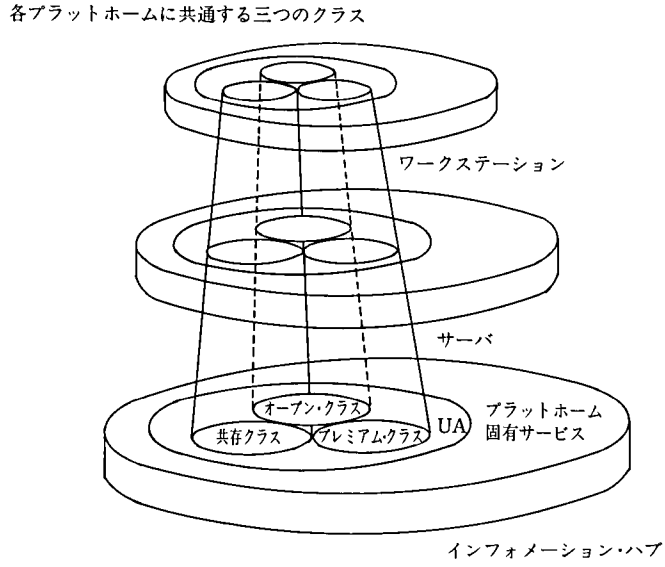


図 14 各プラットフォームに共通する三つのサービス・クラス

Fig. 14 UA-3 classes of UA services

3. インフォメーション管理サービス（IMS）

IMS は，企業情報システムで取り扱われるあらゆる情報の管理に関する機能とインタフェースを明確にする．ここでの情報の管理とは，企業の財産とも言えるデータベースそのものの運用のための手段を意味することは当然のことであり，企業情報システムでの資源にまつわる管理機能やシステム開発ライフサイクルの各工程で発生する情報の管理機能も含んでいる．後者は，リポジトリに関する機能で規定され，異なる機種で貯えられた情報とのインタフェースも考慮される．

IMS では，機能とインタフェースを実現すべきデータモデルについての指針も示す．提示される各データモデルは，適切な三つのプラットフォーム（インフォメーシ

ョン・ハブ，サーバ，ワークステーション）上で実現され，相互運用性/共通性/統合化/透過性等のインフォメーション・ネットワークの要件を満たしていく．

3.1 IMSの概念モデル

UAにおけるIMSの概念を図15のサービス・モデルで示す．IMSで規定された機能は，次の四つのサービスに分けることができる．

1) データベース管理に関するサービス
2) 分散データベース管理に関するサービス
3) データモデルに関するサービス
4) リポジトリに関するサービス

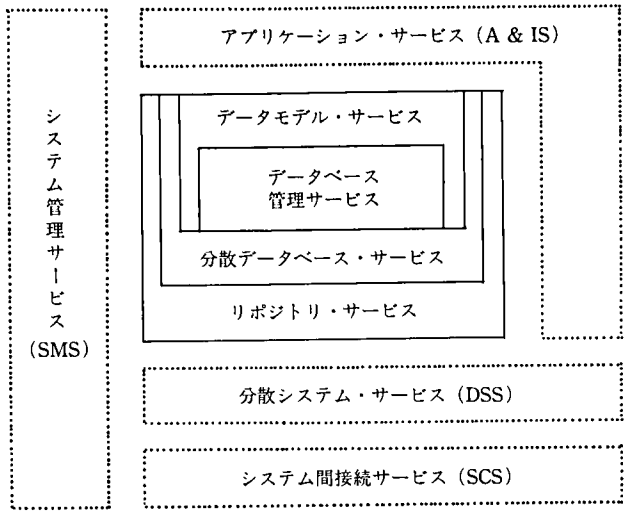


図 15　IMSのサービス・モデル

Fig. 15　UA-IMS service model

3.2 データベース管理サービス

データベース管理システム（DBMS）の基本的要件を持つ．実現されるデータモデルがどんなものであれ，以下に示す機能が具備される．

① データの永続性　：データの格納，保持等の管理機能
② データの定義と操作：データの定義とデータの構造の定義および操作のための手法（設計/実装/操作）
③ 同時実行制御　：データを操作する上で必要な資源のロッキング手法，キューイングとデッドロック検知・解消方法
④ データの独立性　：物理的独立性，論理的独立性，
⑤ データの一貫性　：物理的な整合性，参照整合性
　データの型/領域に関する整合性制約
⑥ データの保全　：データの保全と復元の手法
⑦ 最適化　：物理的格納場所の最適化手法，
　データアクセス処理の最適化手法
⑧ 機密保護　：アクセス制御とユーザの認証

⑨ アドミニストレーション*：アカウンティング，モニタリング

データベース管理に関するサービスは，データベースという用語/概念が使用され始めた60年代の後半から[2]，20年以上にわたって培われてきたデータベース・システム共通の基本的機能とも言える．

3.3 分散データベース管理サービス

分散の対象として，以下のものがある**．

1) データの分散……分散データベース形態
2) アクセスの分散……クライアント/サーバ形態
3) トランザクションの分散……OLTP(OnLine Transaction Processing)モデル
4) アドミニストレーションの分散

各々の分散の対象は，分散データベース管理の実現時のシステム形態および接続形態の中で表現される．

3.3.1 分散のシステム形態

分散データベース管理サービスの実現のためのシステム形態には，クライアント/サーバ（分散アクセス）形態と，分散データベース形態がある．

前者は，異なるプラットフォーム間でデータベースのアクセス要求の処理プロセスを分担するものである．基幹業務等，大容量データベースの場合，インフォメーション・ハブがサーバの役割を担う．クライアント/サーバ形態の標準モデルにはOSIで提案されている遠隔データベースアクセス（RDAモデル）がある（図16）．

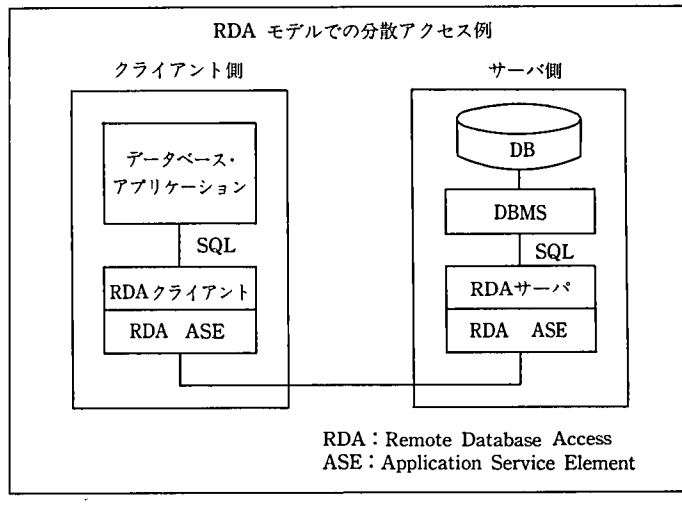


図 16 OSI RDA モデルでのクライアント/サーバ形態

Fig. 16 Client/server model (OSI RDA)

後者の分散データベース形態は，データベースそのものが複数のプラットフォームに分割配置されており，局所性を生かしたアクセスが可能となる．この形態の標準モデルには，X/Openで提唱されているOLTPモデルがある(図17)．UAではいずれの

* アカウンティングやモニタリングによるシステム内部情報はSMSの機能で規定される．
** 分散処理におけるトランザクションの定義，2相コミット・プロトコル，ネーミング機能等はDSSの機能で規定される．

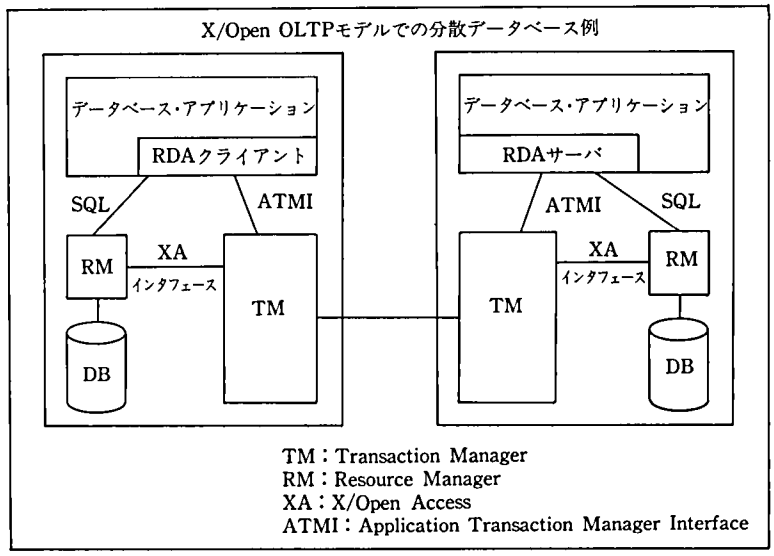


図 17　X/Open OLTP モデルでの分散データベース形態

Fig. 17　Distributed database model (X/Open OLTP)

形態もオープンクラスの形態として規定しており，各プラットフォームに提供される．

なお，ユニシス固有のサービスで，大規模トランザクション処理と無停止連続運転機能（フォルト・トレラント機能）を可能とする拡張トランザクション処理アーキテクチャ（XTPA : eXtended Transaction Processing Architecture）が，インフォメーション・ハブ（UNISYS 2200 シリーズ）で実現している[4]．これは負荷分散の例であるが，アクセス分散の一つの形態ともいえる．

3.3.2　分散の接続形態

分散データベース管理システム（DBMS）がどのプラットフォームに存在するか，またデータモデルの違いも含めてどんな DBMS を対象にしているかにより，いくつかの接続形態が考えられる．

1)　同機種/同種 DBMS……同じプラットフォームで同じデータモデルを分散形態で運用することはあまり考えられず，本来の DBMS の目的でもある集中型の運用にすべきであろう．しかし，異機種分散処理において，互いが補完関係にある場合(たとえば，ホットスタンバイ運用)，この形態の方が運用上都合がよいとも言える．

2)　異機種/同種 DBMS……サービスとプロトコルが標準仕様で規格化されている形態と言える．分散形態として最も一般的である．もちろん，OSI の基本参照モデルで言えばより下位レベルでの接続性も保証されなければならない (DSS および SCS で，どの標準規格を適用していくか規定される)．IMS では，OSI の RDA (SQL Specialization : ISO/IEC DIS 9579-2) をオープンな標準規格候補としている．

3)　同機種/異種 DBMS……同一プラットフォーム内の異なる DBMS 間での分散処理は，システムの移行等の特殊な目的しか考えにくい．データモデルが異なる

場合，DBMS 間のプロトコルを定める必要もあり，定常的に運用する形態としては効率的に無駄が出てくるのではないか．

異なるデータモデルを並行してアクセスすることを満たすためであれば，同一の DBMS で複数のデータモデルへのアクセスを可能とするほうが効果的であろう．高水準プログラミング言語(3 GL)の世界では，その性格上，アプリケーション・プログラミング・インタフェース（API）の共通化の面で統一性が採れにくいが，UNISYS 2200 シリーズ上の大規模データベース・システム UDS (Universal Data System) では，ネットワークモデルもリレーショナルモデルも同一のアプリケーション・プログラムで並行してアクセスすることが可能である．

4) 異機種/異種 DBMS……プラットフォームも異なり，DBMS も異なる形態が，ユーザにとって望ましい分散データベースなのか，効果度合いも含めて疑問があるが，すでに存在するデータベース・システムを相互にアクセスしたいという要求は当然，発生しうる形態である．また，現行システムを稼働させながら，別モデルの新システムへの移行を実施する場合も考えると，むしろ重要な形態と言える．この場合，既存データベース・システムへの影響を最小限にすることが大切であり，接続のプロトコルを定め，双方もしくは一方の DBMS にゲートウェイ・ソフトウェアを導入するのが一般的である．UNIX*シリーズで広く使われている ORACLE**アプリケーションからインフォメーション・ハブ上の基幹データベースへのアクセスを可能にする形態は，すでに実現している具体例である（図 18）．

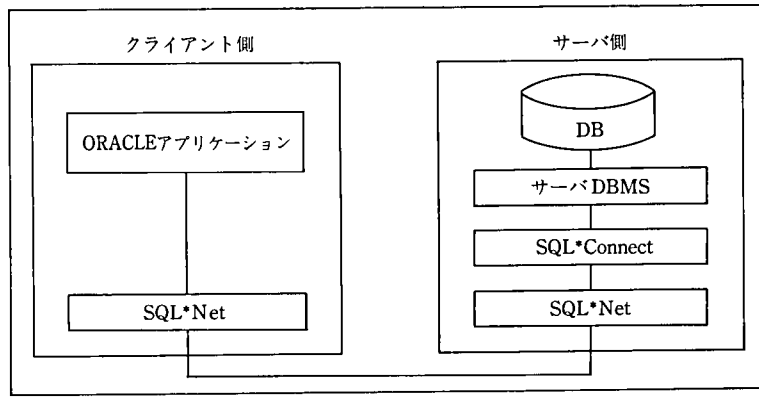


図 18 特定データベースによる異種 DBMS 分散の例

Fig. 18 Example of client/server model (heterogeneous DBMS)

3.4 データモデル・サービス

一般的にデータモデルと言った場合は，実世界の表現の手段の優劣を議論することが多い．つまり，管理すべきデータおよびデータの構造の定義方法と管理しているデータの操作方法の有益性や，理論的背景を持った完備さ等での比較が行われている[5]．

本稿では，IMS で取り上げるデータモデル間の比較は行わない．最初に述べたように，インフォメーション・ネットワークを構築していく上での要件を満たす適切なデ

* UNIX オペレーティング・システムは，UNIX System Laboratories, Inc. が開発し，ライセンスしている．

** ORACLE は，米国 ORACLE 社の登録商標である．

ータモデルを選ぶ環境を提供することが大切と考えるからである.

IMS では，オブジェクト指向モデル，リレーショナルモデル，ネットワークモデル，および ISAM 等のデータモデルを規定している.

その他，既存のデータ辞書システムに実質的に使われている ER モデル（Entity Relationship Model）やインフォメーション・ハブ（UNISYS A シリーズ）で実現されている意味データモデル SDM（Semantic Data Model）についても方向性を示している.

3.4.1 オブジェクト指向モデル

モデルと呼ぶには，一定した理論的な裏づけと共通の仕様が定まっていないと言うべきかもしれない．しかし，オブジェクト指向データベースとしての機能要件は共通認識となっており[6]，ユニシスも ANSI や OMG（Object Management Group）での共通仕様の作成活動に積極的に参加している[7]~[9].

オブジェクト指向モデルでは，必須機能である複合オブジェクトの定義，カプセル化，ポリモルフィズム等の実現により，拡張性のあるデータ表現，プログラムの再利用の促進等，開発の生産性向上への効果が見込まれている．実際，ユニシスではオブジェクト指向モデルを実装した実験システムでのテストを開始し，成果をあげている.

オブジェクト指向モデルであれ，データベース管理システムとしての機能要件は，リレーショナルモデルやネットワークモデルと同等である．このため，インフォメーション・ハブでのプラットフォームでは，既存データベース管理システムとの整合性を検討中であり，オブジェクト指向モデルも含めた異種モデル間の並行アクセスの可能性を追求している.

現在，UNISYS A シリーズで提供されている SIM（Semantic Information Manager）は，意味データモデル SDM を実装したものであり，オブジェクト指向モデルの機能要件を包含している[10]．さらに，ユニシスでは SIM 固有の機能を活かしつつ，オブジェクト指向モデルへ成長させる作業が行われている（図 19）.

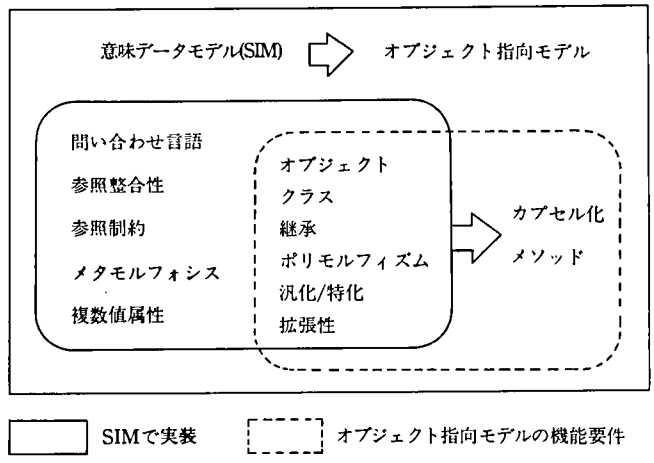


図 19　SIM とオブジェクト指向モデルの機能差

Fig. 19　SIM vs Object Oriented Model

3.4.2 リレーショナルモデル

1987年のSQL国際標準規格の制定 (ISO 9075) およびJIS制定 (JIS X 3005-1987) 後も拡張作業が精力的に行われており，ユニシスも積極的に取り組んでいくことを宣言している．

リレーショナルモデルが，今後ますます基幹業務に浸透していくことは間違いなく，インフォメーション・ネットワークでの相互運用性等のキーとなるデータベースとして拡張提供される．リレーショナルモデルのDBMSは，各プラットフォームで単一的に提供されるものとは限らない．UNIXを中心としたクライアント/サーバ形態でのオープン化の促進のためには，たとえばORACLE等のデータベース管理システムを取り込んでいく方向である（図18)．

3.4.3 ネットワークモデル

具体的な製品として1970年当初より提供され，大規模トランザクション処理等の基幹業務で幅広く使われている．このモデルでの標準化は，ほぼ固まっておりサーバ，ワークステーションも含めたオープン・クラスの機能としてよりは，既存ユーザの財産の継承という意味で重要である．

今後は，インフォメーション・ハブが備えていく各種属性（2.3.1項八つの特性）への対応での機能拡張が継続される．たとえば，大規模トランザクションや無停止連続運転に対応するUNISYS 2200シリーズでのXTPA対応は，ネットワークモデルを含めたすべてのモデルに対して行われていく．

3.4.4 ISAM

データモデルの位置づけではあるが，機能的にはデータベース管理システムとは言いがたい．データの定義と操作，またデータの永続性についての機能は満たされるが，データの一貫性，保全，機密保護等については規定されない．

しかしながら，X/OpenのXPGで公開されており，サーバ規模でのオープン性を維持する重要なモデルとの認識である．

3.5 リポジトリ・サービス

UAのA & ISでは，CASE/4GLをアプリケーション開発支援ツールの柱に規定しており，現行のデータ辞書/ディレクトリ・システム (DD/DS) では，管理されていない各種情報が重要なものとなってきた．このような情報の管理は，リポジトリ・サービスの対象となる．

IMSでは，システム開発ライフサイクル (SDLC) の各工程での共通の情報管理にリポジトリを位置づけている(図20)．SDLCの各工程で，どのような情報が，どんな構造でリポジトリに貯えられ，活用できるのか，すべてが明確にされているわけではないが，各工程で共通となるリポジトリの機能を以下にまとめる．

① 拡張機能：情報モデルの拡張を可能とする．自己記述型の情報モデルとなる．
② 名前管理：名前付け，別名，オブジェクトの一意付け等．
③ バージョン管理：リポジトリ内のオブジェクトとクラスの変更記録や複数の改訂版の保持等．
④ コンフィグレーション管理：一つの括りで定義され，制御されるオブジェクトの集まりを可能にする．

⑤ 分散機能：複数のプラットフォームにまたがったリポジトリの分散処理機能．分散データベース管理サービスと同等の機能を持ち，クライアント/サーバ形態が基本となる．

⑥ ブリッジとゲートウェイ：別種のリポジトリとのコミュニケーションで必要となるブリッジあるいはゲートウェイの支援機能も含む．

⑦ APIおよびエンドユーザ・インタフェース：アプリケーション・プログラムやツールからリポジトリをアクセスするためのインタフェース．

⑧ 問い合わせ/レポーティング：リポジトリ内の情報の簡便な問い合わせ，およびレポーティング機能．

⑨ ワークフロー：SDLCの各段階でのオブジェクトの状況等（ステータス管理）

⑩ 相互運用性：インフォメーション・ネットワーク内の他のリポジトリとのコミュニケーション．情報の大量転送のためのIMPORT/EXPORT機能等．

これに加え，データベース管理システムとしての要件(3.2節)が，実装されるデータモデルにより具備されることになる．

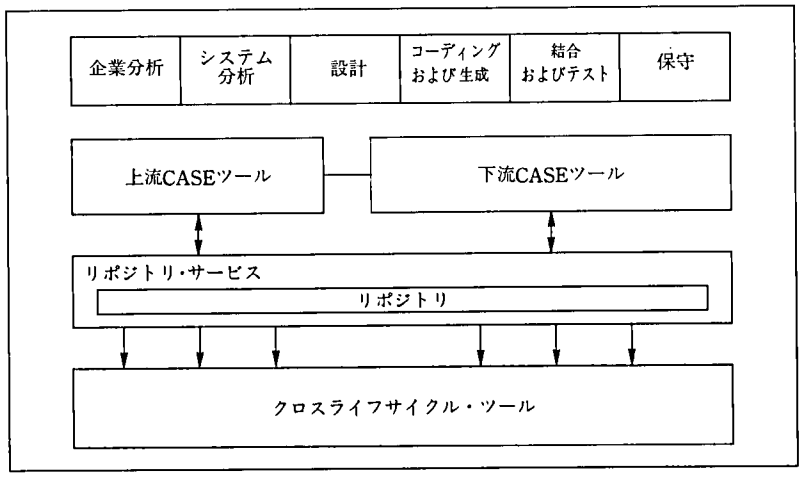


図 20　システム開発ライフサイクルでのリポジトリの位置づけ

Fig. 20　SDLC tool integration using repository

現在，ユニシスではリポジトリのための情報モデル（データモデル）の評価と実験システムでのテストを進めており，それらの成果を踏まえて，リポジトリの情報モデルが明示されることが期待される．この時の情報モデルの要件としては，以下の項目があげられる．

1) 統合化……数多くかつ独立に開発されたツールやサービスからのデータ（リポジトリ情報）のアクセスが可能となること．

2) 既存財産の維持……現存のCASEツールやDD/DSのデータからの移行パスを提供すること．

3) 拡張性/将来性……産業界および企業内の進展に伴って，使われていく新しい技

術に対応可能なこと.

4) 共存性/標準化……実質的な業界標準もしくは国際標準に準拠あるいは互換性を保つこと.

これらの要件を満たすためには，情報モデルにオブジェクト指向モデルを採用することが有利と見ている．たとえば，要件の1)や2)で必要とされるモデルの融通性は，カプセル化を行うことで容易に発揮される．また，3GLからCASE/4GLを含めた共通の情報モデルを，現行のDD/DSで採用しているERモデルやリレーショナルモデルでなく，オブジェクト指向モデルの機能で表現することは魅力的でもある．

4. おわりに

UAは，オープン・システムの指向により，異機種間の相互運用性の向上，ユーザ・インタフェースの共通化，開発生産性の向上等を主なねらいとする今後の企業情報システム基盤のあり方を明確にした．この中でIMSの役割は，最も効果的なデータベース管理とインフォメーション・ネットワークの情報の中核となりうる統合化されたリポジトリ機能を提供することに集約される．新しい機能の提供は，ややもすると既存アプリケーション資産の陳腐化を招く恐れがあった．UAでは，各プラットフォームの固有の技術を活かしつつ，インフォメーション・ネットワークへの組み込みを可能にすることで，新しい情報処理環境での既存資産の継承を明確に打ち出している．

これは，たとえばUNISYS 2200/1100シリーズでのプログラムおよびデータベースの互換性を過去20年に亘って維持し，かつ新しい基盤ソフトウェアへの円滑な移行の実績を残してきた事実を踏まえたユニシスの自信を示している．

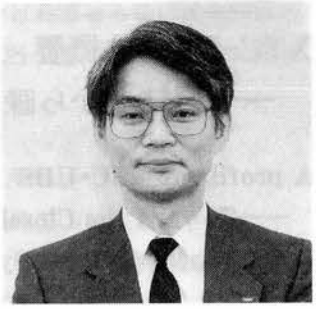
UAで規定される機能は，90年代の柱となるべく，現提供プロダクトで実現されつつあり，今後も新規プロダクトを含め，順次提供される予定である．

参考文献

[1] Unisys Architecture, 080101004-0, 日本ユニシス, 1990.
[2] 植村俊亮, データシステムの基礎, オーム社, 1979.
[3] 佐藤正美, リポジトリ入門解説, ソフト・リサーチ・センター, 1991.
[4] 樫山汎, 拡張トランザクション処理アーキテクチャ, UNISYS技報, Vol. 9, No. 4, 1990.
[5] 宇田川佳久他, 特集高水準データモデルの最近の研究動向, 情報処理, Vol. 32, No. 9, 1991.
[6] M. Atkinson, et al, The Object-Oriented Database Manifesto, Proc. of Deductive and Object-Oriented Database, 1989.
[7] J. P. Thompson, & R. Bigelow, OSMOS-The Object Model, OMG Object Model RFI, 1991.
[8] OMG-Object Model Task Force, The OMG Object Model, draft 0.9, 1991.
[9] ANSI/ASC/X3/SPARC, DBSSG/OODBTG Final Report, 1991.
[10] 山口裕久, InfoExec—SIMを中心として—, UNISYS技報, Vol. 8, No. 2, 1988.

執筆者紹介　藤　島　忠　篤 (Tadaatsu Fujishima)

昭和44年九州大学理学部数学科卒業. 同年, 日本ユニシス(株)に入社. 主としてUNISYS 2200/1100シリーズのデータベース管理システムの開発, 保守業務に従事. 現在, システムプロダクト本部 ソフトウェア三部に所属. 情報処理学会会員.



XTC-UDSの概要
——密結合から疎結合，そして睦結合へ

A profile of XTC-UDS
——Toward the Closely-Coupled System from the Tightly-Coupled or Loosely-Coupled System

阪口喜好

要約 処理装置を並列に動作させることにより，処理能力，信頼性，可用性，および拡張性の点で優れたシステムを実現できる．XTPA（拡張トランザクション処理アーキテクチャ）では，ディスクとRLP（ロック制御専用プロセッサ）を共用する計算機システムをチャネル結合した多重プロセッサ構成をクロズリ・カプルド・システム（Closely coupled system）と呼ぶ．この多重プロセッサ構成の下で，汎用データベースの共用が可能となった．この実現技術が，XTC-UDSである．

本稿は，まずプロセッサ結合方式の比較を行い，クロズリ・カプルド・システムの特徴を明らかにする．次に，データベース共用にあたって必要とされるデータベース技術の課題について述べる．最後に，それらの課題が，XTC-UDSにおいて，どのように解決されたか説明する．

Abstract The parallel processing system provides a great advantage to the implementation of a system advanced in processing capacity, reliability and availability as well as expandability. The eXtended Transaction Processing Architecture (XTPA) defines a closely-coupled system as a multi-processor configuration which channel-links computing systems each sharing disks and a record lock processor (RLP). This multi-processor environment has made it possible to share a general-purpose database with the use of XTC-UDS.

This paper, in the first place, clarifies the advantages of a closely-coupled system, by comparing the methods of processor coupling, followed by a discussion of several difficulties regarding database technology which is considered necessary for database sharing, and finally explains how solutions have been given to those problems in XTC-UDS.

1. はじめに

基幹システムの基盤技術として求められているものは，処理能力，柔軟性，および信頼性である．しかし，大型汎用計算機の処理能力の伸びは，ユーザ・ニーズの拡大に追従することができていない．処理装置を並列に動作させることにより，システムの処理能力，信頼性，可用性，および拡張性の点で優れた高性能計算機システムを実現する技術は，プロセッサ結合構成（Processor coupling）と呼ばれている．この技術の利点は優れた価格性能比にある．プロセッサの価格はプロセッサのスピードに比例する．単一の速いプロセッサ（高価）の処理能力（MIPS）を，遅いプロセッサ（安価）の結合構成による合成処理能力（MIPS）で凌駕できるからである．

本稿では，まず2章でプロセッサ結合構成について述べる．次に，プロセッサ結合

構成の下でデータベースの共用を実現するために解決しなければならない技術的課題を 3 章で説明する．4 章では，これらの課題が，XTC-UDS においてどのように解決されているかを解説する．

2. プロセッサ結合構成

複数のプロセッサを結合する方式は，非共用型と共用型の二つに大別できる．

非共用型では，各プロセッサは個別の主記憶装置とディスク装置を持ち，プロセッサ間は高速バスもしくは通信回線で結ばれる（図 1）．アプリケーションは，一つまたは複数のプロセッサ上で順次実行され，バスを介して命令を送ることによりデータをアクセスする．アプリケーションがプロセッサ間でうまく分散されて，データへのアクセスにおいて集中点（Hot spot）が生じなければ，高処理能力，高信頼性，および拡張性をこの方式で実現することができる．バッファやメモリが分断されることや入出力要求のために他のプロセッサへ送信する負荷等により，スループットや応答時間が低下する．

共用型は，共用する記憶階層の構成要素により，密結合システム（Tightly Coupled System）と疎結合システム（Loosely Coupled System）に分類される（図 2 および図 3）．

密結合システムでは，プロセッサは主記憶装置を共用し，一つのオペレーティング・システムが計算機システム全体を管理する．プロセッサの数を増やしても，オペレー

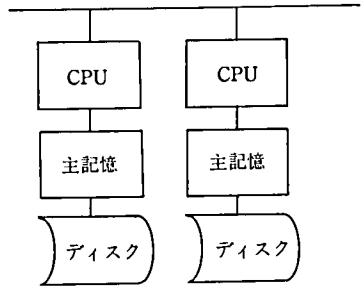


図 1　非共用型プロセッサ結合構成
Fig. 1　Non-shared coupled processor

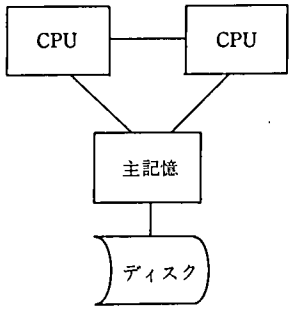


図 2　密結合プロセッサ構成
Fig. 2　Tightly coupled processor

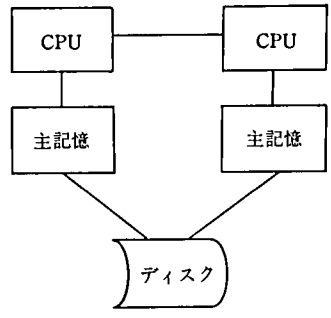


図 3　疎結合プロセッサ構成
Fig. 3　Loosely coupled processor

ティング・システムが一つなので，システムの運用を変更することなく処理能力を上げることができる．この方式は，マルチ・プロセッサ・システムとして，広く使われている．しかし，多重度が上がれば上がるほど，n 台のプロセッサを内蔵しても n 倍の性能に達しないという問題が顕在化してくる．その要因としては，資源競合による待ちの発生，大規模システム効果（Large system effect）によるオーバヘッド，プロセッサ間通信回数の増加，主記憶共用によるプロセッサ間の競合等があげられる．このような性能低下要因の影響を低減する改良が図られてきているが，密結合できるプロセッサの台数は現在のところ 8 台くらいが実用上の限度といわれている．より大規模なシステムに柔軟に対応する方式として，次に述べる疎結合システムが注目を集めている．

疎結合システムでは，各プロセッサに固有の主記憶装置（ローカル記憶）とオペレーティング・システムが付随し，ディスク装置を共用する．密結合に比べ，多重度が上がっても性能低下は著しくなく，性能はプロセッサの数と比例関係を維持するという特徴がある．反面，アルゴリズム上の複雑さ，システム管理上単一システムと同様の運転や運用を実現できること，障害発生時のためのバックアップ，処理負荷の分散が焦点となる．疎結合システムの性能は，ファイル共用のための排他制御の効率とアクセス効率に左右される．これまで疎結合システムの排他制御は，ディスク装置にロック機構を付加する方式やチャネル間結合装置を介してソフトウェアによりロック制御を行う方式等により実現されてきた．しかし，これらの方式では，大規模オンライン・トランザクション処理システムを構築するには負荷が大きすぎるし，ロック管理が単純すぎて汎用データベース管理には不十分であった．

疎結合システムの下で汎用データベースの共用を実現するために，弊社ではロック制御専用プロセッサ RLP（Record Lock Processor）を開発し，RLP とディスク装置を共用するプロセッサ結合方式をクロズリ・カプルド・システム（Closely Coupled System）と名付けた（図 4）．本稿では，この方式を睦結合システムと呼ぶ．RLP を介することによりレコード単位の排他制御が可能であるため，ロックの及ぶ範囲が局所化でき，従来のような広域的なロック（たとえば，ファイル全体に掛かるロック等）によるスループットの低下を回避できる．睦結合システムにより，疎結合システムの

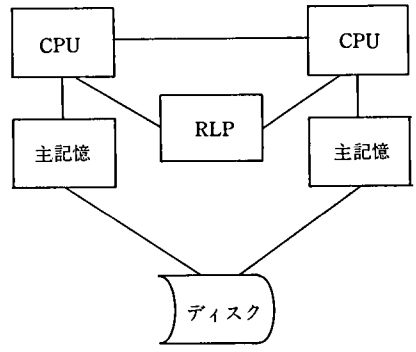


図 4　睦結合プロセッサ構成

Fig. 4　Closely coupled processor

利点を維持しつつ，大規模オンライン・トランザクション処理システムの構築が可能となる．

3. データベース技術の拡張

汎用データベース管理を睦結合システムの下で実現するには，どのようなデータベース技術の拡張が必要となるのであろうか．本章では，これについて述べる．なお，単一プロセッサあるいは複数のプロセッサの密結合により構成される計算機システムを，以下ではホストと呼ぶ．

3.1 大域的ロック管理 (Global Locking)

データベースを共用するためには，排他制御に用いられるロック管理機能が，ホスト内のみならずホスト間でも有効とならなければならない．データベース管理システムがソフトウェア的に実現している論理ロックだけでなく，ハードウェア命令 (Test and Set命令 (TS命令)) を用いた物理ロックも拡張の対象となる．機能的には，施錠 (Locking)，解錠 (Unlocking)，およびデッドロック対策 (検知および解除) が必要である．ロック対象の単位(Granularity: ファイル，ブロック，レコード等)，クラス (Class: 共有ロック，専有ロック等) および継続期間 (Duration) の管理も必要である．

単一ホスト環境では，ホスト障害はすなわちシステムの全面停止につながるが，睦結合システムでは，システム停止には到らない．したがって，ロックを保有するホストに障害が発生しても，リカバリが完了するまでロックは保持されなければならない．また，リカバリ完了時点で，ロックの保有者だけでなく他のホストからロックを解くことができる特権処理が必要となる．論理ロックには，このような考慮が必要となる．逆に，物理ロックは，従来どおりロック保有ホストに障害が発生した時点で解除されることが要請される．

3.2 バッファの首尾一貫性管理 (Buffer Coherency)

データベース管理システムは，入出力の効率を上げるため，主記憶上に入出力バッファ領域を持ち，一度ディスク装置から読み込んだデータは，バッファ上に保持して再利用を図る．単一ホストの場合，データのコピーはバッファ上とディスク装置上に存在するが，コピー間の不整合は更新処理の途中でのみ発生する．したがって，専有ロックを用いて更新処理を排他的に処理することにより，不整合データの参照の危険性は回避できる．

睦結合システムでは，データのコピーは，ディスク上と各ホストのバッファ上に存在する．これらのコピーの整合性を維持するには，前述の大域的ロックのみでは不十分である．なぜならば，更新処理が実行されるホスト (更新ホスト) と参照処理が実行されるホスト (参照ホスト) が異なる場合があるからである．この場合，更新ホストから参照ホストへバッファ上のデータの無効通知 (Invalidation) が必要となる．また，無効通知を受け取ったホストは，ディスク装置より最新のデータを読み込む処理 (Data validation) が必要である．

3.3 ログ・マージ (Log Merging)

データベースの障害回復に用いられる更新履歴データ (ログ) は，アクセスの集中

によるシステム・ボトルネックを避けるため，各ホストで個別に採取される．ロング・リカバリやセレクティブ・リカバリを行うに当たっては，個別に採取された履歴データをマージして，更新の発生順に並べ直す必要がある．このためには，更新の発生順序を示すなんらかの指標を履歴データに付帯することになる．この指標としてタイム・スタンプを利用する場合やリカバリの開始点や終了点を時刻で指示する場合は，ホスト間の時刻調整機能が必要となる．

3.4 局所リカバリ（Local Recovery）

睦結合システムの特徴の一つは，一つのホストで障害が起きても，システム全体の停止につながらないという無停止性と高可用性にある．従来の単一ホストのシステムでは，障害ホストの復旧後ショート・リカバリを実行し，回復処理完了後にトランザクション処理を再開するという手順を踏んできた．リカバリ処理とトランザクション処理は両立しえなかった．高可用性を実現するためには，障害ホストにおける仕掛り処理の回復（データベースの回復とロックの解除）に要する時間を極小化しなければならない．そのためには，障害ホストの復旧を待たずに，稼働中のホストからトランザクション処理と並行して，障害ホストの仕掛り処理の回復が行えるような新しいリカバリ方式（局所リカバリ）の確立が必要となる．

4. XTC-UDS

これまで培われてきたユーザのコンピュータ資産を継承しながら，従来の単一プロセッサ・システム，密結合システム（マルチ・プロセッサ・システム）あるいは疎結合システム（ホット・スタンバイに利用）から睦結合システムへ移行を可能とするためには，

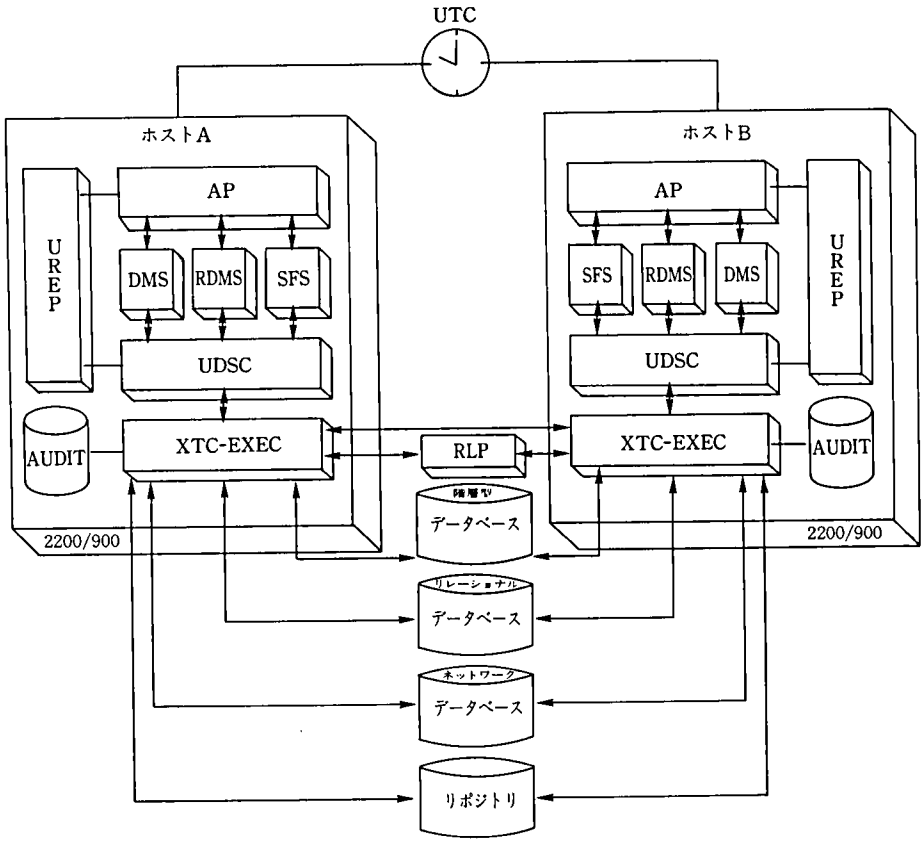
1) コンピュータ・アーキテクチャの互換性
2) オペレーティング・システムの互換性
3) データベース管理システムの互換性
4) アプリケーション・プログラムの互換性
5) データベースの互換性

が必須である．これらの五つの互換性を保ちながら，睦結合システムの下で汎用データベースの共用を可能とする技術体系を XTC-UDS と呼ぶ．XTC は，拡張トランザクション処理能力（eXtended Transaction Capacity）を意味し，UDS は弊社の汎用データベース管理ソフトウェア UDS 1100（Universal Data System 1100）を意味する．図5は，XTC-UDS の概要を示す．

4.1 ホスト計算機およびオペレーティング・システム

XTC-UDS の対象機種は，UNISYS 2200/400, 600, 900 である．これらのホスト計算機と RLP は通常の入出力チャネル（BMC: Block Multiplexer Channel）を経由して接続されるため，ホスト計算機に対するハードウェア上の変更はない．

拡張トランザクション処理を実現するために加えられた OS 1100（2200 シリーズおよび 1100 シリーズ共通のオペレーティング・システム）の拡張機能は，XTC-EXEC と呼ばれている．XTC-EXEC は，OS 1100 の共用ファイル・アクセス機能である MHFS (Multi-Host File Sharing) を前提とする．XTC-EXEC として拡張された部



AP : Application Program
UREP : Unisys Repository
UDSC : Universal Data System Control

図 5　XTC-UDS 概要図

Fig. 5　XTC-UDS system overview

分は，RLP を入出力装置として利用するためのインタフェースが主であり，OS 1100 の互換性を損なうものではない．

4.2 RLP

睦結合システムの中核となるのは，RLP である．RLP は，ロック・モジュール，チャネル・モジュール，保守モジュール，および電源装置からなる．

RLP の障害は全ホストの停止につながるため，フォルト・トレラントなハードウェア機構が採られている．すなわち，ロック・モジュールは四重化，チャネル・モジュールおよび電源装置は二重化された冗長構成を持つ．

四つのロック・モジュールのうち，通常三つが稼働状態にあり，残る一つは待機状態にある．ロック要求は四つのロック・モジュールに送られ，並列に処理される．待機モジュールを除く三つのモジュールの処理結果は，チャネル・モジュール内のボータ (Voter) に返され，ボーティング (Voting) を行い，多数決により処理結果が定められる．異なった結果を返したロック・モジュールは，異常モジュールとして切り離され，ウォームアップ状態にある待機モジュールに切り替えられる．

保守モジュールを経由して，ホストから異常モジュールのオンライン診断が可能である．故障したロック・モジュールを取り替えた後，待機モジュールとして復帰させる作業はホット・メンテナンスと呼ばれ，RLP を止めることなく可能である．このようなフォルト・トレラント制御を始めとして，次に述べるロック制御，インバリデーション管理，ブロードキャスト，およびコミット順序番号管理は，ファームウェアで実現されている．

4.2.1 ロック制御

大規模オンライン・トランザクション処理システムで発生するロックを制御するためには，高速のロック制御専用プロセッサが必要となる．これは，RLP に処理要求が出される秒当たりのロック件数を試算してみれば明らかになる．たとえば，金融機関におけるオンライン・トランザクション・システムの代表的なトランザクションである普通領金の取り引き処理で発生する論理ロックの数は，処理負荷の少ないもので 1 取り引きあたり平均 15 と言われている．各ホストの処理能力を秒当たり 200 件とし，ホストの数を 4 とすると，RLP が処理すべきロックは秒当たり 12000 個(＝15×200×4)となる．このロックをソフトウェアで処理する場合，1 ロック処理に 1000 ステップを要すると仮定すれば，12 MIPS の処理能力が必要となる．

RLP で管理されるロックは，ファイル，ブロック，レコードおよびストリングの 4 種類である．ファイル，ブロック，そしてレコードは，それぞれ UDS におけるロックの単位に対応している．ストリング・ロックは TS 命令等，物理ロックに対応付けられる．UDS はロック単位に基づくロック階層管理 (Hierarchical Locking) を行うが，RLP にはロック階層管理機能はない．

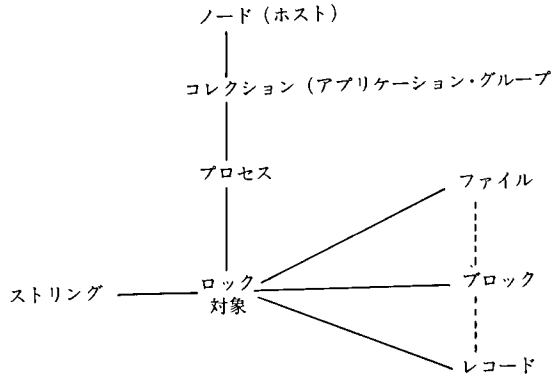


図 6　RLP のロック体系

Fig. 6　RLP lock hierarchy

各ロックは，XR (eXclusive Retrieval) ,XU (eXclusive Update) ,PR (Protected Retrieval) ,PU (Protected Update) ,NR (Non-protected Retrieval) ,NU (Non-protected Update) の 6 種類のクラスを持つ．UDS のロック管理の特徴として，ロック・クラスの変更を必要とする．RLP は，このロック・クラスの変更処理を取り込んでいる．

ロック要求が競合した場合，RLP には即時却下か待ちに入るかの二つの選択枝があ

る．待ちに入った場合には，一定時間を経過する度に RLP は WFGE (Wait-For-Graph Expression) によるデッドロック検知を行う．RLP は，デッドロックを検知した場合，定められた基準に基づき，デッドロック・ループを形成するプロセスの中から一つをロールバックすることにより，デッドロック状態を解除する．なお，RLP の管理対象外の資源に対する競合によるデッドロックもありえるため，一定時間経過しても RLP に対するロック要求が充足されないプロセスは，デッドロック状態にあると見做す時間による推定検知方法が併用できる．

UDS のロック管理では，ロックの継続期間 (Duration) を表すスレッド (Thread) とステップ (Step) という概念がある．RLP では，スレッドにプロセス (Process) が，ステップにパーシャル・プロセス (Partial Process) がそれぞれ対応する．ファイルおよびストリング・ロックはプロセス単位に，ブロックおよびレコード・ロックはパーシャル・プロセス単位に扱われる．

RLP では，施錠(Locking)はロック対象の個別処理しかできないが，解錠(Unlocking) はロック対象の個別処理ばかりでなく，ロック対象の集合に対する処理 (Aggregate Unlocking) が可能である．この集合処理として，あるプロセスが保有するすべてのロックの解錠 (Unlock process)，あるホストの特定アプリケーション・グループに属するすべてのロックの解錠 (Unlock subcollection) 等があげられる．ロックを保有するホストで障害が発生すると，ストリング・ロックはすべて自動的に解錠されるが，ファイル，ブロック，そしてレコード・ロックは残留し更新処理未完了に伴う汚染データ (Dirty data) へのアクセスを防止する．ただし，障害ホストが保有したロックにより待ち状態にいた他ホストのプロセスは，エラー通知とともに解放される．また，残留ロックと競合する新たなロック要求は，待ち入ることなく即時却下される．ロックの解錠は，基本的にはロックの保有者しか許されないが，残留ロックの開放に使われる特権的な解錠命令 (前述の Unlock subcollection) が障害回復ユーティリティのために用意されている．

4.2.2 インバリデーション管理

各ホストの主記憶上のバッファ領域に読み込まれたデータの首尾一貫性を保つためには，あるホストでデータが更新されると他ホストに通知し，バッファ上のデータをリフレッシュする必要がある．汎用データベース管理システムでは，このインバリデーション処理の高速性が要求される．疎結合システムでは，インバリデーション通知は，ホスト間通信処理で実現されてきたが，受信側ホストの処理が割り込み処理になるため，同期取りがむずかしく遅延時間が発生するので高速処理は望めない．

UDS は，まずロックを確立し，その後データをアクセスするという手順を常に踏む．したがって，RLP でロック情報とともに各ホストのバッファ上のデータの妥当性を示すバリディティ情報が維持管理され，ロック要求時に RLP よりバリディティ情報が UDS に渡されればよい．UDS は，XU で確立したロックを解錠する際に，他ホストのバリディティ情報の更新を行う(他ホスト無効化)．また，ディスク装置よりデータをバッファ上に読み込んだ時には，UDS は自ホストのバリディティ情報を更新する(自ホスト有効化)．ロールバック処理を行った時にも，UDS は自ホストのバリディティ情報の更新を行う (自ホスト無効化)．

4.2.3 ブロードキャスト

各ホスト上の基盤ソフトウェアが，RLP を経由してメッセージ交信を行うことができる．1対1あるいは1対n（一斉通知）の交信が可能である．UDS は，各ホストの主記憶上に展開されたシステム制御テーブルの同期取り等，比較的更新頻度の低いものを対象に使用している．

4.2.4 コミット順序番号管理

更新履歴ファイルは各ホストごとに存在し，各ホスト内で発生したトランザクションの処理順序にしたがって記録されている．しかし，これらトランザクションのシステム全体における処理順序は，RLP が発番するコミット順序番号で決められる．コミット順序番号は，更新履歴ファイルのデータブロックの書き出しのたびに，RLP で発番され，データブロックに書き込まれる．

4.3 UDS

XTC-UDS は，UDS の利用者から見た場合，プログラミングや運用に関して従来のインタフェースと異なる点はない．単一ホストと同様に，ネットワーク型(DMS)，関係型（RDMS)，階層型（SFS）の三つのデータ・モデルが使用できる．UDS の内部処理は，下記のように変更される．

4.3.1 ロック処理の変更点

従来のブロック管理の仕組みは踏襲されるが，ロック単位の階層管理とロック・クラスの管理にのみ利用される．ホスト間のロック管理，ロック要求競合時の待ち，デッドロック検知と解除処理は，OS 1100 と RLP に委ねられる．RLP のロック確立時にロック対象データのバリディティ情報が UDS に返される．バッファ上のデータの不正通知を受けた場合，UDS はバッファ上のデータをパージする処理を行う．

コミット処理に伴う解錠の対象が XU のロック・クラスで登録されている場合，UDS は他ホスト無効化のオプション付解錠指令を RLP に出す．ロールバック処理に伴う解錠の対象が XU のロック・クラスで登録されている場合も，UDS は他ホスト無効化のオプション付解錠指令を RLP に出す．

4.3.2 バッファ管理の変更点

ディスク装置よりデータをバッファ上に読み込んだ場合，UDS はバッファ上のデータがリフレッシュされたことを RLP に通知する（自ホスト有効化)．

4.3.3 UDS ブロードキャスト

主記憶装置上で管理されている UDS の各種制御用テーブルがいずれかのホストで変更された場合，残るホストも同期をとる必要がある．UDS は，RLP のブロードキャスト機能を利用し，他ホストに対して一斉通知を行う．UDS で使用される一斉通知は，送信元が送信先ホストからの応答待ちを行い送受信を管理する形式となっている．一定時間経過しても応答がない場合や通知に対する応答が却下された場合，送信元がその対処責任を負う．

他ホストの UDS からの通知を受信し処理するために，各ホストで常駐ラン RUB (Receiver for Uds Broadcast) が必要となる．RUB は，各ホストの立ち上げ時，最初のショート・リカバリを実行する前に起動される．また，UDS のアプリケーションが稼働中は，常に RUB を走行させておかねばならない．これは，従来の運用と異なる

点である．

4.4 統合回復ユーティリティ（IRU）

睦結合システムにおけるデータベースの障害回復の要件は，ログ・マージと局所リカバリであると前章で述べた．2200/1100シリーズの統合回復ユーティリティ（IRU）は，この要件を満たすべく下記のような機能拡張がなされた．

4.4.1 ショート・リカバリ

ホスト障害発生時更新中であったデータは，他ホストからの参照（Dirty read）を防止するため，RLPロックが施錠されたままである．ショート・リカバリ終了時に，IRUはRLPに対し，当該ホストの回復対象アプリケーション・グループが保有しているすべてのロックの解錠（Unlock Subcollection）を要求する処理が追加された．

4.4.2 ミーディアム・リカバリ

あるホストが障害で停止しその復旧に時間がかかる場合，稼働している他のホストから障害ホストのデータベース回復用ファイル（リテンション・ファイルと更新履歴ファイル）を用いて，データベースを回復する新しいリカバリ方式を，ミーディアム・リカバリと呼ぶ．

このリカバリを起動するホストでは，トランザクション処理と回復処理を同時に実行することが可能である．ショート・リカバリと同様に，リカバリ終了時にIRUはRLPに対し，障害ホストの回復対象アプリケーション・グループが保有しているすべてのロックの解錠（Unlock Subcollection）を要求する．遅延更新方式およびQBL方式両方とも，ミーディアム・リカバリの対象となる．QBL方式の場合，クィック・ビフォア・ルックスでデータベースを回復する時，更新履歴を採らねばならない．この履歴のことを，回復更新履歴（Recovered After Looks）と呼ぶ．ミーディアム・リカバリでは，障害ホストと回復実行ホストが異なるため，回復更新履歴に対し特別な配慮が必要となる．すなわち，どのホストの更新履歴ファイルに回復更新履歴を採るかということと，ロング・リカバリにおけるこの回復更新履歴の取り扱いが問題となる．OS 1100環境下では，障害ホストの更新履歴ファイルを回復実行ホストからアクセスすることができない．IRUは，回復更新履歴を回復実行ホストの更新履歴ファイルに採る．また，ロング・リカバリ処理において，回復更新履歴はステージングの対象とせず，即時適用するように変更された．

4.4.3 ログ・マージ

ショート・リカバリやミーディアム・リカバリという局所リカバリが失敗した場合，セレクティブ・リカバリやロング・リカバリといったシステム全体のリカバリが必要となる．システム全体をリカバリするためには，各ホストごとに採られている更新履歴をRLPのコミット順序番号順に整列し直すログ・マージ処理が必要である．IRUでは，このマージ処理は独立した処理ではなく，ロング・リカバリおよびセレクティブ・リカバリの内部処理として実施される．

4.5 ホスト間時刻調整機能

睦結合システムにおいて，各ホストで発生した事象を時系列的に正しく把握するためには，各ホストのクロックが示す時刻にズレが生じてはならない．このホスト間の時刻のズレをなくす機能を，ホスト間時刻調整機能という．XTC-UDSでは，NTTの

時報による時刻供給機をUTCと呼ぶ．UTCは各ホストのPCコンソールに正確な時刻を供給する．各ホストはPCコンソール上の正確な時計に合わせてホストのクロックを補正する．全ホストが同一の時刻を持つことは，単一システムのイメージが焦点となる疎結合システムや睦結合システムではシステム構成要件として不可欠である．

5. おわりに

以上，睦結合システムの特徴とその実現に向けてのデータベース技術の課題，そしてXTC-UDSでの実現方法を述べてきた．90年代の拡張アーキテクチャとして，コンピュータ業界各社とも疎結合システムを主軸に据え，高信頼性と処理能力の拡張性を目指したシステムを発表している．関係データベースと大型の疎結合システムを組み合わせた超大型データベース・サーバを開発しようという動きもある．まさに，XTC-UDSの時代である．

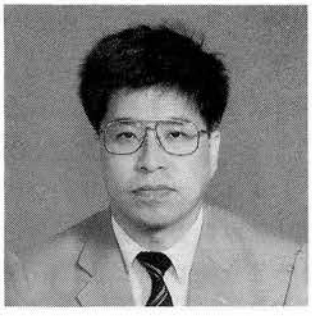
睦結合システムの今後の課題として，次の4点をあげておく．

1) ホスト間のロード・バランス
2) ホスト間相互作用と共用データの取り合いによるボトル・ネック
3) 複雑な問い合わせの処理効率と応答時間
4) システム管理上加わった複雑さ

これらの課題の解決に向けて，新たな技術開発が続けられている．

参考文献
[1] 岩波情報科学辞典，岩波書店，1990.
[2] 樫山汎，拡張トランザクション処理アーキテクチャ，UNISYS Technology Review，第24号，1990.
[3] P. G. Selinger, "The impact of Hardware on Database Systems", Database Systems of the 90s, Lecture Notes in Computer Science, 466, Springer-Verlag.
[4] W. Kim, "Highly Available Systems for Database Applications", Computing Surveys, ACM Vol. 16, No. 1, March 1984.

執筆者紹介 阪口喜好 (Kiyoshi Sakaguchi)
1970年京都大学工学部機械工学科卒業．1973年日本ユニシス(株)入社．応用ソフトウェア部にてDSS 1100の開発に従事．以来データベース関連プロダクトの開発，保守および適用支援に従事．1987年4月より1990年9月まで米国ユニシスにてXTC-TIP/UDSの開発に従事．現在，システム技術本部 データベース・ソフトウェア部 データベース一課課長．



UNISYS シリーズ 2200・1100UDS ロック制御

The UDS Locking Control for Unisys Series 2200/1100

下 田 隆 夫, 小 林 雅 浩

要 約 UNISYS シリーズ 2200・1100 の汎用データベース管理システム UDS (Universal Data System) のロック制御は，データベース管理システムが本来備えるべき基本的なすべてのロック制御機能を提供している．また，「マルチデータ・モデル」，「大規模データベース」および「ハイトラフィック・システム」のロック制御機能を提供するという特徴を持っている．

本稿では，一般的なデータベース管理システムでのロック制御が備えるべき機能を紹介し，それと対比する形で，UDS のロック制御の構造と機構について，各々の特徴に注目した技術解説を行う．とくに，大規模データベース・トランザクション・システム上での処理効率向上化技術，複数の異なるデータモデルのデータベースの統合的なロック管理，UDS 下のリレーショナル DBMS である RDMS 1100 のレコードレベルのロック管理等を解説する．また，将来の UDS の新技術となる XTC-UDS 下でのレコードロック・プロセッサ (RLP) を使用した UDS ロック制御を解説する．

Abstract The lock control provided in the Universal Data System (UDS), the general-purpose database management system for the Unisys Series 2200/1100, supports all the fundamental lock control functions which ought to be essentialy available in database management systems. It is also featured with the lock control support for a 'multi-data model,' a 'large database' and a 'high-traffic system.'

This paper discusses the lock control functionality that must be provided in an ordinary database management system, the structure and mechanism of UDS's lock control by comparison, and the technique adopted in each of the features. Special mention is also made of a technique for improved performance on a large database transaction system, of an integrated lock control for a database consisting of different data models, and of the record-level lock management supported by RDMS 1100, the relational DBMS under the UDS architecture.

An additional remark covers the UDS lock control for which a record lock processor (RLP) is used under XTC-UDS, a supposedly new future UDS architecture.

1. はじめに

UNISYS シリーズ 2200・1100 の汎用データベース管理システム UDS (Universal Data System) は，複数の異なるデータモデルを共存する形で統合的に管理し，大規模データベース・アプリケーションを効率良く処理・支援するプロダクトである．本稿で紹介する UDS のロック制御とは，UDS 下のデータベースに対する同時並行処理制御のことであり，この制御を処理する UDS 下の主要モジュールの一つの名前(ロッキング・サブシステムとも呼ぶ) としても使用する．図1は，UDS の構成を示している．

UDS ロック制御は，他の主要モジュールであるスレッド制御 (TCL) とキャッシュ

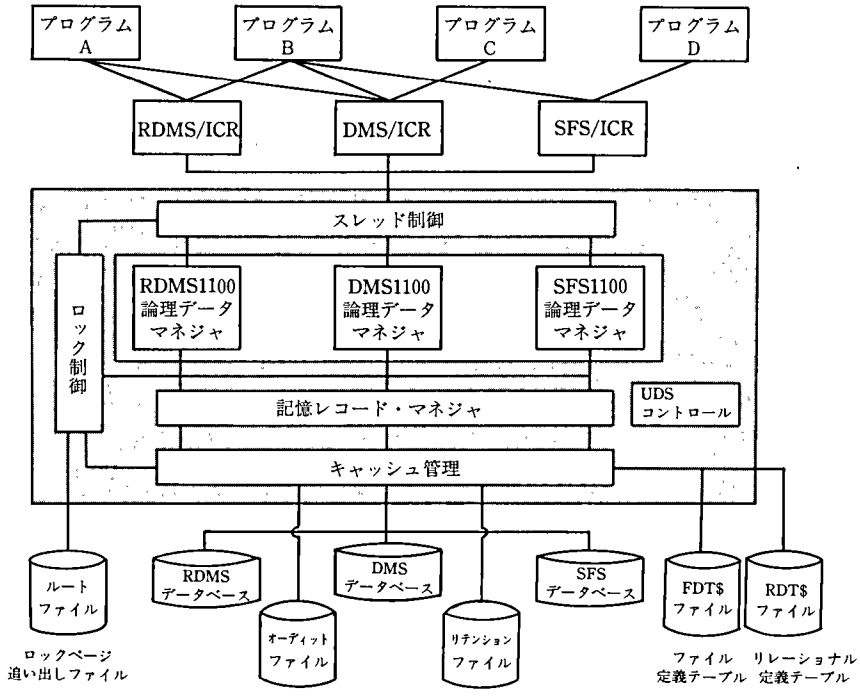


図1 UDSの構成

Fig. 1 UDS configuration

管理(CAM)等と連動して，UDS下の次に示す三つの異なるデータモデルのデータベース管理を行う．

① ネットワーク型データモデル（DMS1100）

② リレーショナル・データモデル（RDMS1100）

③ 階層型データモデル（SFS 1100）

UDSのロック制御は，データベース管理システム（以降DBMSと略す）下のロック制御として必要とされるすべての基本機能を保持している．

データベース上のデータは，同時並行に多数のデータベース使用者から共有してアクセスされる．このデータのアクセスには検索や更新があり，何らかの制約を与えないと同時並行処理における矛盾[1]（Lost Update, Dirty Read, Un-repeatable Read）が見られる．この制約として，UDSでは，J. Gray, I. Traiger およびF. Putzolu等の1976年に出版した論文 “Granularity of locks and degrees of consistency in a Shared Data Base”[2]で定義された一貫性の度数2をベースとしたロック方針を支援している．

これを単純に表現すると，次のようになる．

スレッドTは，以下を保証していること

① Tは，他の任意のスレッドが更新し未だコミットしていないレコード(群)を読まないし，更新（書き込み）も行わない．

② Tにより更新され未だコミットしていないレコード(群)を他のスレッドは読むことも更新（書き込み）も行わない．

一貫性の最高の度数は度数3であり，度数2の一貫性の他に以下の制約が加わる．

Tによって読まれた任意のレコードは，Tがコミットするまで他のスレッドにより更新されない．

度数3の一貫性は，表レベルに対する明示的なロック指令の使用により実現されている．実際には，スレッドTの検索するすべての表に対してSHARED, PROTECTED またはEXCLUSIVEのいずれかのロックモード設定を要求することになる．さらに，この後の二つのロックモードが設定されると，他のスレッドによりこのロックされた表の中にレコード（群）が挿入されることは決してないことが保証される．

一貫性のより低い度数（度数1未満）は“Read-Through”（他のスレッドにより更新されたが，未だコミットしていないレコード（群）を読むこと）を許す形で，将来提供されると言われているが，現時点では，その可能性は考慮されていない．

UDS ロック制御は，こうしたロック方針を支援し，とくに次に挙げる UDS に要求される二つの特徴となる機能を効率良く実現している．

① 大規模データベース・システムのロック制御機能

② ハイトラフィック・システムのロック制御機能

本稿では，こうしたUDSのロック制御の特徴を解りやすく解説するため，まず一般のDBMSに必要とされるロックの機能と体系を2章で述べ，それと対比させる形でUDSのロック制御に採用されている基本的な構成要素と機構の概要を3章で解説する．

また，UDSの特徴の一つである「複数のデータモデルの統合的な管理：一つのプログラムから同時に複数の異なるデータモデルのデータベースをアクセスできる」機能を提供するUDSのロック制御について，続く4章で解説する．

さらに，UDSの最新技術として注目されているXTC-UDS(eXtended Transaction Capacity-UDS) 下でのレコードロック・プロセッサ（RLP）を使用したUDSのロック制御（複数ホスト間での共用データベースのロック管理）を5章で解説する．

2. DBMSのロック体系

DBMSは通常データの完全性(一貫性)を保証するために，以下のものを管理する．

1) ロック単位（Granularity）
2) ロック階層（Hierarchical locks）
3) アクセスモードとロックモード
4) ロック保持期間（Lock duration）
5) デッドロック

2.1 ロック単位

ロック単位とは，データの完全性を確保するため自動的にロックが設定されるデータの集まりで，ロック可能な単位である．たとえば，ロック可能な単位とは，ファイル，ブロックおよびレコードである．

ロック単位はその大きさにより，コンカレンシとオーバヘッド間のトレードオフを表現するものである．

ロックの単位を小さく（細かく）すればするほどコンカレンシは向上する．しかし，

大量のレコード群をアクセスするトランザクションにとって,小さい単位のロックは,データベース・システムへの沢山の呼び出しオーバヘッドおよび,多くのロックを表現する格納オーバヘッドの増加を余儀なくする.大きい(粗い)単位のロック(たとえばファイル)はこのファイルの多くのレコードをアクセスするトランザクションにとっては,多くの場合有効になる.しかし,トランザクション間でロックの衝突の発生頻度が高くなり,トランザクションのコンカレンシを低下させる[3].

したがって,同じ一つのデータベース・システム内で異なるロック単位の共存が求められる.

2.2 ロック階層

ロック可能なデータ資源は,ロック単位を基にした階層構造を持つ.たとえばロックの階層構造は,図2のように親子関係の木構造として各レベルを表すことができる.

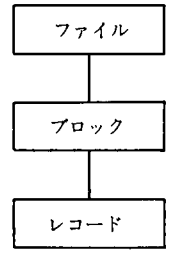


図2 ロック階層の例
Fig. 2 A sample lock hierarchy

ロック階層の各々のレベルは,得る必要のあるデータ資源を一意に識別することをデータベース・システムの使用者に許すものである.また,使用者にその階層の異なるレベルでロックすることを許すものである.

ロック階層のもとでは,データベース・システムの使用者が,あるレベルの特定の資源に対し排他アクセスを要求し,その要求が受け入れられた時には,その資源および暗黙のうちにそれ以下の各々に対して排他アクセスすることができる.すなわち,このアクセスモードのタイプは,要求資源レベルのルートとなる部分木全体をロックする.

2.3 アクセスモードとロックモード

通常,二つのトランザクションで,同じデータ資源に対するロック要求が同時に認められた場合,ロックが共存したこと(Compatible)を示す.ロック要求時のモードは,アクセスモードと呼ばれ,以前に他のトランザクションが同じデータ資源に対して行ったロック要求と共存するかどうかを決定付ける.データベース・システムは,データ資源に対するロック要求の認可を行うため,すでに認可したアクセスモードをロックモードとして管理する.

アクセスモードとロックモードは以下の特性を持つ.

①共有(Shared)モード　　:ロックの共存を許す.
②排他(Exclusive)モード　　:ロックの共存を許さない.

③部分排他（Intention）モード　：ロックの共存を部分的に許す．
④参照（Retrieval）モード　：データ資源を読み込むことを許す．
⑤更新（Update）モード　：データ資源を更新することを許す．

2.4 ロック保持期間

データの整合性を保つための最短ロックの保持期間であり，この保持期間を適切に設定することにより，余分なロック待ちを発生させずにトランザクションのコンカレンシを上げ，デッドロックの発生を最小限に抑える．

2.5 デッドロック

DBMS のロック管理では，デッドロックを検出しなくてはならない．一般に知られているデッドロック検出方式としては，TIMEOUT 方式および DEADLOCK DETECTION 方式等がある．

TIMEOUT 方式によるデッドロック検出とは，ロックの衝突により待ち行列に入っている使用者が，システムで定められたインターバルを過ぎた場合に検出する方法である．この方式は，システムに，より負荷がかかり，よりトランザクションのタイムアウトが発生する特徴を持っている．一般的にこの方法は，負荷の少ないシステムには向いているが，負荷の大きいシステムには向いていない．

DEADLOCK DETECTION 方式は，一般的には，データベース・システムにとって重い処理ではない．デッドロックを発見するアルゴリズムで代表的なものは，グラフ理論を用いてデッドロックを見つける手法である．この手法で用いる WAIT-FOR GRAPH という概念を次に示す[4]．

1) ノードはトランザクションとロックからなる．
2) グラフの端は次のように管理・構成される．
   ① あるデータ資源においてトランザクション T のロックが認知された場合，何も線は引かない．
   ② トランザクション T のロックがトランザクション L に対して待つ場合，T から L に線を引く（WAIT-FOR GRAPH の作成）．

WAIT-FOR GRAPH がサイクルになった場合，デッドロック解消は，サイクルを構成するトランザクションの中から一つ選び出し，ロールバックすることにより成される．

3. UDS のロック制御機構

本章では，まず 3.1 節で UDS のロック体系について述べる．次の 3.2 節では，UDS のロック処理の概要について説明を行う．続く 3.3 節では，UDS ロック制御のロック管理を解説する，最後の 3.4 節では UDS ロック制御機構の特徴について述べる．

3.1 UDS のロック体系

ここでは 2 章で述べたロック体系に対して UDS ロック制御では，どのように体系化しているのかを述べる．

3.1.1 UDS のロック単位

UDS のロック制御が提供しているロック単位は「ファイル」，「ブロック」または「レコード」単位のロックである．「ファイル」が最も大きい（粗い）ロック単位で，「レ

コード」が最も小さい（細かい）ロック単位である．また，ロック単位が「ブロック」か「レコード」かは各ファイルに対する指定（仕様）で決定される．

3.1.2 UDSのロック階層

UDSのロック制御は，図2に示すロック階層の下で使用者にデータ資源をロックすることを許している．

基本的にUDSのロック制御は，データ資源に対してロック機能を果たすために以下の3種類の情報を必要とする．

- 誰からの要求か？——使用者識別子（スレッド-ID）
- ロック対象のデータ資源は？——ロック・エンティティ識別子
- データ資源をどのように欲しいか？——アクセスモード

3.1.3 UDSのアクセスモードとロックモード

UDSのロック制御は，表1に示すロックモードを提供する（アクセスモードも表1と同様である）．

表1 ロックモード
Table 1 Lock mode

Lock Mode	説　　明
Null (NL)	データへのアクセスがない．すなわち要求がないことを表す．
Retrieval (R)	データ資源への検索アクセスを要求し，その下位のデータにXR, PRまたはRモードの設定を許す．
Update (U)	データ資源への更新アクセスを要求し，その下位に対して任意のモードの設定を許す．
Protected Retrieval(PR)	データ資源への検索アクセスを要求し，その下位のデータへのロック設定を不要とする．
Protected Update(PU)	データ資源への更新アクセスを要求し，その下位のデータへのロック設定を不要とする．
Exclusive Retrieval(XR)	データ資源とその下位のデータに対し，検索・排他アクセスを与える．
Exclusive Update(XU)	データ資源とその下位のデータに対し，更新・排他アクセスを与える．

UDSのロック制御は，ロック要求がなされた場合，指定されたアクセスモードが他のスレッドにより確立されたロックモードと共存するかどうかをコンフリクト（ロック衝突）・マトリックス（表2参照）を用いて調べる．

ロック衝突した場合，デッドロックを調べる（3.1.5項参照）．デッドロックが発生しない場合，相手のスレッドがロックを解放するまで待たされる．ロック衝突しない場合，同一のデータ資源に対し，すでに（この要求者が）ロックを確立していないなら，アクセスモードをロックモードとして確立する．

すでにロックを確立している時はロックモードの整合性を保つために，コンバート・マトリックス（表3参照）を用いてアクセスモードとロックモードを基にコンバートを行いロックを確立する．

表2　コンフリクト・マトリックス

Table 2　Confilict matrix

		存在するロックモード (Existing Lock Mode)						
		NL	R	U	PR	PU	XR	XU
アクセスモード	NL	T	T	T	T	T	T	T
	R	T	T	T	T	T	F	F
	U	T	T	T	F	F	F	F
	PR	T	T	F	T	F	F	F
	PU	T	T	F	F	F	F	F
	XR	T	F	F	F	F	F	F
	XU	T	F	F	F	F	F	F

T：コンフリクトする．
F：コンフリクトしない．

表3　コンバート・マトリックス

Table 3　Convert matrix

		存在するロックモード (Existing Lock Mode)						
		NL	R	U	PR	PU	XR	XU
アクセスモード	NL	NL	R	U	PR	PU	XR	XU
	R	R	R	U	PR	PU	XR	XU
	U	U	U	U	PU	PU	XU	XU
	PR	PR	PR	PR	PU	PR	XR	XU
	PU	PU	PU	PU	PU	PU	XU	XU
	XR	XR	XR	XU	XR	XU	XR	XU
	XU	XU	XU	XU	XU	XU	XU	XU

3.1.4　UDS のロック保持期間

UDS のロック制御が認識するロック保持期間を表4に示す．

表4　ロック期間

Table 4　Duration

ロック期間の単位	説　　明
スレッド期間	スレッドの終了まで
ステップ期間	コミットまで，スレッドの終了まで
Current of LDM 期間	明かに LDM から Unlock 要求がくるまで，コミットまで，またはスレッドの終了まで
コマンド期間	各 LDM の一つの指令の終了まで，コミットまで，またはスレッドの終了まで
リクエスト期間	次のロックエントリが作成されるまで，LDM の一つの指令の終了まで，コミットまで，またはスレッドの終了まで

保持期間に達したロックは解錠(Unlocking)される．ロック階層の上位のレベルに

属するロックが解錠されるとき，自動的にその下位レベルのロックも解錠される．また，ステップ期間とスレッド期間に達し解錠する場合，ロック集合に対する解錠処理（Aggregate Unlocking）がなされる．

3.1.5 UDS のデッドロック検出方法

UDS のデッドロックの検出処理は，UDS の主要モジュールである待ち行列管理ルーチン（QAD）が行う．

QAD はロック衝突が UDS のロック制御により検出された場合，衝突の相手のロック情報を QAD に渡す．QAD は，この情報をもとに WAIT-FOR GRAPH を作成する．作成後ただちに，デッドロック・サイクルを調査する．もしサイクルを発見した場合，デッドロック解消のため，そのサイクルを構成するスレッドの中で最も優先度の低いスレッドをロールバックする．サイクルができなかった場合は，そのスレッドを衝突の相手がロックを解放するまで待ちに入れる．

3.2 UDS のロック処理の概要

本節では，UDS のロック制御インタフェースと，さまざまなロック要求の処理概要を解説する．

3.2.1 UDS ロック制御インタフェース

UDS ロック制御へのインタフェースは，二つの基本的な「Lock 指令」と「Unlock 指令」を経由する．これらの指令は，特定の入力を受け入れ，使用者に要求のステータスを提供する．

以下に Lock および，Unlock 指令の簡単な説明を行う．

[Lock 指令]

Lock 指令では，UDS ロック制御に対し，以下の入力を与える．

［使用者識別子，ロック・ファシリティ識別子，ロック単位，アクセスモード，ロック保持期間，ロック階層］

一般に，Lock 処理のために UDS ロック制御は，以下の手順で処理される．

1) Lock 指令を実行したスレッドが，すでに要求したロック・エンティティにロックを確立しているかどうかを調査する．
2) 要求したロック階層のすぐ上位に，すでに適切なロックモードでロックされていることを確かめる．
3) 要求したアクセスモードが，そのロック・ファシリティに他のスレッドがすでに確立しているすべてのロックモードとロック衝突していないかを調査する．
4) 衝突がなければ，ロックを確立する（ただし，すでにロックを確立していた場合は，ロックモードのコンバート処理を行う）．ロック衝突していれば，待ち行列管理ルーチン（QAD）を呼び出し，デッドロックの調査を行う．

[Unlock 指令]

Unlock 指令は，その使用目的により以下の 3 種類の指令がある．

1) ロック集合に対する Unlock 指令

この指令では，UDS のロック制御に対し，次の入力を与える：[使用者識別子，ロック保持期間]

この Unlock 指令に対し UDS のロック制御は，以下の手順で処理される．

① 要求したスレッドの持つロック集合のうち，要求されたロック保持期間以下のロック期間に属するすべてのロックを解放する．

② ロック解放の際，もし，そのロックに対して待ち状態にあるスレッドがいるならば，QAD を呼び出し，待ち状態を解消する．

2) ファイルレベルのロックに対する Unlock 指令

この指令では，UDS のロック制御に対し，次の入力を与える：[使用者識別子，ロック・ファシリティのファイル識別子]

この Unlock 指令に対し，UDS のロック制御は以下の手順で処理される．

① 要求されたファイルレベルのロックを，このスレッドが確立していることを調べる．

② 要求されたファイルレベルのロックが参照系の場合，そのファイルレベルのロックおよび，その下位に属するすべてのロックを解放する．ファイルレベルのロックが更新系である場合，Current of LDM 期間以下の期間に属するすべてのロックを解放する．

③ ロック解放の際，もし，そのロックに対して待ち状態にあるスレッドがいるならば，QAD を呼び出し，待ち状態を解放する．

3) Current of LDM 期間のロックに対する Unlock 指令

この指令では，UDS のロック制御に対し，以下の入力を与える．

[使用者識別子，ロック階層，ロック・ファシリティ識別子]

この Unlock 指令に対し，UDS のロック制御は以下の手順で処理される．

① 要求されたロック・ファシリティに対して，Current of LDM 期間のロックを，このスレッドが確立していることを調べる．

② 要求されたロックを解放する．

③ ロック解放の際，もしそのロックに対して待ち状態にあるスレッドがいるならば，QAD を呼び出し待ち状態を解消する．

3.3 UDS ロック制御のロック管理

この節では，UDS ロック制御がどのようにロックを管理しているのかを述べる．

3.3.1 ロック管理の概要

UDS ロック制御は，使用者がアクセスするデータ資源の物理的なロックをロック・エントリとして管理する．

UDS ロック制御は，以下の 4 種類のロック・エントリを管理する．

1) ファイルロック・エントリ——ファイルレベルのエントリ

2) ブロックロック・エントリ——ブロックレベルのエントリ

3) シュードロック・エントリ——ブロックレベルのエントリ
（レコードロック・エントリを管理するのに用いる）

4) レコードロック・エントリ——レコードレベルのエントリ

個々のロック・エントリは以下の情報を待つ．

・ロック階層　・ロック単位　・ロックモード
・ロック保持期間　・ロック・ファシリティ
・ロックチェイン（後述）　・ユーザチェイン（後述）

UDS ロック制御は共用のデータ資源に対するアクセス認可のために，Global Lock List を保守する．Global Lock List とは，あるデータ資源をアクセスすることが許された複数（もしくは一人）の使用者の持つロックエントリの集合を，個々のデータ資源単位でリスト管理したものである．この Global Lock List は，ロック階層の各レベルで異なる手法で管理されている（後述）．

UDS ロック制御は，スレッドの持つロックの解錠処理で対象ロックの集合解錠処理を行うため，User Lock List を保守する．User Lock List とは，スレッドの持つロックエントリの集合をロック期間単位でリスト管理したものである．

UDS ロック制御は，使用可能なエントリをフリーエントリとして管理する．

UDS ロック制御がロックを確立する操作は，要求されたロック・ファシリティに対するアクセスモードが，そのロック・ファシリティの Global Lock List の中のすべてのロックエントリの持つロックモードとロック衝突しないことを確認した後，フリーエントリからロックエントリを作成し（ただし，すでにロックエントリが作成されている場合には，そのロックエントリを使用する），Global Lock List と User Lock List に追加することによりなされる．

また，UDS ロック制御がロックを解放する操作は，対象となるロックエントリを Global Lock List と User Lock List から取り除き，そのロックエントリをフリーエントリにすることによりなされる．

このロックエントリとフリーエントリは，ロックバンクと呼ばれる一つのコモンデータバンク上に保持される．

以下に，ロックエントリが Global Lock List，User Lock List としてどのように管理されているか，およびロックエントリを管理するロックバンクの構成について説明する．

[Global Lock List の管理]

データ資源（ロック・ファシリティ）に対するアクセス認可の管理をする Global Lock List は，ロック階層ごとに以下のように管理される．

1) ファイルレベル……このレベルの個々の Global Lock List は，ファイル記述テーブル（FDT）から Global Lock List を構成する各ファイルロック・エントリをリンク付けした双方向のチェイン（ファイルチェイン）により管理される．

ファイルチェイン内でのリンク付けの順番は，ロックの衝突の検証を効率良く行うため，個々のファイルロック・エントリのロックモードの強い順になっている（図 3）．

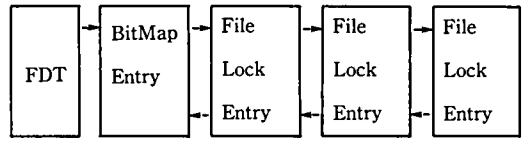


図3 ファイルチェイン

Fig. 3 File lock chain

2) ブロックレベル……このレベルの Global Lock List は，ハッシュチェインにより管理されている．このチェインは，ブロックロック・エントリとシュードロック・エントリを双方向にリンクしており，ハッシュテーブル［ロックバンクの構造］上のハッシュエントリからリンク付けされている．このハッシュエントリの総数は 997 であり，トランザクション処理を高速に行うことを考慮してある．ハッシングでは，ロック・ファシリティが使用されるため，同一のハッシュチェイン上に，複数の Global Lock List が存在する．このチェイン内でのリンク付けの順番は，Global Lock List のロック・ファシリティの識別子の昇順になっており，かつ同一の Global Lock List を構成する各ブロックロック・エントリとシュードロック・エントリの順番は，ロックモードの強さの降順になっている（図 4）．

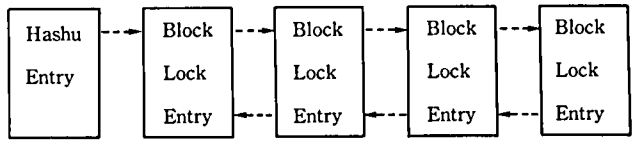


図 4　ブロックチェイン
Fig. 3　Block lock chain

3) レコードレベル……このレベルの個々の Global Lock List は，レコードチェインにより管理されている．このチェインは，ハッシュチェイン上に存在するシュードロック・エントリからレコードロック・エントリをリンク付けした双方向のチェインにより管理される．このチェイン上の並び順は，ハッシュチェインと同様である（図 5）．

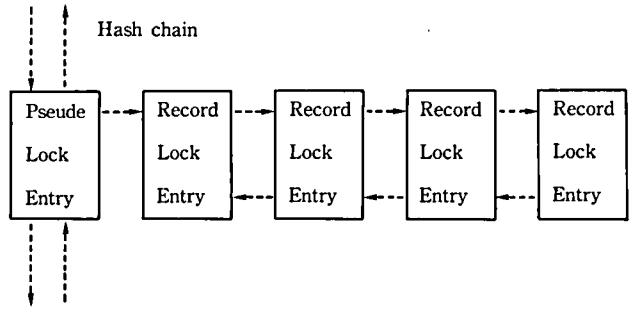


図 5　レコードチェイン
Fig. 5　Record lock chain

［User Lock List の管理］

User Lock List は，スレッドの持つすべてのロックエントリをロック保持期間単位にユーザチェインで管理している．このユーザチェインは，スレッド固有のバンク（スレッドデータ・バンク）上のテーブルより片方向に各ロックエントリをリンク付けしている（図 6）．

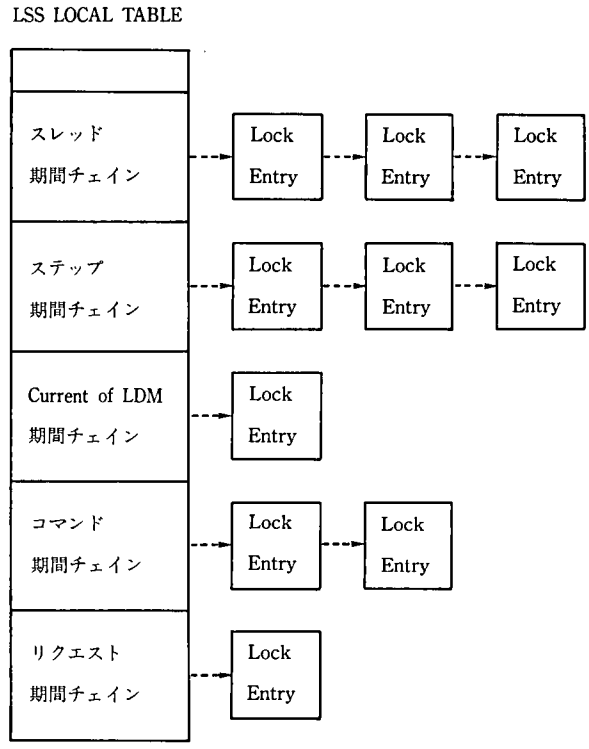


図6　ユーザチェイン
Fig. 6　User chain

[ロック・バンクの構造]

ロックエントリは，まずロックバンク上の Page 0 領域に作られる．Page 0 領域の大きさは，ロックバンクの 60%の大きさを持ち，かつ Page 0 領域で保持できるロックエントリの個数は，UDS の標準構成で約 4500 エントリである．この領域は，UDS ロック制御が管理する．

そこが溢れると CAM 領域にページを確保して管理する．このページは，31 ロックエントリが収容できる．CAM 領域はロックバンクの 35%の大きさを持ち，UDS のモジュールの一つであるキャッシュ管理（CAM）により管理される．

この CAM 領域を使い尽くすと，ページはルートファイルに掃き出され，CAM 領域とルートファイルとの間で I/O（入出力）を行いながら管理する．Page 0 領域は，ロックバンクが足りなくなった場合でもルートファイルへ追い出されることはない．図 7 は，UDS ロックバンクの構成を示している．

PBDs 領域は，CAM が CAM 領域のページ管理をするための PBD の領域である．

ロックバンク上のロックエントリのデータ構造は，ハッシュおよびチェインである．ハッシュテーブルは，ロックエントリのハッシュ管理に用いる 997 個のハッシュエントリから構成されている．

また，UDS データバンク上には，ロックバンク全体の制御を行うグローバルロック管理テーブルがある．

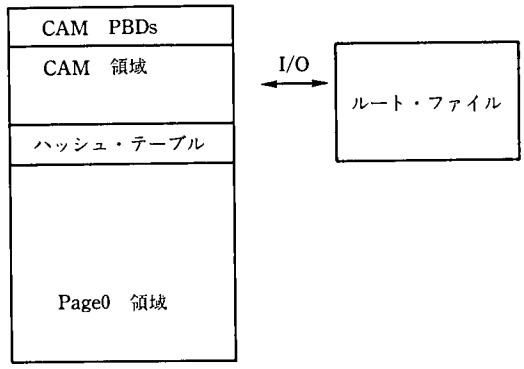


図7 ロックバンク構成
Fig. 7 Lock bank description

3.4 UDS ロック制御機能の特徴

UDS ロック制御は，次の特徴となる機能を効率良く実現するために設計されている．

① 大規模データベース・システムのロック制御機能
② ハイトラフィック・システムのロック制御機能

これらの制御機能をどの制御機構が実現しているのかをまとめて解説する．

1) ロックバンクとルートファイルを管理することにより，最大で同時に約 300 万個のロックエントリを制御できる．これは，同時にアクセスするファイル，ブロックおよびレコードの総数が 300 万個に収まっていれば良いことを意味している．
2) ロックエントリ方式でロックを管理しているため，データへのロックの書き込み方式に比べて入出力が少ないため効率良くロック制御ができる．
3) Global Lock List は，一番強いロックモードを持つロックエントリが先頭にあるため，ロックの衝突の検出が容易にできる．
4) ロックの解放が効率良くできるように User Lock List の管理は設計されている．
5) ハッシュチェインでブロックロック・エントリを管理しているため，ブロックロックの同時アクセス負荷を軽減している．

4. マルチデータ・モデルの支援

UDS ロック制御は，前述の通りマルチデータ・モデルの支援のため，物理レベルでの共通ロック機構を採用し，一元管理を行っている．すなわち，各データモデルに対応した表 5 に示すような階層を使用し，異なるデータモデル間でのデッドロック検知と解消を可能にしている．

ここで，RDMS の第 3 位（最下位）レベルの使用はオプションであり，記憶域ごとの選択が可能になっている．この RDMS のレコードレベルのロックは，リレーショナル DBMS を基幹データベースに使用する際の「ハイトラフィック処理での応答時間に与える排他待ち問題」改善のため，必要とされている機能である[5]．とくにほとんど

表5 データモデルとロックレベル

Table 5 Data model and Lock level

データモデル	DMS 1100	RDMS 1100	SFS 1100
第1位レベル	エリア	記憶域(表)	ファイル
第2位レベル	ページ	ページ	ブロック
第3位レベル		レコード	

の競合他社製品では未だ実現されていない機能性であり，特徴の一つと言える．

デッドロック検知の例題を考えてみる．

まず，スレッド1がRDMSの表Aのある行を更新してから，DMSのレコード型Bのあるオカレンスを検索しようとする．続いて同時に，スレッド2がDMSのレコード型Bのそのオカレンスを更新してから，RDMSの表Aのある行を検索しようとする．この時に，各々の対象行とオカレンスが同一ページにあると仮定すると，デッドロックの状態が発生することがある(図8)．UDSのロック制御は，このようなデッドロック検知と解消を無駄なく支援する．

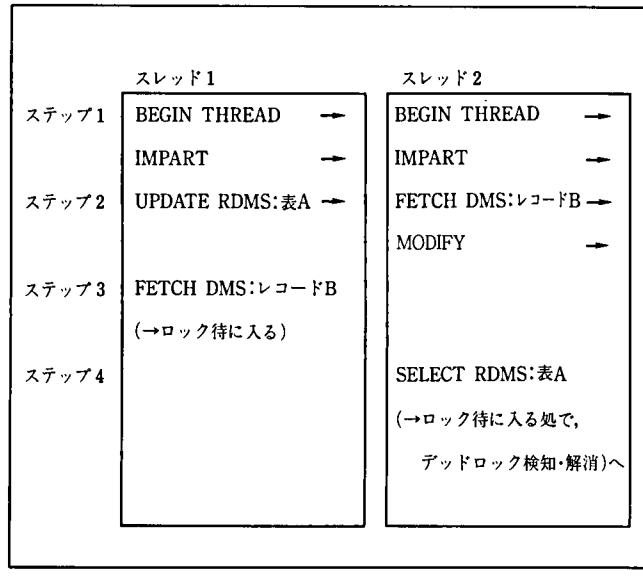


図8 マルチデータ・モデル下のデッドロック

Fig. 8 Deadlock under multi data models

もちろん，同時SFSアクセスによるDMSとSFS間，またはRDMSとSFS間のデッドロック検知と解消も可能であり，3者以上でのスレッド間のデッドロック検知・解消も可能である．

以上見てきたように，DMS，RDMS，SFS間のデータをどのように検索・更新してもデータベースの一貫性を保つようにロック制御がなされ，デッドロック検知・解消も問題なく処理・管理されている．

さらに将来の話になるが，UDSではこの三つのデータモデルに加えて，オブジェクト指向データモデルの提供を予定している．この際にも，UDSのロック制御はこの共

通ロック機構を応用した一元管理を行い，オブジェクト指向データモデルが単に追加された形でのマルチデータ・モデルを支援することができる（図 9）.

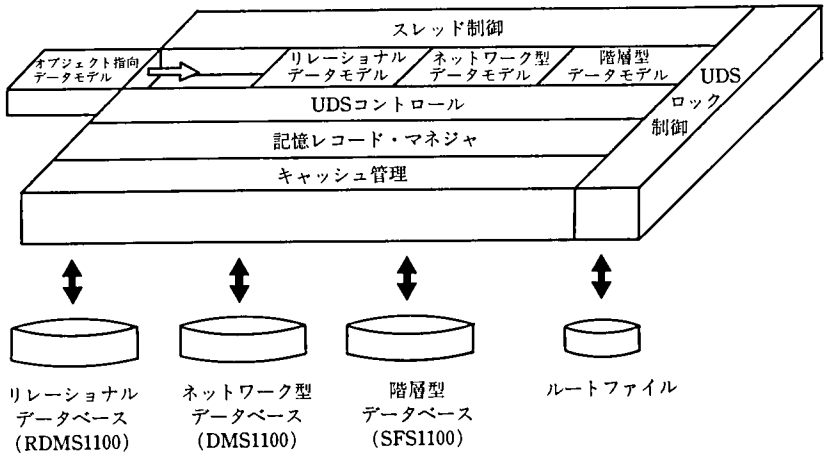


図 9　UDS 下のオブジェクト指向データモデル

Fig. 9　Object oriented data model under UDS

5. XTC-UDS でのロック制御

XTC-UDS は，弊社の新アーキテクチャ Extended Transaction Processing Architecture (XTPA)のもとで Closely Coupled System（睦結合システム）の各ホストからデータベースの共有を可能にするプロダクトである（図 10）.

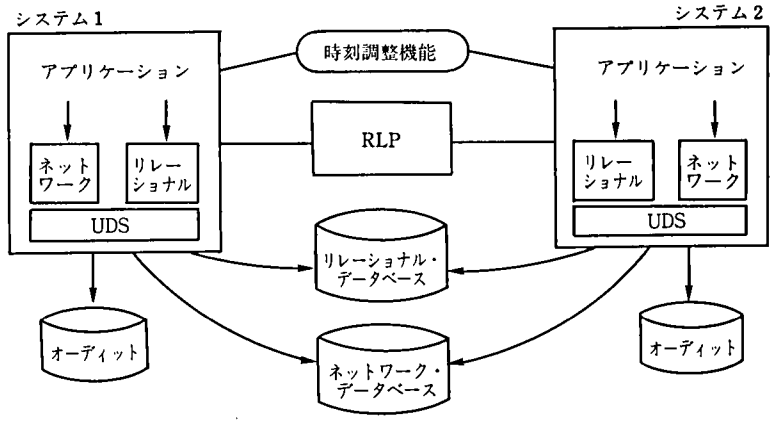


図 10　XTC-UDS 構成図

Fig. 10　Configration of XTC-UDS

この XTPA のもとでのロックの管理は，レコードロック・プロセッサ（RLP）と XTC-UDS のロック制御が行っている.

RLP は睦結合システムの全ロック情報を一元管理し，ロック衝突の検査，Queue/

Dequeue処理，デッドロックの検出，の役割を担っている．

また，XTC-UDSのロック制御は，ロックレベルの整合性の検査，User Lock Listの管理，の役割を担っている．

次にXTC-UDSのロック制御が，使用者からのLock要求/Unlock要求をどのように制御しているのかを紹介する．

1) Lock要求時の制御

① XTC-UDSのロック制御は，その使用者が十分なロックモードをLock要求のあった対象ロック・ファシリティに掛けているかどうかをそのエンティティの属するGlobal Lock Listから調べる．

② 十分なロックモードを掛けていれば⑤の処理を行う．掛けていなければ，RLPにロック要求を行う．

③ RLPがロック衝突の検査を行い，結果をXTC-UDSのロック制御に返す．XTC-UDSのロック制御はその結果を検証する．

④ デッドロックが起きていればロールバック処理を行う．

⑤ ロックエントリを作成し，Global Lock ListとUser Lock Listを更新する．

2) Unlock要求時の制御

User Lock Listをたどり RLPに Unlock要求を行う．その際次の情報を渡す．
(ファシリティ識別名，使用者識別名)

6. おわりに

以上，現在リリースされているUDSのレベルでの「UDSロック制御」を中心に解説してきた．一部に将来の「オブジェクト指向データモデル」および「XTC-UDS下のRLPを使用した共用データベース」でのUDSロック制御についても触れたが，解説できなかったUDSロックに関する今後の課題を，最後にまとめの意味で挙げておくことにする．

1) 分散データベース環境下でのロック制御

2) オブジェクト指向データベースのロック制御

3) EM化（拡張モードで稼働する）UDS下でのロック制御の効率改善・検討1
(ルートファイルへの追い出しのタイミング減少)

4) EM化UDS下でのロック制御の効率改善・検討2
(ラージロック・バンクを利用した処理効率改善)

5) XTC-UDS下でのロック制御における更なる効率改善・検討
(とくに，RDMS 1100データベースのロック制御について)

ここに挙げた課題を眺めて見ると，UDSを取り巻く新機能/新技術の下での課題ばかりであることに気付く．これらの新技術の下でも，データベース一貫性確保のためのロック制御の果たす役割は大きい．

参考文献 [1] J. D. Ullman, "Principles of DATABASE SYSTEMS Second Edition" Computer Science Press, Inc. 1982.

[2] J. N. Gray, R. A. Lorie, G. R. Putzolu, I. L. Traiger "Granularity of Locks and Degrees of Consistency in a Shared Data Base", in Modeling in Data Base Management System, (Nijssen editor), North Holland, 1976.
[3] J. R. Jordan, J. Benerjee, R. B. Batman, "Precision Locks", The proceeding of the 1981 ACM SIGMOD International Conference on Management of Data, pp. 143～147.
[4] J. N Gray, "Notes on Data Base Operating Systems", IBM Reserch Laboratory San Jose, California. 95193, summer 1977.
[5] 「基幹データベースをめざすメインフレーマー系 RDB の課題」分野解説　データベース管理 DP 3-713-071 日経データプロ・EDP 1991.9.

執筆者紹介　下　田　隆　夫 (Takao Shimoda)
昭和 27 年生, 49 年中央大学理工学部数学科卒業. 同年日本ユニシス(株)入社. データベース管理システムINFORM, DMS, RDMSの開発・保守作業に従事. 現在, システム技術本部データベースソフトウェア部所属. SC21/WG 3 SQL SG 委員.

小　林　雅　浩 (Masahiro Kobayashi)
昭和 36 年生. 61 年東京電機大学理工学部数理学科卒業. 同年日本ユニシス(株)入社. データベース管理システムUDS の開発保守作業に従事. 現在, システム技術本部データベースソフトウェア部所属.

UDSコントロールのバッファ管理 ——バッファ管理の変遷と今後の課題

Buffer Management in UDS Control ——The History of Buffer Management and its Prospects

大田静枝, 行成敦

要約 UNISYSシリーズ2200・1100汎用データベース管理システム（DBMS）UDS 1100の特徴の一つに，“大規模ページバッファ”がある．これはUDS以前のDBMSであるDMS 1100には実装されていなかったものであり，したがってページバッファ管理方式も大きく異なる．バッファ管理方式はDBMSの効率に影響を及ぼすが，その性能はデータベースの構造やアプリケーション・プログラムの参照方法等，使用者の環境と密接に関連し合う．そのため，互いの相性が悪いと，大規模ページバッファの威力が発揮できないこともある．そこでUDSでは使用者の環境に合わせて，バッファ管理方式を調整するためのインタフェースも提供している．

本稿は，UDS 1100のバッファ管理方式の特徴と，各使用者の環境でその効力を十分に発揮できる調整方法について述べる．さらに，より知的なバッファ管理となるための改良点について考察する．

Abstract UDS1100, the general-purpose database management system (DBMS) for the Series 2200/1100, supports a “large-scale page buffer” as one of its major features. Since this functionality was not imbedded in DMS 1100, predecessor of UDS 1100, a significant difference is seen in the method of buffer management between the two. The way of managing buffers has a great effect on the efficiency of DBMS performance which is closely related to a user environment ; e.g., how the database is structured, how applications programs are accessed. The fact that buffer management and a user environment are not suited for each other sometimes prevents a large page buffer from wielding its full power. Then, UDS 1100 also provides an interface to partially control buffer management in accordance with a user environment.

This article describes the advantages of UDS 1100's buffer management and the way of tuning it which gives individual users the greatest possible benefit. Also considered are some enhancement programs to make its buffer management more intelligent.

1. はじめに

データベース管理システム（DBMS）の効率の重要な評価の一つに入出力回数がある．入出力回数はデータベースのスキーマ構造(論理構造および配置方式)，検索方式，およびアプリケーション・プログラムの処理手順等の使用者環境と，DBMSのI/Oバッファの管理方式に影響される．すなわち，DBMSのバッファ管理方式がさまざまな使用者環境に対してどこまで考慮して設計されているかが，DBMSの効率を左右する．しかし，一つのバッファ管理方式がすべての使用者環境において，効率面からの要求を満足させることには限界がある．そのためシリーズ2200・1100の汎用DBMS

UDS 1100 では，さまざまな使用者環境に対応できるよう，バッファ管理方式を調整する手段を提供している．

本稿は，UDS 1100 コントロール（以下 UDS コントロールと呼ぶ）のバッファ管理方式の特徴と，各使用者環境でその性能を発揮できる調整方法を紹介する．

2. DBMS のバッファ管理は何をするか

本章では，バッファ管理とは何かを理解するために，バッファ管理の役割と基本的な手法について述べる．

2.1 DBMS のバッファ管理の役割

アプリケーション・プログラムからデータベースを参照するには，そのデータを含んだページが主記憶上に存在していなければならない．DBMS はアプリケーション・プログラムに代わって，データベースへの入出力を行う．データベースに対する入出力は，各データベースを等しい大きさに分割したページ単位に行う．

図 1 に示すように，レコードの参照が発生すると，DBMS は内部的な管理情報からそのレコードが格納されている物理アドレス（ファイル，ページ）を決定し，目的レコードを含むページを DBMS 内のバッファに読み込む．したがって，レコード参照時，目的レコードは DBMS のバッファ経由でアプリケーション・プログラム内の領域に渡り，レコード更新時は逆にプログラム内の領域から DBMS のバッファ経由でデータベースに書き込まれる．

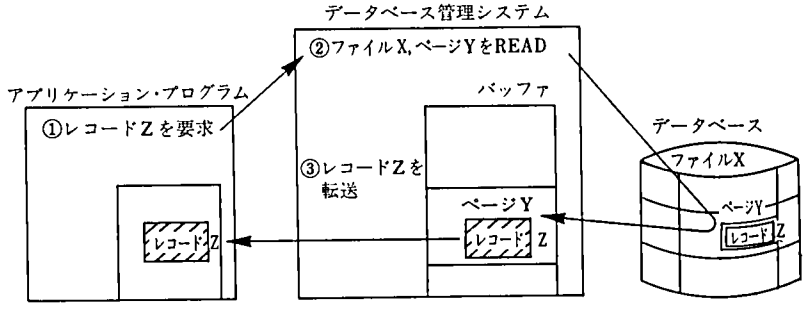


図 1　アプリケーション・プログラムがデータベース上のデータを得るまで

Fig. 1　Process of how application program gets data from database

ここで，データベースの入出力領域を管理する手法をバッファ管理方式と呼ぶ．DBMS におけるバッファ管理の役割は，要求者から要求ページの物理アドレスと大きさを与えられ，DBMS のバッファ上に要求ページの領域を確保してページを読み込み，そのアドレスを要求者に返すことである．この一連の作業の中で最も重要かつ難しい仕事が領域の確保であり，以降バッファ管理の役割とはこの領域確保の処理に限定する．

バッファ管理は，バッファ探索，アロケーション，およびリアロケーションの 3 要素から構成され次の手順で行う．

1)　バッファ上に要求ページが存在しているか否か，探索する（バッファ探索）．

2)　1) で要求ページがバッファ上に存在していなければ，空き領域を割り当てる（ア

ロケーション).

3) 2)で空き領域がなければ，バッファ上のページから犠牲者を選択し，その領域を割り当てる（リアロケーションおよびアロケーション).

この3要素は，実現する際には互いに密接に関連し合う．これらの要素の代表的な手法のいくつかはUDSコントロールのバッファ管理方式にも採用されているため，以降の節で説明する．

2.2 バッファ探索ロジック

バッファを直接探索すると，バッファが大きくなるにつれ，効率が悪くなる．通常は，図2に示すようにバッファ上のページに関する内部テーブル「ページテーブル」(ページ番号とバッファ上のアドレス情報を持つ)を設けて，これを効率良く探索する方法を考える．

ページテーブル内のエントリをページ番号をキーに分類し，二分木探索を使用したり，エントリに索引をつければ，効率良く探索できる．しかし，エントリの追加/削除の負荷がかかる．この負荷は，ページテーブル内にエントリを分類した順に並べるのではなく，チェインにつなぐことで軽減することができる．

効率向上のためにページ番号をキーにハッシング（ちらし法）を行い，ページテーブル・エントリを特定する方法も考えられる．ハッシングで重複した場合，オーバフロ（あふれ）用のエントリを割り当て，ホームエントリからチェインにつなぐ(図2)．探索とエントリの追加/削除両方のコストを考えると，このハッシングが最も効率が良い．

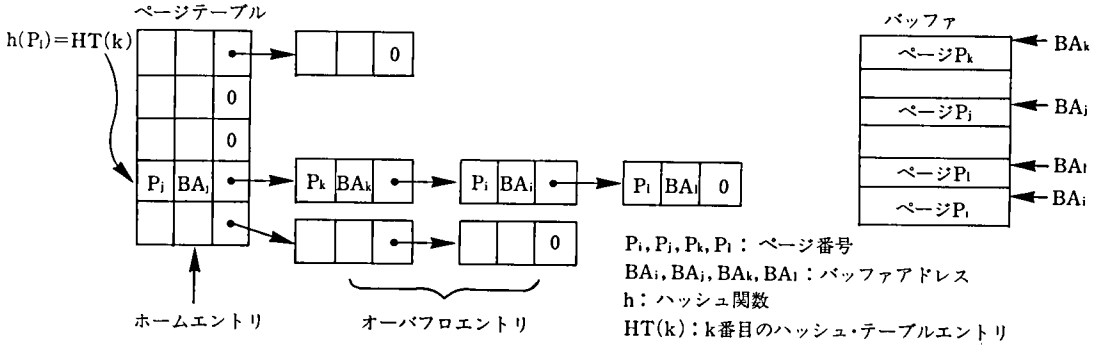


図2 ハッシング・ページテーブルを使用したバッファ探索[2]

Fig. 2 Use of a hash page table to improve buffer searching[2]

2.3 アロケーション・ロジック

複数のランが同時に稼働する環境では，バッファをある特性別に分割し，特性ごとに使用できる範囲を限定し，互いに影響を与えないように領域を割り当てる考慮も効率上有効である．特性には，ランの参照特性（乱処理/順処理），更新特性（遅延更新/即時更新）あるいはページタイプ（システムのページ/使用者のページ）等が考えられる．分割の指定を行う時期として，システムの導入時までに決定する（静的）場合と，導入後に決定（動的）する場合が考えられる．

また，分割の配分は特性ごとに等サイズにするか可変にするかの2通りが考えられ

る．使用環境の変化に対応するには，分割の指定は静的よりも動的の方が望ましい．動的を実装するためには，分割の配分を指定する使用者インタフェースが必要になる．

さらに，バッファ管理がその時の使用者環境から適切な分割の配分を推測し，環境の変化に合わせて自動的に配分を変更してくれることが望ましいが，まだ実現には到っていない．

分割された領域を使い切った場合には，次の節で説明するリアロケーション・ロジックを使用して領域を再使用する．

2.4 リアロケーション・ロジック

バッファ上のページから犠牲者を選択して，要求ページを読み込むための領域として再使用するアルゴリズムをいくつか紹介する．

FIFO (First-in, First-out) は最も前 (時間的) に読み込まれたページを犠牲にする．実現方法としては，図3で示すクロック方式のように，選択ポインタを用意してポインタが指すページを犠牲にし，選択する度にポインタを順番に動かすだけである．FIFO は参照頻度の考慮がないため，参照方法がシーケンシャルである場合のみ有効な方法である．

LFU (Least Frequently Used) は，最も参照頻度の少ないページを犠牲にする．LFU の場合は参照した時期の考慮がないため，短期間に集中的に参照が発生して参照回数を稼ぎ，その後まったく参照のないページでもバッファ上に残りやすくなる側面がある．そのため，LFU は不随的な選択基準として使用される程度である．

最も広く使用されているのは LRU (Least Recently Used) であり，参照した時期と回数の両方を考慮し，最も影響度の少ないページを犠牲にする．クロック方式を例に，LRU を実現する方法について説明する (図4)．

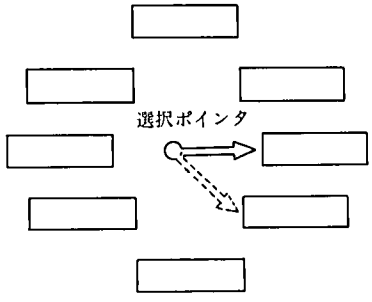


図 3　クロック方式による FIFO
Fig. 3　FIFO by the CLOCK algorithm

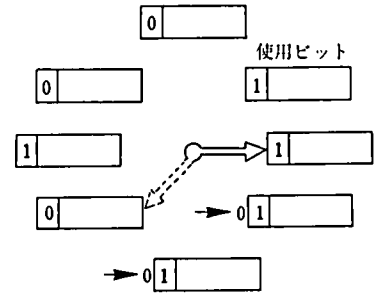


図 4　クロック方式による LRU
Fig. 4　LRU by the CLOCK algorithm

ページを参照すると，使用ビットを1にする．犠牲となるページの探索は選択ポインタを起点として時計回りに探索し，最初に出会った使用ビット0のページを犠牲にする．この探索の間に使用ビット1のページに出会うと，使用ビットを1から0に書き換え，選択ポインタも書き換えたページの次に動かす．この間に再度参照されたページは，次の探索でも犠牲にならない．

リアロケーションは，上述のアルゴリズムや他の二つの要素以外にも考慮すべきことがある．たとえば索引ページのように使用頻度の高いページや，システムが使用す

る特別なページには重みづけをしてバッファ上に残りやすくする．また，通常更新ページを犠牲にすると書き込みを伴い，参照のみのページと比べてコストが高くつくことも考慮しなければならない．

ページの要求者がページ内を参照している間は，そのページがメモリ上からなくならないようにロックをかけ(FIX すると呼ぶ)，参照の必要がなくなるとロックをはずす(UNFIX すると呼ぶ)．リアロケーション側もこの FIX/UNFIX メカニズムと連動を取り，FIX 中のページは対象としないようにする．FIX の期間は，ランの特性，データベースの構造，または上位モジュールが FIX/UNFIX をどのように実行するかに影響される．

3. UDS 以前のバッファ管理

本章では，UDS のバッファ管理方式の出現の背景を理解するために，UDS の前身の DBMS である DMS 1100 のバッファ管理方式とその問題点について述べる．

3.1 DMS1100 レベル 8R2 のバッファ管理

3.1.1 バッファ探索

バッファ探索はページテーブル探索である．ページテーブルはファイル単位に分割され，ページの参照要求が発生すると，テーブル内をファイル単位に直接探索する．

3.1.2 アロケーション

アロケーションはファイルの物理的属性のみ考慮し，バッファは TIP ファイル用と EXEC ファイル*用の 2 種類がある．すなわち，TIP ファイルは TIP ページバッファを使用し，EXEC ファイルは EXEC ページバッファを使用する．これを利用して，参照頻度が高くかつ更新がない索引ファイルを EXEC ファイルとし，他のファイルを TIP ファイルとすることで，互いに影響し合うことを防げる．

3.1.3 リアロケーション

リアロケーションは，基本的に LRU である．LRU を実現する手段として，チェイン方式を使用している．バッファ上のエントリはエントリの内容により，空きチェインあるいは参照チェインのいずれかにつなぐ (図 5)．

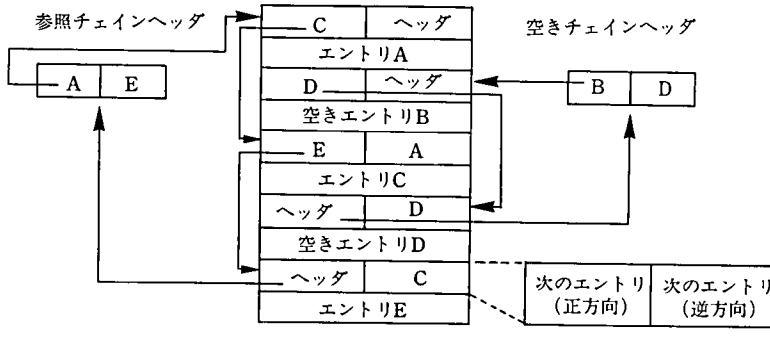


図 5 チェイン方式による LRU

Fig. 5 LRU by the CHAIN algorithm

* TIP ファイル/EXEC ファイル：OS 1100 では管理方法の異なる 2 種類のファイル，EXEC ファイルと TIP ファイルを提供している．EXEC ファイルは主にバッチ/デマンド処理に使用され，TIP ファイルは主にトランザクション処理に使用される．

空きチェイン（2方向）には，不要になった領域を空きエントリとして大きさの順につなぐ．新たにページを読み込んだ場合，またはバッファ上にすでに存在するページを再度参照した場合，そのページは参照チェイン(2方向)の先頭につなぐ．したがって，参照チェインを正方向にたどるにつれ，ページは古くなる．これらのチェインを使用し，次に示す二つのステップで新たに読み込むページのための領域を決定する．

① ステップ1

空きチェインを小さい大きさのエントリから，要求サイズを満たすものを探索する．

② ステップ2

参照チェインを古い順からたどり，UNFIX のエントリを探索する．この時そのエントリが要求サイズを満たさなくとも UNFIX であれば，物理的に上のエントリ，下のエントリ，さらにその下のエントリ…の順に，要求サイズを満たすまでエントリを併合する．この間 FIX のエントリに出会うと，そのエントリは使用できないためそれまでの併合はキャンセルして，参照チェインの次のエントリを探索し，同様の処理を行う．

条件に合うエントリが見つかった場合，更新ページなら書き込み（データベースあるいは保持ファイル）を行う．見つからなかった場合，待ちに入る．この待ちは他のランが UNFIX を行う，あるいはロールバック等で空き領域が発生すると解除される．

3.2 DMS1100 レベル 8R3

遅延更新の導入に伴い，遅延更新ランの入出力回数を少なくするため，レベル8R2のアロケーションとリアロケーションを次のように改良した．バッファサーチはレベル8R2と同じである．

3.2.1 アロケーション

ランのタイプを考慮して，バッファを遅延更新用と即時更新用とに分割する．具体的には，同時に稼働している全遅延更新ランがバッファ上に保持できる総更新ページ量の限界値(GLO: Global Limit，バッファの大きさに対するパーセンテージで表す)と，個々の遅延更新ランがバッファ上の保持できる更新ページ量の限界値（LOC: Local Limit，GLO に対するパーセンテージで表す）を設ける．

たとえば，バッファの大きさを 50 K 語，GLO を 80%，LOC を 20%とする．ここで，ある遅延更新ランが更新を行った際に GLO (50 K * 0.8=40 K) を越えると，そのランはロールバックされ，バッファ上のそのランの更新ページ領域は空き領域となる．

また，遅延更新ランが LOC（40 K * 0.2=8 K）を越えて更新すると，そのランの更新ページをまとめて1回の入出力で保持ファイルに書き込み，バッファ上のそのランの更新ページ領域は，空き領域となる．これをページングと呼ぶ．

次の節で説明するが，リアロケーションで遅延更新ランの更新ページを対象としなくなったため，このように限界値を設けて強制的に空き領域を作る必然性が出てきた．これらの限界値を動的に変更するインタフェースも提供している．

3.2.2 リアロケーション

アロケーションで設けた限界値を実現するためと，リアロケーションで遅延更新ラ

ンの更新ページを対象としないために，参照チェインは非更新ページ用チェイン，遅延更新ランの更新ページ用チェインおよび即時更新ランの更新ページ用チェインの3種類に分けられた．

ページは最初は非更新ページ用チェインにつながれるが，更新すると，更新したランのタイプに応じた更新チェインにつながれる．これらのチェインを使用して，ランのタイプにより，次に示すステップで新しくページを読み込む領域を確保する．

1) 遅延更新ランの場合

① ステップ1（レベル8R2と同じ）

空きチェインを小さい大きさのエントリから，要求サイズを満たすものを探索する．

② ステップ2

非更新ページ用チェインを古い順から探索し，そのエントリが要求サイズを満たさなくても，UNFIXかつ更新ページでなければ，物理的に上にあるエントリ，下にあるエントリ，さらにその下にあるエントリ…の順に要求サイズを満たすまでエントリを併合する．この間，FIXのエントリまたは更新ページに出会うと，そのエントリは使用できないため，それまでの併合はキャンセルして，非更新ページ用チェインの次のエントリを探索し，同様の処理を行う．

2) 即時更新ランの場合

① ステップ1（レベル8R2と同じ）

空きチェインを小さい大きさのエントリから，要求サイズを満たすものを探索する．

② ステップ2

非更新ページ用チェインを古い順から探索し，そのエントリが要求サイズを満たさなくても，UNFIXかつ遅延更新ランの更新ページでなければ，物理的に上にあるエントリ，下にあるエントリ，さらにその下にあるエントリ…の順に要求サイズを満たすまでエントリを併合する．この間，FIXのエントリあるいは遅延更新ランの更新ページに出会うとそのエントリは使用できないため，それまでの併合はキャンセルして，非更新ページ用チェインの次のエントリを探索し，同様の処理を行う．

③ ステップ3

即時更新ランの更新ページ用チェインを使用して，ステップ2と同様の探索を行う．

すなわち，遅延更新ランのリアロケーションは入出力がともなわず，また他のランのリアロケーションで，遅延更新ランの更新ページが対象となることはない．条件に合うエントリが見つかった場合，即時更新ランでエントリが更新ページならば，データベースへ書き込みを行う．見つからなかった場合，待ちに入る．この待ちは他のランがUNFIXを行うか，あるいはロールバックやページング等で空き領域が発生すると解除される．

3.3 問　題　点

3.1節および3.2節で説明したバッファ管理方式は，同時に稼働するランの数が20

未満程度であれば，50～60 K 語のバッファでも LRU ロジックにより効率よく稼働する．しかし，トランザクション量の増加に伴い同時に稼働するランも増加し，現行のバッファ管理方式では解決できない問題が現れてきた．バッファ探索アロケーション，およびリアロケーションの各視点から，これらの問題点を考察する．

1) バッファ探索の問題点……バッファ探索は同時に処理できない．したがって，複数のページ要求が同時に発生すると，バッファ探索の処理で待たされる．これは，ページテーブル探索のためのロックがシステムで一つであることに起因する．ページテーブル探索で要求ページが見つからないと，ページテーブル上に要求ページのエントリを作成する．このとき，そのエリアのページエントリ用ブロックが不足すると，ページテーブル内でブロックを動的に拡張する．したがって，同じページのエントリが複数登録されたり，ブロックの拡張により，複数エリア間でブロックの領域が重複しないよう，探索からエントリ作成までの処理の正当性を保証するために，ページテーブル全体をロックする必要がある．

2) アロケーションの問題点……バッファサイズには限界があり，通常 50 K 語，多くても 100 K 語程度である．レベル 8 R 3 の場合，同時に稼働するランの数が多くなると，GLO を越えてロールバックが発生しやすくなる．これを回避するために LOC を小さくすると，ページングが発生する．ページングが複数回発生し，ページングされたページを再度要求することになると，遅延更新ランの利点がなくなる．

3) リアロケーションの問題点

① リアロケーションを同時に処理できない．したがって，複数のリアロケーション要求が発生すると，リアロケーション処理待ちとなる．これは，リアロケーションのためのロックがシステムで一つであることに起因する．バッファのチェインを探索し，対象エントリをチェインからはずして付け換えるまでの処理で，チェインの整合性を保証するためには，チェイン全体をロックする必要がある．これは，チェイン方式を採用したことで発生した問題である．

② 異なるページサイズを使用すると，併合により LRU が無効になる．また，無駄な空き領域（フラグメンテーション）ができやすくなる．さらに，チェイン方式を実装する際に，個々のエントリにチェインの管理情報(4 語) を付加したため，ページの大きさが A，B の 2 種類で，A＝2 ＊ B であっても，A の領域に B が 2 ページ分おさまらず(A＜2 ＊ (B＋4))，無駄な空き領域ができてしまう．これは，チェイン方式の実装方法の問題である．

③ レベル 8 R 3 の場合，参照のみのページがリアロケーションの対象になりやすい，バッファ不足の問題とも関連するが，GLO を大きくすると，この傾向は顕著となり参照頻度の高いインデックス・ページであっても，常駐しにくくなる．

UDS コントロールは，これらの問題点を踏まえて設計している．次の章で新しいバッファ管理の世界を紹介する．

4. UDS コントロールにおけるバッファ管理の方法

4.1 特　　徴

UDS コントロール(以下 UDSC と呼ぶ)におけるバッファ管理には，以下の特徴がある．

1) 多重バンク*を使用することにより，ページバッファの大きさが論理的に無限になる．

2) 専用ページバンクを使用することにより，特定のファイルの特定の範囲を特定のバンクに読み込ませるよう指定できる．

3) 各バンクごとに，UDSC がブロック化する単位を指定することができる．専用ページバンクに読み込まれるデータベースのページの大きさとブロックの大きさを等しくすると，記憶域の使用効率が最大になり，ブロックの分割・併合が発生しないためアクセス効率も上昇する．

4) 各バンクごとにバンクの大きさを指定することができる．とくに専用ページバンクに対しては，適切な大きさを指定することにより，物理的な記憶域の無駄を省くことができる．

4.2 バッファ管理方法

バッファ管理方法には，バッファ探索，アロケーションおよびリアロケーションの3要素が関連する (2章参照)．

これら3要素を考慮した UDSC のバッファ管理の方法を述べ，各々の視点から分析する．

ページ要求が UDSC に対して出されると，

1) 要求ページが主記憶域上に存在するかどうか検査する．

この処理は内部テーブルの探索によって行われる．内部テーブルは，スキーマごと，ファイルごとにグループ化されたチェイン構造を採用している．

2) ページが主記憶域上になければ，

① 読み込むべきバンクを選択する．

基本的には，クロック方式を採用しており (2章参照)，システムが保持しているバンクのポインタからバンクを決定し，そのポインタを一つ進める．ただし，1ページ以上更新しているランがすでに1回以上ページを読み込んでおり，かつバンク選択時，稼働中のランの数がページバンクの数より多い時は，前回読み込んだバンクと同じバンクを選択する．これは，他のランに対する影響を最小にするためである．

② バンクを選択したら，各バンク内のクロック (バンク内のどこから探索するかを指し示すポインタで，探索が終了すると更新される)から内部テーブル(以下 PBD (Page Buffer Descriptor) と呼ぶ) を順処理で探索する．バンクは等しい大きさのブロックに分割されており，PBD はブロックと1対1の関係にあり，ブロックの物理的な順番と同じ順番に並んでいる．

PBD には以下の情報が含まれている．

---

* バンク：主記憶域の割り当ての単位であり，OS 1100 の下では，プログラムは同時に四つのバンクのみが可視である．この可視バンクを次々と切り換えることにより無限大の仮想記憶を実現している．

・管理している領域が空き領域か，読み込まれているページが参照ページか，または更新ページかに関する情報
・管理している領域の大きさ：一つの PBD が管理する領域はすべて等しいが，たとえば 2 ブロック分が空き状態であればそれらは併合される．このとき対応する二つの PBD のうち，最初の PBD は 2 ブロック分の大きさを保持し，次の PBD は前の PBD と“結合”していることを示す．
・LRU ビット：領域を確保した時に 1 を設定し，主記憶域のリアロケーションの際参照されると 0 に設定する．

上記三つの情報について，以下に示すレベル n に合致する領域を探す．

③ 条件に合う領域が見つかると，まずその領域が使用可能かどうか検査する．空き領域または読み込まれているページが参照ページであれば，無条件に使用される．読み込まれているページが更新ページであれば，そのページを更新したランがデータベースへの書き込み中でなければ，データベースまたは保持ファイルにその更新ページを書き込み領域を確保する．

④ 要求サイズが満たされれば，その領域を使用する．要求サイズが満たされなければ，物理的に次の領域を探索し，②，③のステップを繰り返して領域を確保し，領域の併合を行う．

⑤ 併合を繰り返しても要求サイズの領域がレベル n (後述) の探索のもとで見つからない場合には，レベル n+1 の基準の下で②，③，④を繰り返す．見つからない場合でも，併合された領域は併合された状態のままである．

⑥ レベル 1 からレベル 4 の要求で満たされなければ，次のバンクを探す．バンク内の探索方法は②～⑤と同じである．

⑦ すべてのバンクを探し，見つからなければ待ち行列に入れられる．

各レベルの探索基準は以下の通りである．

レベル 1：空き領域か参照ページが読み込まれている領域で，要求サイズの 100%以上のものを探索する．
LRU ビットが 1 のものは探索の対象外にする．LRU ビットに対して操作は行わない．

レベル 2：空き領域か参照ページが読み込まれている領域で，要求サイズの 50%以上のものを探索する．
LRU ビットが 1 のものは，そのビットを 0 にするが探索の対象としない．

レベル 3：空きまたは更新/参照ページの領域で，要求サイズの 50%以上のものを探索する．
LRU ビットが 1 のものは，そのビットを 0 にするが，探索の対象としない．

レベル 4：更新/参照ページの領域で要求サイズの 0%以上のものを探索する．
LRU ビットが 1 のものも探索の対象とする．

いずれのレベルにおいても，ランによって FIX と宣言されている領域は対象外にする．

4.3 DMS1100 8Rにおける問題点についての考察

DMS 1100 8R2および8R3での問題点に対してどのように解決されているかを以下に示す.

1) バッファ探索の問題点……ページが主記憶域上に存在するかどうか検査するためのロックはファイルごとに行われる. したがって, DMS 8Rに比べてコンカレンシが上がりシステムのスループットが上昇する.

2) アロケーションの問題点……ページバンクを複数枚構成することにより, バッファの大きさを論理的に無限大にできる.

3) リアロケーションの問題点

① リアロケーションのためのロックは各PBDで保持する. したがって, バッファ探索のコンカレンシはDMS 8Rに比較して上昇する.

② 異なるページサイズを使用すると, 無駄な空き領域ができうるが, PBD領域と実際のページを読み込む領域が完全に分離されているため, DMS 8Rで発生する, チェインの管理情報 (4語) によるフラグメンテーションは発生しない.

5. 効果的な使用方法

ここでは, 4章で述べられた事実をもとに効果的な使用方法について述べる.

1) 専用ページバンクを使用することにより, システムの制御テーブル的なページおよび参照頻度の高いインデックスページを常駐化させ, 入出力回数を減らす.

2) 専用ページバンクを使用し, 同一の大きさのページを同一バンクに対応づけることにより, ページサイズの不一致による記憶域の使用効率も最大になる.

3) 一般のページバンクはすべて同じ大きさにし, そのバンクに読み込まれるページの大きさを統一し, ブロックの大きさもページサイズに等しくする.

これにより, ページバンクによる片寄りおよび領域の分割/併合の負荷を減らし, フラグメンテーションをなくすことができる.

4) 参照ページがリアロケーションされやすい管理方法になっているので, 参照ランの比重が高いシステムでは, 更新ファイルを専用ページバンクに対応づけ, 更新ランによるリアロケーションの干渉を少なくする.

5) 更新ランが多いシステムでは, ページバンクの数を増やすことにより, 更新ランも多くのページバンクを使用できるようにして入出力回数を減らす.

6. 今後の課題

現在のUDSCの改良すべき点を, 主にプロダクト側からの視点で考察する.

1) バッファ探索の視点……UDSCのもとで管理されるデータベースのページは, スキーマごと, ファイルごとに管理されているが, 同一ファイル内では要求順にページ情報が蓄えられるため, ページが主記憶域上に存在するかどうかの検査は直接探索になる. ハッシング方法の採用等, 改善の余地がある.

2) アロケーションの視点

① 更新ランでも, 全バンクを使用対象とするかどうか, バンク選択基準を使用者が操作できるようにする.

② バンク選択の際，システムのクロックはバンクの使用状況を加味する等の考慮が必要である．空き領域が多いバンクをクロックに設定する等の考慮がほしい．

③ ランのタイプ（即時更新・遅延更新）によって，ページの操作が異なることを考えれば，ランのタイプごとに使用バッファを指定できる機能がほしい．

3) リアロケーションの視点

① 現在のレベル1→4の探索では，参照ページがリアロケーションの対象になりやすい．各レベルでの探索基準もシステムおよびアプリケーションによって変更するべきものであり，調整するためのインタフェースがほしい．

② LRU ビットは最初の参照かどうかを管理するにすぎない．参照頻度を加味して，参照頻度の少ないものをリアロケーションの対象とする方法が望まれる．

③ プリフェッチ機能が有効に働くように，論理データマネジャの要求指令に応じてプリフェッチする，論理データマネジャ・インタフェースがほしい．

④ 論理データマネジャごとに処理が異なること，互いのシステムが干渉することにより効率が低下することがありうることを考えれば，各論理データマネジャ専用のバンクを設定できる機能が必要である．

7. おわりに

以上，UDS 1100 のバッファ管理方式の概要，利点および問題点について述べた．ここでは触れなかったが，FIX/UNFIX は各論理データマネジャに任せられているので，論理データマネジャの視点から見た問題点の洗い出しおよび解決法の検討が今後必要である．また UDS 1100 の拡張モード対応において，拡張モードの特徴を最大限に利用したバッファ管理方式の検討が必要である．

参考文献 ［1］ 宮本勲，“Database Management System における Buffer 管理方式”，情報処理，1974 年 3 月，pp.172～179.

［2］ W. Effelsberg, T. Haerder, “Principles of Database Buffer Management, ACM Transactions on Database Systems, Vol. 9 No. 4, 1984 年 12 月, pp. 560～595.

執筆者紹介 大田静枝 (Shizue Ohta)

1957 年生．1980 年青山学院大学理工学部経営工学科卒業．同年日本ユニシス(株)入社．以来，一貫してシリーズ 2200・1100 のデータベース管理システムの保守，開発業務に従事．現在，システム技術本部 データベースソフトウェア部 データベース二課に所属．

行　成　　敦 (Atsushi Yukinari)

1961年生. 1986年東京大学大学院理学系研究科数学専門課程修士課程修了. 同年日本ユニシス(株)入社. 以来, 一貫してシリーズ2200•1100のデータベース管理システムの保守に従事. システム技術本部 データベースソフトウェア部 データベース一課に所属.

OSI RDA 概略——そのモデルとサービス

An Overview of OSI RDA——Its Model and Services

坂谷祥史, 島村隆一

要　約　RDAは異なるコンピュータ・システム間でデータベースの相互運用を実現するためのOSI応用層の規格である．本稿では，RDA基本標準の内容を，モデルおよびサービスの二つの側面から論じている．モデルの面からはOSIの他のサービス要素との関係を中心に，またサービスの面からはRDAが提供する各サービスの紹介と実際のデータ操作言語の転送および結果の受取を中心に述べる．さらに1991年11月に行われたINE'91におけるRDAのデモンストレーションについても，その概要を紹介する．

Abstract　Remote Data Access (RDA) is a standard for the OSI application layer drawn up for shared database operation in the hetrogeneous computer environment. This article describes basic RDA standards from two different aspects of its model and services, focusing on RDA's relationships with other OSI application service elements (ASE) when the model is discussed, and also on what services are available from RDA including the practical transmission of a data manipulation language and the reception of results when its services are explained.

In addition, a brief description is given of the RDA demonstration performed as part of the INE '91 in November, 1991.

1. はじめに

RDA (Remote Database Access：遠隔データベース・アクセス) は，異なるコンピュータ・システム上に分散配置されているデータベースを，相互にアクセスするためのOSI応用層の規格である．本規格は1985年にECMA (European Computer Manufactures Association：ヨーロッパ・コンピュータメーカ協会) から提出された技術レポートTR 30をベースにISOが規格制定作業を行っているものであり，1991年5月に国際規格案 (DIS) とする勧告がなされた．現在も1992年6月の国際標準化 (IS) をめざして作業中である．

日本においては，国際標準化作業と並行して国内の実装規約をINTAP (Interoperability Technology Association for Information Processing, Japan：情報処理相互運用技術協会) 主導の下に作成しているが，この実装規約の相互接続性を実証するためにINE'91*においてRDAのデモンストレーションが行われた．このデモは，1988年11月に行われたものに続く2回目のデモで，前回行われた検索機能のデモに加え更新機能のデモも行われた．弊社もINE'91に参加すべく作業を行ってきたが，ここで作られたソフトウェアは今回のデモに参加することを目的に作られた，いわばプロトタイプのソフトウェアであり，標準プロダクトとして提供するためには改良すべき点が数多く存在する．たとえば，INE'91では，RDAが提供しているすべてのサービス

* Interoperable Networking EventのOSIによる相互接続の公開実験のことである．1991年は11月5日から8日まで科学技術館において実施された．

を使用しているわけではないので，今回実装しなかった部分に関して標準プロダクトではどういう扱いにするかの検討が必須となる．

本稿は，INE '91 用に開発したプロトタイプ RDA をデータベース管理システムとのインタフェースを中心に紹介するとともに，標準プロダクト化へ向けての問題点の洗い出しおよびその解決案の検討を行うことを目的としている．ここではまず OSI RDA の概略を解説し（2 章），その後に INE '91 における実装を各モジュール間のインタフェースを中心に述べる（3 章）．そして今回の実装方法の中で，標準プロダクト化を行う時に問題になる事項を明確にし（4 章），その中の何点かに関し解決策を検討していく（5 章）．

2. OSI RDA 概要

2.1 RDA の規定範囲

RDA の規格は遠隔データベース・アクセスを実現するために必要な，クライアント（データベース操作の要求者）/サーバ（データベース操作の実行者）間のサービスとプロトコルのみを規定している．すなわち，位置透過性*等を含めた応用プログラムとのインタフェースは RDA の規定範囲外である．したがって，応用プログラムから明示的に遠隔データベースの所在情報を指定するような実装であったとしても，RDA としては“規格準拠”のシステムということになる．

また，今のところ RDA で使用することができるデータベースは ISO/IEC 9075（データベース言語 SQL）で定義されている SQL データベースのみである．

2.2 RDA の応用コンテキスト

前述したように，RDA は異なるコンピュータ・システム上に分散配置されているデータベースを，相互にアクセスするための OSI 応用層の規格である（図 1）．しかしながら RDA は，それ単独では RDA としての機能を果たすことができない．応用層の他の応用サービス要素（ASE：Application Service Element）と組み合わされて，はじめてその機能を果たせるようになる．この組み合わせは応用コンテキストと呼ばれ，RDA においては基本応用コンテキストと TP 応用コンテキストの二つが規格化されており，以下のような特徴を持つ．

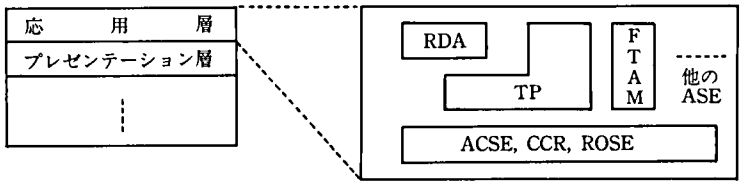
1) 基本応用コンテキスト……基本応用コンテキストは，RDA と ACSE, の二つの応用サービス要素からなるコンテキストである．このコンテキストでは 1 フェーズ・コミットメントのみが提供される．すなわち，複数サーバ間の同期をとった更新を保証することはできない．

2) TP 応用コンテキスト……TP 応用コンテキストは，RDA と ACSE, TP, CCR の四つの応用サービス要素からなるコンテキストである．このコンテキストでは 2 フェーズ・コミットメント**を提供しているので，複数サーバ間の同期をとった更新を保証することができる．

---

* データベース利用者から見た場合，実際は複数サイトに分割されて存在しているデータベースが一つのデータベースであるかのように扱えること．

** すべてのサーバの更新情報の保管が完了したことを確認した後で，データベースの実更新を行うことにより，複数サーバ間の同期をとった更新を保証する方法．



TP　：Distributed Transaction Processing
FTAM：File Transfer. Access and Management
ACSE：Association Control Service Element
CCR　：Concurrency Commitment and Recovery
ROSE：Remote Operation Service Element

図1　応用層における RDA の位置

Fig. 1　Location of RDA within OSI application layer

2.3　RDA が提供するサービス

遠隔データベース操作を実現するために RDA は，以下のようなサービスを提供している（表1）.

1)　ダイアログ管理サービス（R-Initialize, R-Terminate)……RDA ダイアログ* の確立および開放を行うためのサービスである.

2)　トランザクション管理サービス（R-BeginTransaction, R-Commit, R-Rollback)……基本応用コンテキストを使用する場合のみ提供されるサービスで，RDA トランザクションの開始およびコミット，ロールバックを要求するためのサービスである.

TP 応用コンテキストにおいてはトランザクション管理を TP ASE が行うため，このサービスは提供されない. RDA 使用者は RDA が提供するサービスに代わり，TP が提供するトランザクション管理サービスを使用する.

3)　制御サービス (R-Cancel, R-Status)……すでに要求ずみの RDA サービスをキャンセルする，あるいはその実行状況を問い合わせるためのサービスである.

4)　資源ハンドリング・サービス（R-Open, R-Close)……データ資源（データベース）の使用開始および使用終了を要求するためのサービスである.

5)　データベース言語サービス (R-ExecuteDBL, R-DefineDBL, R-InvokeDBL, R-DropDBL）……遠隔データベースにおいて，データベース言語の実行を要求するためのサービスである. このサービスには，要求されたデータベース操作を即時に実行する即時実行型 DBL サービス (R-ExecuteDBL) とあらかじめ DBL を登録しておき，後で引数のみを送信して当該 DBL を実行する蓄積実行型のサービス（R-DefineDBL, R-InvokeDBL, R-DropDBL）の2種類がある.

3.　INE '91 における実装

3.1　実装にあたっての基本方針

今回の開発は INE '91 に参加するために必要な機能を開発することを目的とし，以下のような基本方針の下に作業を進めた.

1)　実装する範囲はデモで用いる機能のみとし，それ以外の部分は考慮しない.

* クライアント/サーバ間に論理的な関係を確立するためのもの

表1 RDA サービスの使用例
Table 1 Usage example of the RDA service

クライアント		サーバ
	TP ダイアログの確立	
TP-BeginDialogue-req	→	TP-BeginDialogue-ind
	RDA ダイアログの確立	
R-BeginDialogue-req	→	R-BeginDialogue-ind
R-BeginDialogue-cnf	←	R-BeginDialogue-rsp
	データ資源のオープン	
R-Open-req	→	R-Open-ind
R-Open-cnf	←	R-Open-rsp
	トランザクションの開始	
TP-BeginTransaction-req	→	TP-BeginTransaction-ind
	データベース操作の実施	
R-ExecuteDBL-req	→	R-ExecuteDBL-ind
R-ExecuteDBL-cnf	←	R-ExecuteDBL-rsp
:		:
:		:
:		:
	トランザクションのコミット(フェーズ1)	
TP-Commit-req	→	TP-Commit-ind
TP-Commit-Result-ind	←	TP-Commit-Continue-req
	トランザクションのコミット(フェーズ2)	
TP-Done-req	→	TP-Commit-Result-ind
すべてのサーバから応答があった後		TP-Done-req
TP-Commit-Complete-ind	← コミットの完了 →	TP-Commit-Complete-ind
	データ資源のクローズ	
R-Close-req	→	R-Close-ind
R-Close-cnf	←	R-Close-rsp
	RDA ダイアログの開放	
R-EndDialogue-req	→	R-EndDialogue-ind
R-EndDialogue-cnf	←	R-EndDialogue-rsp
	TP ダイアログの開放	
TP-EndDialogue-ind	←	TP-EndDialogue-req

2フェーズコミットにおいてクライアントおよびサーバにコミット完了を通知するのは TP の役目である．すなわち，すべてのサーバからコミット完了の通知を受けた後 TP はクライアントに TP トランザクション全体のコミットが完了したことを通知する．また，サーバ側の TP はサーバがクライアントに対してコミット完了を通知した時点で TP トランザクション全体のコミットが完了したことをサーバに通知する．

2) 既存プロダクトは改造しない．

3) パフォーマンスは，応答時間がデモシナリオ上の許容範囲におさまることを目標とし，それ以上の改善は行わない．

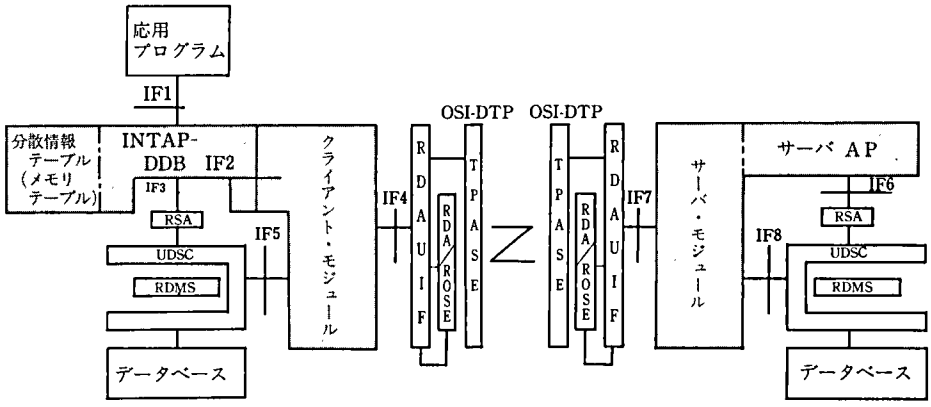
4) あくまでデモ対応の開発とし，そのままの形で商品化することは考えない．

5) 使用する RDMS のレベルは 4R1 とする．したがって，本文中には上位レベルですでに解決されている問題も問題点として一部記述されている．

3.2 モジュールとモジュール間のインタフェース

3.2.1 RDA のモジュール構成

今回の実装では，クライアント固有の INTAP-DDB (Distributed Date Base)，クライアント・モジュール，サーバ固有のサーバ・モジュール，サーバ AP そしてクラ



IF1：ユーザ・インタフェース
IF2：クライアント・インタフェース
IF3：RSAインタフェース（クライアント側）
IF4：RDAUIFインタフェース（クライアント側）
IF5：トランザクション・コマンド・インタフェース（クライアント側）
IF6：RSAインタフェース（サーバ側）
IF7：RDAUIFインタフェース（サーバ側）
IF8：トランザクション・コマンド・インタフェース（サーバ側）
網かけ部分は今回のデモで作成したデータベース関連のモジュールである。

図2　INTAP デモ モジュール図

Fig. 2　INTAP demonstration module

イアント/サーバに共通の RDAUIF(RDA User Interface), RDA/ROSE*, OSI-DTP の七つのモジュールを開発した（図 2).

以下, 各々のモジュールの機能を解説する. なお RDAUIF, RDA/ROSE, OSI-DTP の三つは, データベースと直接インタフェースをとらないモジュール群であり, 本技術報告のスコープ外であるため解説を省略する.

1)　INTAP-DDB……INTAP-DDB は, 応用プログラムと RSA (Relational Syntax Analyzer) 間および応用プログラムとクライアント・モジュール間のインタフェースで,

①　ユーザ環境のセーブおよびリストア,

②　分散不可視の実現,

を行う.

これらのうち分散不可視は, 次のような方法で実現される. まず, ユーザから与えられた SQL 文から分散ノードを決定するために必要なキーワード**を抽出し, そのキーワードで分散情報テーブル***を検索する(表 2). もしテーブル上にキーワードが存在した場合は, アクセスしようとしているデータベースが分散ノードに存在するものと判断し, 存在しなかった場合はローカルノードに対するアクセスと判断する. ただし, ユーザからの要求がスレッド・コマンドまたはトランザクション・コマンドであった場合は, キーワードの抽出は行わず無条件にロ

---

* 最新の ISO 標準では ROSE は使用しないと規定されているが, INE '91 で使用した国際規格では ROSE の使用は任意であった. 弊社は ROSE を使用する形で RDA の実装を行った.

** ここでいうキーワードとは, SQL 文中のテーブル名, ビュー表名, カーソル名のことである.

*** キーワードと分散ノードの属性データとの対応表.

表2　分散辞書情報

Table 2　The distributed information table

TABLE-NAME	FACIL-NAME	IDENTITY-OF-USER	REC-AE-NAME	REC-TPSU-TITLE	OPEN-MODE-NAME	CONTEXT-NAME
OKIAUTH-HOTELINTR	OKIDB1	OKINULU1	AE-OKI	2	MD-OKI	CN-OKI
OKIAUTH-HOTELRES	OKIDB1	OKINULU1	AE-OKI	2	MD-OKI	CN-OKI
OKIAUTH-HOTELRSV	OKIDB1	OKINULU1	AE-OKI	2	MD-OKI	CN-OKI
H3	OKIDB1	OKINULU1	AE-OKI	2	MD-OKI	CN-OKI
TOSAUTH-HOTELINTR	TOSDB1	TOSNULU1	AE-TOSHIBA	2	MD-TOSHIBA	CN-TOSHIBA
TOSAUTH-HOTELRES	TOSDB1	TOSNULU1	AE-TOSHIBA	2	MD-TOSHIBA	CN-TOSHIBA
TOSAUTH-HOTELRSV	TOSDB1	TOSNULU1	AE-TOSHIBA	2	MD-TOSHIBA	CN-TOSHIBA
H1	TOSDB1	TOSNULU1	AE-TOSHIBA	2	MD-TOSHIBA	CN-TOSHIBA
MITAUTH-HOTELINTR	MITDB1	MITNULU1	AE-MITSUBISHI	2	MD-MITSUBISHI	CN-MITSUBISHI
MITAUTH-HOTELRES	MITDB1	MITNULU1	AE-MITSUBISHI	2	MD-MITSUBISHI	CN-MITSUBISHI
MITAUTH-HOTELRSV	MITDB1	MITNULU1	AE-MITSUBISHI	2	MD-MITSUBISHI	CN-MITSUBISHI
H2	MITDB1	MITNULU1	AE-MITSUBISHI	2	MD-MITSUBISHI	CN-MITSUBISHI
NECAUTH-TRAININTR	NECDB1	NECNULU1	AE-NEC	2	MD-NEC	CN-NEC
NECAUTH-TRAINRES	NECDB1	NECNULU1	AE-NEC	2	MD-NEC	CN-NEC
NECAUTH-TRAINRSV	NECDB1	NECNULU1	AE-NEC	2	MD-NEC	CN-NEC
T1	NECDB1	NECNULU1	AE-NEC	2	MD-NEC	CN-NEC
HITAUTH-AIRLINEINTR	HITDB1	HITNULU1	AE-HITACHI	2	MD-HITACHI	CN-HITACHI
HITAUTH-AIRLINERES	HITDB1	HITNULU1	AE-HITACHI	2	MD-HITACHI	CN-HITACHI
HITAUTH-AIRLINERSV	HITDB1	HITNULU1	AE-HITACHI	2	MD-HITACHI	CN-HITACHI
A1	HITDB1	HITNULU1	AE-HITACHI	2	MD-HITACHI	CN-HITACHI
FJIAUTH-GOLFINTR	FJIDB1	FJINULU1	AE-FUJITSU	2	MD-FUJITSU	CN-FUJITSU
FJIAUTH-GOLFRES	FJIDB1	FJINULU1	AE-FUJITSU	2	MD-FUJITSU	CN-FUJITSU
FJIAUTH-GOLFRSV	FJIDB1	FJINULU1	AE-FUJITSU	2	MD-FUJITSU	CN-FUJITSU
G1	FJIDB1	FJINULU1	AE-FUJITSU	2	MD-FUJITSU	CN-FUJITSU
G2	FJIDB1	FJINULU1	AE-FUJITSU	2	MD-FUJITSU	CN-FUJITSU
G5	FJIDB1	FJINULU1	AE-FUJITSU	2	MD-FUJITSU	CN-FUJITSU
G7	FJIDB1	FJINULU1	AE-FUJITSU	2	MD-FUJITSU	CN-FUJITSU
G8	FJIDB1	FJINULU1	AE-FUJITSU	2	MD-FUJITSU	CN-FUJITSU
NULAUTH-GOLFINTR	NULDB1	NULOKIU1	AE-UNISYS	2	MD-UNISYS	CN-UNISYS
NULAUTH-GOLFRES	NULDB1	NULOKIU1	AE-UNISYS	2	MD-UNISYS	CN-UNISYS
NULAUTH-GOLFRSV	NULDB1	NULOKIU1	AE-UNISYS	2	MD-UNISYS	CN-UNISYS
G3	NULDB1	NULOKIU1	AE-UNISYS	2	MD-UNISYS	CN-UNISYS
G4	NULDB1	NULOKIU1	AE-UNISYS	2	MD-UNISYS	CN-UNISYS
G6	NULDB1	NULOKIU1	AE-UNISYS	2	MD-UNISYS	CN-UNISYS
G9	NULDB1	NULOKIU1	AE-UNISYS	2	MD-UNISYS	CN-UNISYS
G10	NULDB1	NULOKIU1	AE-UNISYS	2	MD-UNISYS	CN-UNISYS

36 TABLE-NAME RECORDS SELECTED
5 5　OUTPUT TO ORIGINATOR

TABLE-NAME：分散ノードに存在しているテーブル名，またはカーソル名
FACIL-NAME：分散ノードに存在しているデータベースを一意にするための名前
IDENTITY-OF-USER：分散ノードをアクセスするのに必要な認証パラメタ
REC-AE-NAME：
REC-TPSU-TITLE：　分散ノードの所在位置を示すパラメタ
OPEN-MODE-NAME：
CONTEXT-NAME：　TP の実行上必要なパラメタ．RDA としては意味を持たない．

ーカルノードであると判断する．

2)　クライアント・モジュール……クライアント・モジュールはローカル・システムと RDA 実装規約で定められたプロトコルとのインタフェースで，

①　クライアント・ユーザ（INTAP-DDB, UDSC (Universal Data System Control)）からの要求を，実装規約で定められたプロトコル・サービスに変換し実行する．

②　受信したサービスの実行結果を，ローカル・システムで扱える形式に変換し，クライアント・ユーザに返却する．

③　UDSC と RDAUIF の間に入り，2フェーズ・コミットメントの調整を行うという三つの役割を果たす．

ここで行われる変換は，クライアント・ユーザの要求から実装規約で定められたプロトコル・サービスへの変換，受信した SQL コードから RDMS のエラーコ

ードへの変換，ホスト変数名の変換*の3種類で，それ以外の変換(たとえばASN.1**への変換）はRDA/ROSEで行われる．

3) サーバ・モジュール……サーバ・モジュールは，ローカル・システムと実装規約で定められたプロトコルとのインタフェースで，

① 受信したプロトコル・サービスをローカル・システムで扱える形式に変換し，実行する．またその実行結果を実装規約で定められたプロトコルに変換し，クライアント側へ返信する．

② サーバ・ユーザ（サーバAP，UDSC）からの要求を，実装規約で定められたプロトコルに変換し，適当なプロトコル・サービスを要求する．

③ サーバAPとRDAUIFの間に入り，2フェーズ・コミットメントの調整を行う．

④ 条件式にホスト変数を使ったカーソルが宣言された場合，カーソル名と変数の属性（変数型とサイズ）の対応表を作成し管理する***．

という四つの役割を果たす．

ここで行われる変換は，送信されてきたプロトコル・サービスからローカル・システムが提供するサービスへの変換，SQLの文法からRDML指令****への変換，ホスト変数名の変換である．

4) サーバAP……サーバAPはサーバ・モジュールが生成したRDML指令を実行し，実際にデータベースのアクセスを行うモジュールである．またローカルDBMSから返されたエラー・コードのSQLコードへの変換も行う．

3.2.2 モジュール間のインタフェース

1) ユーザ・インタフェース……本RDAシステムにおいては，ユーザからのエントリ・ポイントはINTAP-DDBのみである．データベースに対する要求は，それがどこに存在しているかにかかわらず，まずINTAP-DDBに入る．また逆に，ユーザへのリターンもDDBを経由して行われる．このため本インタフェースではユーザ環境のセーブ/リストアを行う．

2) クライアント・インタフェース……INTAP-DDBによって，分散ノードへのアクセスと判断された場合，ユーザの要求をクライアントに渡すためのインタフェースである．このときINTAP-DDBは，クライアントが遠隔データベース・アクセスを行うのに必要な情報（遠隔ノードの属性データ，ユーザから与えられたSQL文やパラメタのパケットアドレス）を引き渡す．一方，遠隔データベース・アクセスが終了した場合は，このインタフェースを通してクライアントから結果が戻される．

3) RSAインタフェース（クライアント側）……INTAP-DDBによって，ローカル・ノードへのアクセスと判断された場合，RSAに制御を渡すためのインタフェ

---

* サーバへ送信されるSQL文のホスト変数名は，すべて':H'でなければならない．たとえば，SELECT * FROM TABLE1 WHERE COL1=VAR1 AND COL2=VAR2というSQL文を送信する場合は，SELECT * FROM TABLE1 WHERE COL1=:H AND COL2=:Hという形に変換しなければならない．

** Abstract Syntax Notation 1の略．値の内部表現形式が異なるシステム間で，データのやりとりを行うための表現形式．

*** カーソル宣言時にはホスト変数の属性のみが送信され，カーソルを開く時にはホスト変数の値のみが送信されてくるため，このような管理が必要となる．

**** SQLに準拠したリレーショナル・データベース操作言語

ースである．このときINTAP-DDBは，ACOB（ASCII COBOL）のENTER MASM*インタフェースと同じインタフェースでRSAに制御を渡す．

一方，ローカル・データベース・アクセスが完了した場合は，このインタフェースを通してRSAから結果が返される．

4) RDAUIFインタフェース（クライアント側）……クライアント・ユーザからの要求を，実装規約で定義されているサービス群に変換しRDAUIFにそれらを要求する．またRDAUIFから受け取ったサービスの実行結果をクライアント・ユーザに返却する．

5) トランザクション・コマンド・インタフェース（クライアント側）……複数ノード間での同期のとれたコミットメントおよびロールバックを実現するために，クライアントは自ノードのステージング**終了後，コミット処理終了後，およびロールバック完了時にサーバに対してステージング処理要求，コミット処理要求，およびロールバック処理要求/完了通知***を出さなくてはならない．UDSCは，ユーザ起動（応用プログラムからのCOMMITやROLLBACKの実行）およびシステム起動（UDS（Universal Data System）の内部的なスレッド・ロールバック）で発生したコミット要求あるいはロールバック要求を本インタフェースを通してクライアントに渡し，分散ノードに通知する．また，分散ノードにおけるコミット，ロールバックの結果を本インタフェースを通して受け取り，クライアント・ユーザに通知する．

6) RSAインタフェース（サーバ側）……サーバ・モジュールによって生成されたRDML指令を実行し，データベースをアクセスするためのインタフェースである．このインタフェースは通常のACOB/RSA間のインタフェースと同じものである．

7) RDAUIFインタフェース（サーバ側）……クライアント側から送信されてきたプロトコル・サービスをサーバ側で実行可能な形に変換し，その実行結果をクライアントへ返信するためのインタフェースである．多くの場合，送信されてきたサービスはRDML指令にマッピングされ，上述のRSAインタフェース（サーバ側）を通してデータベースをアクセスする．

8) トランザクション・コマンド・インタフェース（サーバ）……複数ノード間での同期のとれたコミットメントおよびロールバックを実現するために，サーバは自ノードのステージング終了後，コミット処理終了後，およびロールバック完了時にクライアントに対してステージング完了通知，コミット完了通知，およびロールバック処理要求/完了通知を出さなくてはならない．UDSCはシステム起動（UDSの内部的なスレッド・ロールバック）で発生したロールバック要求を本インタフェースを通してサーバ・モジュールに渡し，分散ノードに通知する．また，分散ノードにおけるコミット，ロールバックの結果を本インタフェースを通して受け取り，サーバ・ユーザに通知する．

---

* ACOBから他の言語への制御を渡す場合のインタフェース．

** 更新情報を保管するための処理．

*** 自ノード起動のロールバックの場合はロールバック処理要求を出し，他ノード起動の場合は完了通知を出す．

表3　RDML から R サービスへ/R サービスから RDML へのマッピング
Table 3 Mapping example of the RDML to RDA services/RDA services to RDML

＜RDML から R サービスへのマッピング＞
TAB1 はローカルノードに存在し，TAB2 は分散ノードに存在するものとする．

BEGIN THREAD FOR UDSSRC UPDATE(DEFERRED)；	
SELECT * FROM TAB1；	
DECLARE C1 CURSOR FOR SELECT * FROM TAB2；	TP-BeginDialogue-req R-BeginDialogue-req R-Open-req TP-BeginTransaction-req R-ExecuteDBL-req
FETCH C1；	R-ExecuteDBL-req
COMMIT；	TP-Commit-req TP-Done-req
END THREAD；	R-Close-req
RDA$TERM	R-EndDialogue-req

＜R サービスから RDML へのマッピング＞
RDML へマッピングされる R サービス/TP サービスは以下の通りである．

R-Open-ind	——→BEGIN THREAD
R-ExecuteDBL-ind	——→RDML COMMAND
TP-Commit-ind	——→COMMIT
TP-Rollback-ind	——→ROLLBACK
R-Close-ind	——→END THREAD

RDA$TERM は，RDA ダイアログを開放するためのプリミティブで，今回のデモのために新たに作成した．

実際にユーザの要求が，RDA サービスおよび TP サービスへどのようにマッピングされていくかについては表3に示す．

4. 標準プロダクト作成時の問題点

4.1 応用プログラムとのインタフェースに関する問題点

RDA サービスにはさまざまなサービス・パラメタが存在している．それらの値をどのように決定するかは，そのパラメタが必須であるか否かにかかわらず RDA を実装する上での重要な問題となる．ここでは，RDA サービスの中でも最も使用頻度が高いと考えられる即時実行型データベース言語サービス（R-ExecuteDBL）を例にとり，そのサービス・パラメタ値の決定方法に関し，現状の実装方法の是非を検討する．

R-ExecuteDBL のサービス・パラメタは以下の通りである（クライアントがデータベース操作を要求する時に設定しなければならないパラメタのみ）．

OperationID	：RDA サービスを一意にするための識別子
data ResourceHandle	：使用するデータベースを一意にするための識別子
SQLDBLStatement	：実行する SQL 文
SQLDBLArgumentSpecification	：引数の型と長さ

SQLDBLResultSpecification	：結果の型と長さ
dBLArguments	：引数の指定方法（singleArg/multiArg 選択）
・singleArgument	：SQL 文を同じ引数の値を用いて repetition
repetitionCount	Count で指定した回数繰り返して実行する
SQLDBLArgumentValues	引数の値
・multiArgument	：SQL 文を list of SQLDBLArgumentValues で指
list of SQLDBLArgumentValues	定した引数リストの個数回だけ，引き数の値を変
	えて実行する．

このうちとくに問題となるのは，singleArgument と multiArgument の選択基準，repetitionCount の決定方法の 2 点である．なぜなら，他のパラメタはユーザから渡されるか，RDA システムが一意となる値を与えるか，であるのに対し，この三つのパラメタは状況に応じて RDA が決定しなければならないパラメタであるからである．デモ・システムにおいては singleArgument のみ使用し，repetitionCount は，FETCH 文の場合は 100 回，それ以外の SQL 文の場合は 1 回，という決めうちの値を応用プログラムから指定するような実装を行った．しかし，これは応用プログラムの独立性を著しく損なう実装であり，標準プロダクトの実装方法としては採用できない．

これらのパラメタは RDA システムのパフォーマンスにかかわる重要なパラメタであり，標準プロダクトにおいては値決定のための明確なロジックが必要である．次章において具体的なロジック検討を行う．

4.2 RDA システムとしての問題点

ここでの問題は未実装の機能をどう扱うかが中心である．RDA サービスという観点でみると，制御サービスおよび蓄積実行型のデータベース言語サービスが未実装となっている．また，複数サーバ間にわたるデッドロックの検知と解消といった機能も実装されていない．ここでは各々の機能の実装の必要性を検討し，必要があると判断された機能に関してはその実装方法を 5 章において検討する．

1) 制御サービス……このサービスは，すでに要求ずみの RDA サービスをキャンセル（R-Cancel）する，あるいはその実行状況を問い合わせる（R-Status）ためのサービスである．このうち R-Cancel に関しては，キャンセルの対象となるサービスが，実行待ちとなっている特定の RDA サービスのみであるため，その有効性は疑問である．なぜなら，一般にユーザがキャンセルしたいものは個々の RDA サービスではなく，データベースに対して行ったなんらかの操作であり，これを実現するためには SQL でロールバックを要求すれば十分であるからである．一方の R-Status に関しても R-Cancel と同様の理由から（ユーザが知りたい情報は各々の RDA サービスのステータスではなく，自分が実行したデータベース処理のステータスであるはず），その有効性には疑問がある．強いていうなら，運用管理者が分散ノードの実行状況をモニタするために使用可能であるといえるかも知れないが，そうであるなら RDA サービスの実行状況だけではなくデータベース・システム内でのステータス（たとえばキュー中である等）もわからなければ機能が十分ではない．

INTAP においても，これらのサービスの必要性に関しては疑問の声が多く（とくにキャンセルの有効性を疑問視する声が多い），最新の実装規約では規定外とな

っている．したがって，標準プロダクトにおいてもこれらの機能を実装する必要はないと考える．

2) 蓄積実行 DBL サービス……このサービスは，あらかじめ任意の SQL 文をサーバ側に登録しておき(R-DefineDBL)，あとからその SQL 文の識別子と SQL の実行に必要な引数を送信し実行する(R-InvokeDBL)サービスで，通信量の削減(毎回 SQL 文を送信する必要がない)とサーバ側での効率改善(あらかじめサーバ側で SQL 文をプリ・コンパイルしておくことができる)を目的としている．しかしながら，これらのサービスには，

① 複数の SQL 文を一つの識別子のもとにプロシージャとして登録することができないため，応用性にかける，

② 現在使用中のデータベースの使用を終了すると(R-Close)，SQL 文の登録も解除されてしまうため本当に期待するような効率改善が実現できるかは疑問である，

③ その DBL を定義したのと同じダイアログからしか使用できない，

という問題点があり，制御サービス同様，INTAP においてもその有効性を疑問視する意見が少なくなく，最新の実装規約では規定外となっている．以上の理由から，標準プロダクトとしても蓄積実行 DBL サービスを実装する必要はないと考える．

RDA ラポータグループでは，現在の蓄積実行 DBL サービスの問題点を解決するために新しいコンセプトの蓄積実行 DBL サービスを検討中である．これは，すべての使用者から恒久的に使用可能な蓄積実行 DBL サービスを提供しようとするもので，このサービスが国際標準となれば十分実装する価値があると考える．

3) 複数サーバ間のデッドロックの検知と解消……一般にデータベースシステムにおいて，デッドロックの検知と解消は必須の機能であるが，これは RDA 環境においても同様である．とくに RDA システムでは，複数サーバ間にわたるデッドロック（グローバル・デッドロック）というローカル・データベースのみを扱うシステムでは存在しえなかったタイプのデッドロックが発生する．今回のデモ・システムでは，

① クライアント側でタイマ管理を行い，一定時間待っても応答がなかった場合にはデッドロックとみなし，ダイアログの強制終了要求を行う．その後，サーバからロールバック完了通知が来たあとで*ユーザに制御を返す，

② サーバ側ではデッドロックのための処理をとくに行わない，

という実装を行った．しかしながらこの実装には，デッドロックの解消が完全にはできないという致命的な欠陥がある．なぜなら，今回のデモ・システムのような実装をしたクライアント/サーバ間でデッドロックが発生した場合，サーバはデータベースの中で待ち続けているため，クライアント側からのダイアログ強制開放要求を検知することができず，したがってロールバック完了通知を返すことが

* 実装規約上，トランザクション中にダイアログを強制終了させた場合は，サーバからのロールバック完了通知を受信しなければならない．

できないからである．この問題の解決は必須であり，5章においてその解決策を検討する．

4.3 SQLに関する問題点

この分野に属する問題は以下のように大別できる．

1) 機能上の問題……RDMSでは提供しているがSQLでは提供していないような文法，あるいはその逆の文法をどう扱うか．この問題は，RDAで比較的簡単に対処できる問題と，そうでない問題とがある．前者に属する問題としては，OPEN CURSOR時の文法の違い*やRDML指令の最後にセミコロン(；)が必要な点があげられ，後者の問題としてはRDMSとして未実装な機能（例 副問い合わせ）や逆にRDMSにのみ存在する機能（例 LOCK指令）があげられる．

2) SQLコードの問題……DELETE文またはUPDATE文の処理において，処理の対象となるレコードが存在しなかった場合，SQLではSQLコードに+100を返すが**，RDMSではエラーとはせず，補助情報に処理対象件数を返す(すなわちRDMSでは補助情報が0の場合，処理対象行が存在しないということになる)．この違いをどう解決するか．

3) 補助情報の問題……2) の問題にも関連するが，SQLではDELETE文やUPDATE文を実行した時に応用プログラムに処理対象件数を返却するようなインタフェースはない．この問題をどう解決するか．

4) コード系の問題……文字型のカラムを用いてソートを行ったり，値の大小の比較を行った場合，ローカル・システムがどのようなコード系の実装を行っているかによって結果が異なってくるが，これをどう解決するか．

5) SELECT文の扱いの違い……複数の行が選択対象となった場合，SQLではSQLコードに+100を返すが，RDMSでは条件に合う行を1行だけ選択し，エラーとはしない．この違いをどう解決するか．

以上の問題は解決することが必須の問題ばかりであるが，本稿で解決策を検討するには対象範囲が広すぎる．なぜならこれはRDAだけの問題ではなく，RDMS 1100およびSQL国際標準にも関係する問題であるからである．したがって本稿では，とくに問題になりそうなSQLとRDMSの機能の相違に関して、問題解決の方向性を検討することとし，本質的な解決策の検討は行わない．

5. 解決策の検討

5.1 アーグメント・タイプ選択

本節では，アーグメント・タイプ（singleArgumentとmultiArgument）の選択と繰り返し回数（repetitionCount）の決定方法について検討を行うが，具体的な解決策を検討する前に，まず各アーグメント・タイプの特徴を明確にしておく必要がある．

まずsingleArgumentであるが，このアーグメント・タイプは一つのSQL文とそのSQL文で使用される引数を1セットだけ送信することができる（図3)．したがって，

---

* カーソル宣言にホスト変数が用いられている場合，RDMSではカーソルを開く時にもホスト変数を指定する必要があるがSQLの場合は指定しない．

** “処理対象行なし”を示すコード．

1回だけ SQL 文を実行するような場合や，同一の引数を用いて SQL 文を繰り返し実行するような場合，あるいは引数なしの SQL 文を繰り返し実行する場合に用いられるべきである．一方 multiArgument は，一つの SQL 文とその SQL 文で使用される引数の複数のセットを送信することができる（図 4）．したがってこのアーグメント・タイプは，異なるデータを大量に挿入したり削除したりする場合に使用されるべきである．

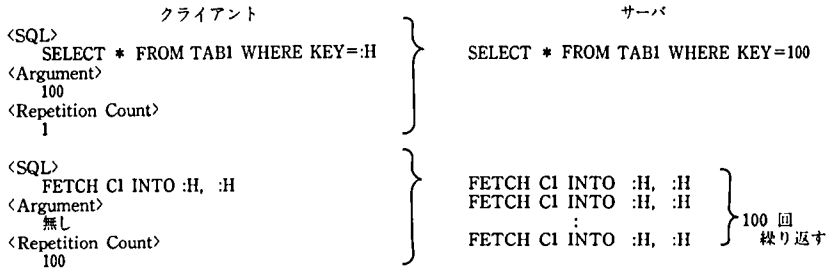


図 3　singleArgument の使用例

Fig. 3　Example of the single argument

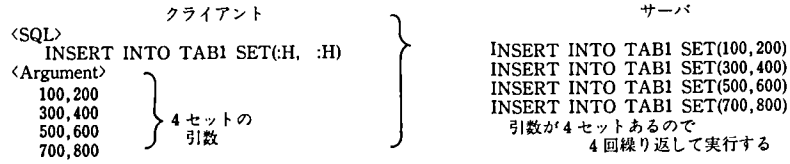


図 4　multiArgument の使用例

Fig. 4　Example of the multiple argument

各アーグメント・タイプの使用方針は上述のようにするとしても，どのようなユーザ・インタフェースでこの二つのアーグメント・タイプを使い分けるかという問題が残る．応用プログラムの独立性を最大限に保証することを考えれば，応用プログラムと RDA システムとの間のインタフェースは SQL 文のみとしなければならない．応用プログラムから明示的にアーグメント・タイプを指定するような実装は避けるべきである．しかしそのような前提に立った場合，SQL 文と multiArgument をどのように対応させるかが問題となる．なぜならこのアーグメント・タイプは一つの SQL 文を引数をかえて複数回実行するものなので，SQL 文から直接マッピングすることはできないからである．したがってこのアーグメント・タイプは SQL 文から直接マッピングするのではなく，たとえば大量データロード用のユティリティ（RDMSLOAD 等）から使用されるべきと考える．

一方 singleArgument 使用時の繰り返し回数であるが，このパラメタの設定に関しては二つの問題がある．すなわち，どの SQL 文に対して 2 以上の値を与えるか，2 以上の値を設定する場合，具体的にどのような値を設定するべきか，という二点である．まず前者の問題であるが，一般に同一の SQL 文を複数回繰り返して意味があるのは

FETCH 文のみであり，標準プロダクトにおいても繰り返し回数に 2 以上の値を与えるのは FETCH 文のみでよいと考える．また後者の問題については，このパラメタが効率に影響を与えるパラメタであることから可能な限り大きな値を設定すべきである．具体的には，1 回に転送できる最大サイズ*を 1 タプルの転送に必要なサイズで割った結果を繰り返し回数として設定すべきと考える．以上の検討をまとめると，以下のようになる．

〈アーグメント・タイプの選択方針〉

① multiArgument は，RDMSLOAD のような大量データロード用のユティリティから使用し，一般の応用プログラムからは singleArgument のみを使用する．

② singleArgument の繰り返し回数は，FETCH 文の場合のみ 2 以上の値を設定することとし，その値は 1 回に転送できる最大サイズを，1 タプルの転送に必要なサイズで割って得られた値とする．FETCH 以外の SQL 文の場合は繰り返し回数として 1 を設定することとする．

5.2 グローバル・デッドロック

ここでは，グローバル・デッドロックの解決方法について検討を行う．デモ・システムの実装で問題となる点は，サーバ側でクライアントからのダイアログ強制終了要求が受け取れないことにある．これはあるランが UDS の中でキューに入ってしまうと誰かに起こされるまではずっとキューに入りっぱなしになってしまうことによるものである．したがって，ある一定時間キューしていてもロックが成功しない場合，ロック不成功というステータスとともに RDA に制御が戻ってくれば，そのときクライアントからの要求を受け付けられるため，この問題を解決することができる．これを実現するためには，

1) リカバリ・モードをコマンド・ルックスとし，かつアクセスする表に対し ON CONFLICT RETURN を指定したロックを明示的にかける**，

2) キュー中のランを監視する常駐ランを作り，UDS の中で一定時間以上キューしているランを強制的に起こすようにする，

という方法が考えられるが，1)の方法ではコミット時に更新情報を保存することができず，2 フェーズ・コミットメントが行えないと言う問題点がある．一方，2)の方法であるが，これは UDS 5 R 2 で導入された FCSS ファイルと UDS 間のデッド・ロック検出方法である UDS タイマの機能と同一であり，この標準機能を応用することにより RDA においてグローバル・デッドロックの検知と解消が可能と考える．

5.3 SQL と RDMS の機能の相違

ここでは，SQL と RDMS の機能の相違についての検討を行う．ただしこの問題は，他の二つの問題と異なり問題の範囲が広範囲にわたるため，問題の具体的解決策を検討するのではなく，問題解決の方向性を探ることを目的としたい，

SQL と RDMS の機能の相違は，大きく分けると，

① 単純な文法上の相違，

---

* OSI のセッション層で扱える最大データサイズ．

** これによりロックが成功しなかった場合，キューせずにユーザへリターンするようになる．

② RDMS では提供しているが SQL では提供していない機能の扱い，

③ SQL では提供しているが RDMS では提供していない機能の扱い，

の 3 タイプに分類できる*．このうちとくに問題となるのは②である．なぜなら①の問題は単純な文法上の変換を行えば対処できるし，③の問題は本質的には RDA の問題ではなく DBMS の問題だからである（③に関しては最悪文法エラーとすることもできる）．それに対し②の問題は RDA 以外には対処する場所が存在せず，標準プロダクトを作成する場合にはこの問題をどのように扱うかについて明確な指針が必要となる．

②の問題は，クライアント側で発生する問題とサーバ側で発生する問題の二つに分類できる．前者は，クライアント側の応用プログラムから RDMS 固有の指令が要求された場合の問題であるが，この問題は RDA で完全に解決することは不可能である．なぜなら RDMS は SQL では提供していない機能を提供しており，それらのすべてを SQL のみ提供している DBMS で実行することは不可能であるからである．この問題に対する唯一の解決策は応用プログラム側で RDMS 固有の指令を使用しないようにすることであるが，これは既存のプログラムの書換が必要となることを意味し，その実現可能性は疑わしい．せいぜい新規開発の応用プログラムは，SQL のみで記述するように指導することが実現可能な解決策といえる．ただし RDMS 固有の指令のうち SQL に置換可能な指令を洗い出し，その置換方法を検討することは将来課題として意味があると考える．一方後者は RDMS の運用上必要となる RDMS 固有の指令の扱いの問題であり，その代表的なものはロックの問題である．RDMS ではデッドロック回避の手段や効率改善の手段として表を明示的にロックすることを勧めている．しかしながら SQL には RDMS の LOCK 指令に該当する命令はなく，クライアント側から明示的にロックが要求されることはない．したがって，クライアント側からの要求に応じてではなく，サーバ側の判断によりこれらの指令を実行する必要がある．たとえば，すべてのタプルを削除するような要求がきた場合（たとえば，条件なしの DELETE 要求が送信されてきた場合），サーバ側で該当する表を明示的にロックする等の仕組みが必要となる．この場合，ロックを行う条件とロックを解除するタイミングをプロダクト作成の方針として明確にしておく必要がある．

〈RDMS と SQL の機能の差異に対する対処方針〉

① 単純な文法上の違いは RDA で変換する．

② SQL でのみ提供されている機能に関しては，RDA では対処しない．

③ RDMS でのみ提供されている機能に関しては，運用上必要とされる LOCK 指令の対処法をまず検討する．それ以外のものに関しては，対処が必要なものの洗い出しを行う．

6. おわりに

以上 INE '91 における RDA の実装とその実装を通して得られた問題点，およびそ

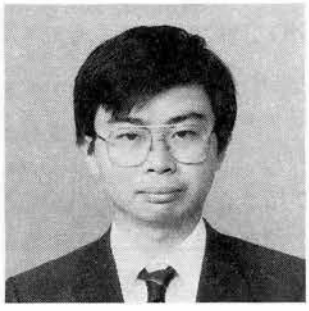
* 今回のデモでは，2，3 の相違に該当するような SQL 文は使用されなかったため 1 の相違のみ対応した．

の解決方法を見てきたが，ここで触れた問題はRDAシステムを実装する時に解決しなければならない問題のほんの一部にすぎない．実際には，ここで検討した以上の問題が存在している．たとえば，分散ノード情報の管理方法やリカバリに代表される運用面の問題は，RDAを商品として提供する前に解決することが必須の問題である．また分散環境において，応用プログラムの独立性をどのように保つべきかという問題も同様に重要である．これは各社がローカルに提供しているDBMSの機能をRDAの世界でどのように活かしていくべきか，すなわち分散不可視を保ちつついかに非標準機能を使用可能とするかという問題であり，これからのプロダクトの提供方針にもかかわる重要な問題といえる．

RDAは歩き始めたばかりの若い技術であり，まだまだ不備な点も多くある．計画通り1992年に国際標準となっても，しばらくの間は機能拡張等の作業が続くであろう．また，現在の規格はプロダクト側の視点から作成された規格であるため，利用者側から見たとき必ずしも使いやすい規格となっていない点もある．たとえば蓄積実行DBLがよい例で，このサービスは利用者側からの意見により，より強力なサービスに置き換えられようとしている．これからは，利用者により近い立場から現基本標準のブラッシュ・アップが必要となってくるであろう．

参考文献 ［1］ Information technology-Open Systems Interconnection-Remote Database Access, Part 1: Generic model, service and protocol, ISO/IEC DIS 9579-1.
［2］ Information technology-Open Systems Interconnection-Remote Database Access, Part 2: SQL Specialization, ISO/IEC DIS 9579-2.
［3］ INE '91 OSI相互接続実験'91ガイドブック INTAP.
［4］ INE '91 RDAonTP実験仕様書第2版，INTAP RDAonTP実証評価WG.

執筆者紹介 坂谷祥史（Yoshifumi Sakatani）
1964年生．1986年慶応義塾大学商学部卒業．同年日本ユニシス入社．主に2200・1100シリーズのデータベース・ソフトウェアの保守・開発業務に従事．現在システム技術本部 データベースソフトウェア部 データベース二課所属．SC 21/WG 3 RDA SG委員会委員，INTAP第一専門委員会委員，データ管理調査研究委員会WG 2（RDA）委員．



島村隆一（Ryuichi Shimamura）
1948年生．1972年日本大学文理学部卒業．1973年日本ユニシス入社．INFORMの開発・保守，2200・1100シリーズのデータベース・ソフトウェア全般の業務に従事．現在システム技術本部 データベースソフトウェア部 データベース二課課長．INTAP RDAonTP実証評価委員会委員．

システム開発環境IDESのリポジトリ

The Repository of the Integrated Development Environment Support System (IDES)

板　倉　　教

要　約　システム開発環境IDES (Integrated Development Environment support System) は，システム開発の生産性と品質向上を目的とする開発支援システムである．

本稿ではIDESの基盤となっているリポジトリに焦点をあて，そのデータモデルの考え方，機能，実現およびリポジトリに保存されるIDESデータの構造について説明する．

このリポジトリのデータモデルは実体関連型モデルを基本に，最近のオブジェクト指向データベースでのクラスの階層化，属性の継承機能をも実現している．これにより，複雑なデータ構造が扱える拡張性の高いリポジトリとなっている．

IDESではこのリポジトリ機能を利用し，システム設計データを構造化して蓄積，整理しておき，開発作業支援のために利用している．開発作業を支援するツールも透過性の高い構造化されたリポジトリデータを入力とするため，その実現が簡明なものになった．

Abstract　The Integrated Development Environment support System (IDES) is a software development support system designed to help increase the productivity and quality of systems development efforts. Addressing the repository, the base of IDES, this report gives an account of how its data model is built, what functions it provides, what is to be done for implementation, and how IDES data stored in the repository are structured.

Based on the entity relationship model for its data model, the repository of IDES has made available such functions as hierarchical classification and attributive inheritance as seen in the object-oriented database in recent years, thus making it capable of manipulating more complex data structure, and making it higher in expandability as well as more flexible in the changing of the data schema during systems designing and implementation.

With the help of its repository functionality, IDES serves to develop systems by using systems design data already structured, accumulated and laid out in order. The entry, as input, of structured repository data of high transparency also makes it easier to create all sorts of tools which support systems development.

1. はじめに

システム開発環境IDES (Integrated Development Environment support System) は，システムのライフサイクル全般にわたる開発の生産性と品質向上を目的とする開発支援システムである．この支援システムは，システム開発の一連の開発工程(分析，設計，プログラミング，テスト，保守) で開発作業を支援する新しい環境と支援ツールを用意し，従前と異なった便利な開発環境を提供することを目的とする．

IDESでは，開発のそれぞれの工程で発生するいろいろな定義情報をリポジトリとよぶデータベースに蓄積・保存し，開発データの信頼性を高めるとともに，後続の工程では蓄積データを再利用したり，蓄積データを基にした支援ツールを提供してシス

テム開発の効率化に寄与している．システムが扱うデータの定義，システムを構成するソフトウェアの定義，システム開発に携わる要員の情報，システム開発の状態等，開発に関する情報を一元的にリポジトリに記録する．リポジトリで一元管理されるデータが全工程におけるツールの基礎データとなり，同時に出力データとしても活用される．

図1はIDESの各工程での支援ツールとそれらを結合するリポジトリとの関連を示している．

IDESの一つの基盤ソフトとなっているこのリポジトリについて，そのデータモデル，機能，実現およびIDESで蓄積するデータのスキーマについて報告するのが本稿の目的である．

IDESでリポジトリに蓄積するデータの種類は次のようなものである．

1) 開発システムのファイル情報，レコード情報，項目情報

2) システムの構成情報（システム，手続き，ランストリーム，プログラム，モジュール，登録集）

3) その他

これらの情報は，それぞれの個体情報と個体間のいろいろな関連情報からなっている．IDESではこれらの実世界の実体と実体構造，およびそれらの関連をそのままリポジトリに反映できるリポジトリが望ましかった．

CODASYL型のデータベースではその拡張性，リレーショナルデータベースではデータモデルの単純性が問題であった．すなわちIDESの複雑なデータはこれらには過重であった．実体関連型モデル[1]は，実世界を実体と実体間の関連によって表現しようとするためのデータモデルである．

このデータモデルは，データが相互に関連するIDESのデータによく一致するものであった．しかし，このモデルでは一つの実体型の実体群をさらに複数の実体群に類別表現したい時は，それぞれを別々の実体型とするか，無理に一つの実体型にしなければならない．別々の実体型に分けると，より抽象度の高い実体に対する扱いが簡単にはできず，逆に一つにまとめると，特定の型のものに対する操作が不便になる．

同様に実体の属性についても，実体を分けた時は，共通に持つ属性を個々の実体型に重複して定義する必要があり，逆に一つの実体型にまとめると，特定の実体に不要な属性も混在して定義しなくてはならなくなる．このように，二次分類，三次分類可能な場合でも，実体属性を構造化して定義することができず冗長なモデルを定義しなければならなくなる．

そのためIDESでは実体関連型モデルを拡張して，より構造化したデータモデルを採用した．それは最近のオブジェクト指向データベース等での，クラスの階層化と属性の継承機能を取り入れたことである．すなわち，実体関連モデルでの実体型を木構造化し，階層表現可能にした．さらに，属性も階層ごとに定義できるようにし，上位から下位への継承機能を付加した．

これによって，IDESの持つデータをより素直に反映したリポジトリができ，データのスキーマも透明度の高いものになった．以下ではこのリポジトリのデータモデル，機能と実現，さらにIDESのデータのスキーマについて説明する．

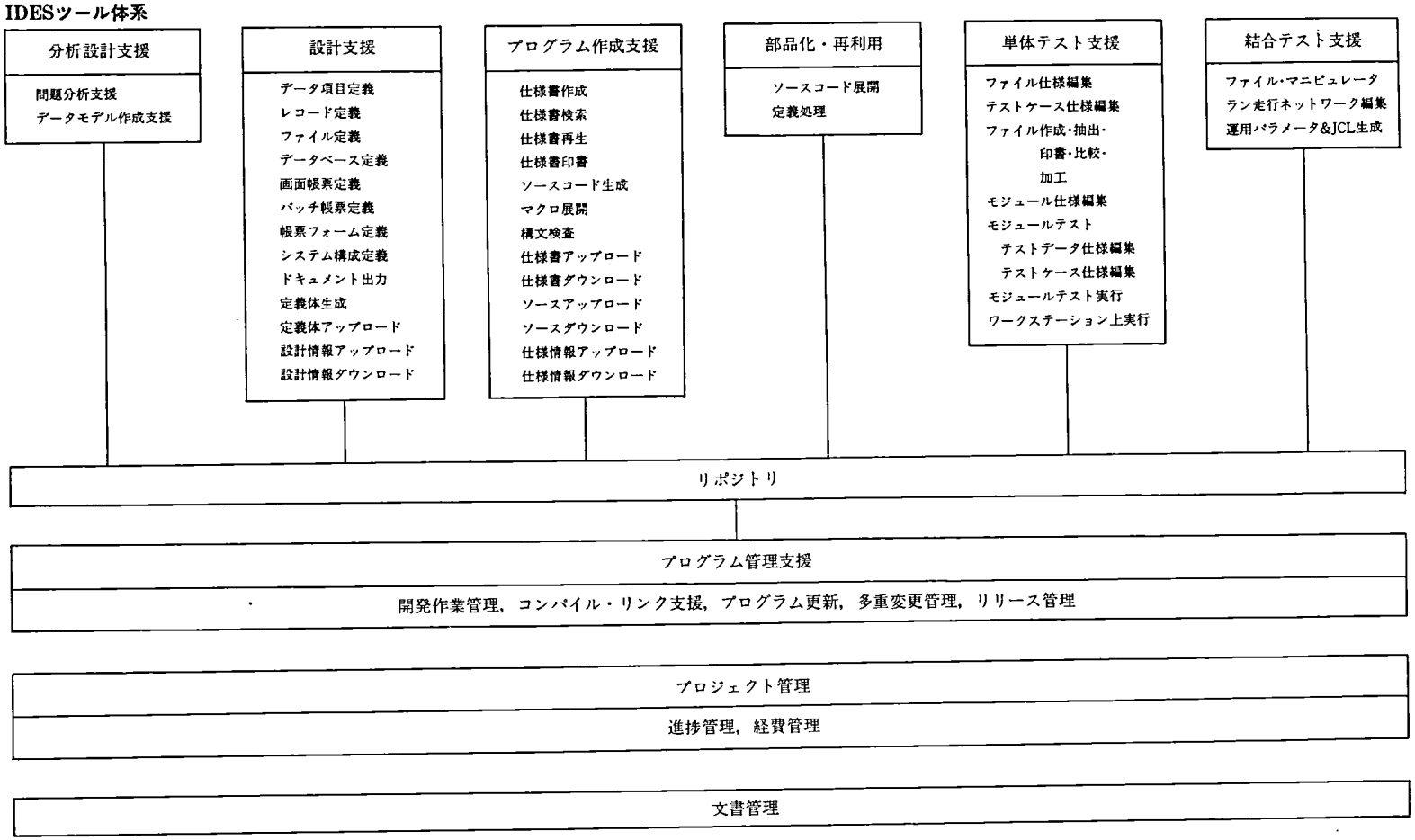


図 1　IDES ツールと IDES リポジトリ

Fig. 1　IDES tools and its repository

2. IDES リポジトリのデータモデル

2.1 IDES リポジトリの基本的な考え方

IDES リポジトリの基本的な考え方は次のようになる.

1) リポジトリはアプリケーションが対象にする実世界の実体（インスタンスと呼ぶ）の集まりである.

2) インスタンスはその種類ごとにクラスに類別できる.

3) 一つのクラスはさらにより詳細な特性に着目したクラスに再類別できる.

4) クラスは他のクラスと関係があるとき関連によって関係づけられる.

5) クラスやインスタンスはその特性を示す属性を持つ.

利用者がリポジトリ上に実世界を表現する論理的なデータビュをリポジトリスキーマというが，以下ではこのスキーマについて説明する.

2.2 クラスとインスタンス

特定の適用業務の中で関心の対象となっているオブジェクトを共通の性質によって類別し，類別されたそれぞれの集まりを「クラス」という．ソフトウェア資産をリポジトリで管理しソフトウェアの保守に役立てるような場合では，プログラムや登録集，ロードモジュール等がオブジェクトとなる.

それぞれのオブジェクトを共通の性質で類別すると，プログラムクラス，登録集クラス，ロードモジュールクラスに分けられる．そして類別したそれぞれのクラスのオブジェクトをクラスの「インスタンス」という．1本1本のプログラムはプログラムクラスのインスタンスであり，同じく一つ一つの登録集は登録集クラスのインスタンスである.

また，あるクラス内のインスタンスはそのクラス内で他のインスタンスと識別可能な識別子を持っており，その値を「インスタンス名」と呼ぶ.

プログラムクラス，登録集クラス，ロードモジュールクラスは図2のように表す.

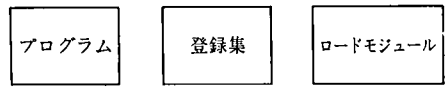


図 2 クラスの表現

Fig. 2 Representation of classes

2.3 クラス属性

「クラス属性」は，あるクラスそのもの，あるいはそのクラスのすべてのインスタンスについて述べる属性である.「プログラム」クラスでいえば「プログラム本数」等である.

2.4 インスタンス属性

クラスに参加するインスタンスは共通の特性項目を持っており，それらを説明する共通の性質をインスタンスの属性という．図2のプログラム保守を考えた時の属性としては，たとえば次のようなものが挙げられる.

プログラム
- 言語
- 行数
- エントリポイント
- 引数
- プログラムテキスト格納ファイル名
- プログラムテキスト更新時間
- オブジェクトモジュール名
- オブジェクトモジュール格納ファイル名
- オブジェクトモジュール作成時間
- .....

2.5 クラスの階層化—汎化関係—

リポジトリで対象とするインスタンスをクラスに類別したが，さらにそれぞれのクラスのインスタンスをいくつかの集合に再類別することができる．この集合に対してもクラスを定義できる．図2のプログラムクラスは，もうすこし詳細な管理のためにプログラムクラスをさらに類別し，プログラムクラスを主プログラムクラスと副プログラムクラスに分け，図3のようにかく．

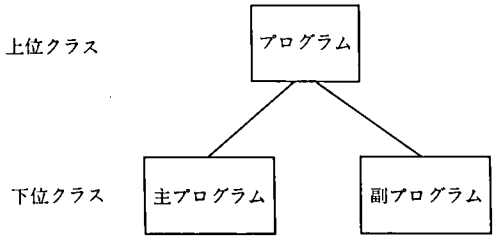


図 3　クラスの階層化
Fig. 3　Class hierarchy

主プログラムクラスと副プログラムクラスをプログラムクラスの「下位クラス」，プログラムクラスを主プログラムクラスと副プログラムクラスの「上位クラス」と呼ぶ．

図4は「受験生」を対象にしたその例である．

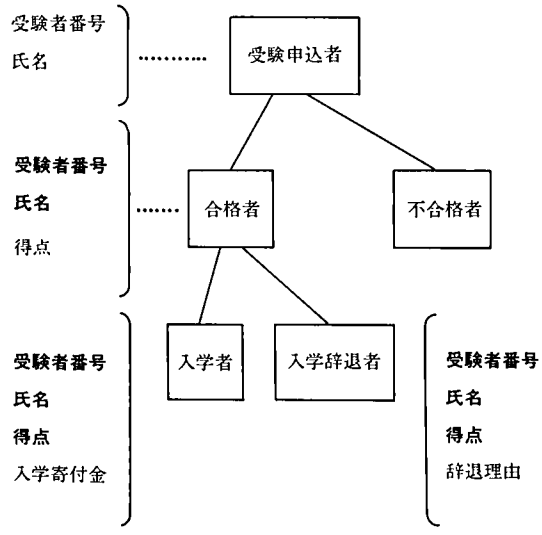


図 4　受験生のクラス
Fig. 4　Class of examinees

図3と図4の下位クラスは上位クラスをより具体化ないし特化したものである．下位クラスのインスタンスは，上位クラスと自分自身の両方の属性を併せ持つ．図4で「入学者」は「合格者」であり「受験申込者」でもあるので，「合格者」の属性と，さらに「受験申込者」の属性とを持っている．下位クラスのインスタンスの属性は，上位クラスから引き継いだものと下位クラス固有のものからなる．このように木構造の根から葉に向かって属性が引き継がれていくことを「属性の継承」と呼ぶ．これを逆に木構造の葉から根の方向に見ると，上位クラスは下位クラスをより抽象化あるいは一般化したものなので汎化関係と呼ぶ．図4で太字で示した属性は継承した属性である．

2.6 クラス関連

二つのクラスのインスタンス相互間に関係があるとき，その関係を次のように表現する．

関連名<オーナクラス名，メンバクラス名>

ここで関連名は，オーナクラスのインスタンスとメンバクラスのインスタンス間の関係につけた名前である．図5でのプログラムと登録集の関係は

COPY<プログラム，登録集>　COPY：プログラムがコピーする登録集

と表現できる．これは二つのクラス間の関係を満たすインスタンス対の全体を示しており，「クラス関連」とよぶ．図5ではさらにプログラムがCALLするプログラムという関係も表現している．

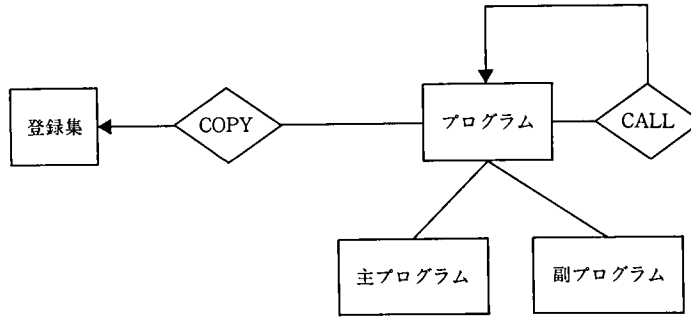


図5　プログラムと登録集のCOPY関連

Fig. 5　COPY relation between program class and copy library class

スキーマでは関連名をひし形で囲み，図の矢印の根のクラスをオーナクラス，矢先のクラスをメンバクラスとよぶ．

2.7 インスタンス関連

クラス関連が定義されると，そのオーナクラスのインスタンスとメンバクラスのインスタンス間に関連を持つことができる．これを「インスタンス関連」という．この関連はN：Mの関連である．指定したオーナクラスまたはメンバクラスが上位のクラスならば，指定したクラスおよびそのクラスを根とする部分木にあるインスタンスもインスタンス関連を持つことが可能である．

スキーマが図6のようであると，下記のようなインスタンス関連を設定でき，次の

ようにかく．

関連名<オーナクラス名，オーナインスタンス名，メンバクラス名，メンバインスタンス名>

COPY<プログラム，program 1，登録集，recordpdp 1>
COPY<プログラム，program 1，登録集，tablepdp 1>
COPY<プログラム，program 2，登録集，recordpdp 1>
COPY<プログラム，subprog 1，登録集，screenpdp 1>
COPY<プログラム，subprog 1，登録集，tablepdp 1>

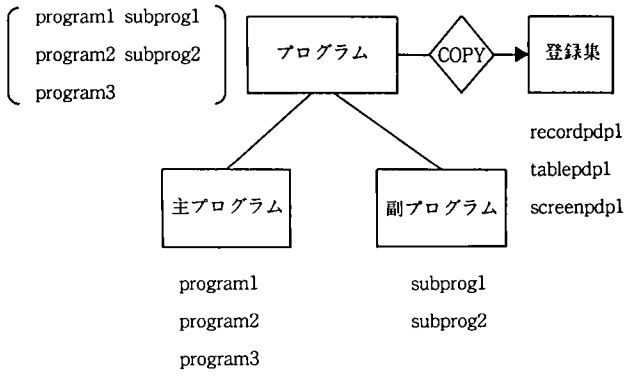


図 6　インスタンス関連の例

Fig. 6　Example of instance relationship

2.8　クラス関連属性とインスタンス関連属性

「クラス関連属性」は，ある特定のクラス関連そのもの，あるいはそのクラス関連の実現である全インスタンス関連について記述される内容である．たとえば「インスタンス関連の数」が考えられる．

「インスタンス関連属性」は個々のインスタンス関連が持つ特性項目である．図5の「CALL」のインスタンス関連でいえば，副プログラムが複数の入り口名を持つとき，CALLしている入り口名をインスタンス関連の属性とすることができる．

2.9　ビュークラス

利用者がリポジトリを更新するとき，対象とするクラスを指定しなければならない．「関連」の時はオーナ，メンバ両クラスを指定する．対象クラスを指定すると，利用者

表1　ビューと可視インスタンス属性

Table 1　Views and visible instance attributes

ビュークラス	更新可能属性値
受験申込者	受験者番号，氏名
合格者	受験者番号，氏名，得点
不合格者	受験者番号，氏名，得点
入学者	受験者番号，氏名，得点，入学寄付金
入学辞退者	受験者番号，氏名，得点，辞退理由

が更新できるデータの領域もそれに応じて決定される．利用者が指定した対象クラスを「ビュークラス」とよぶ．

図4で属性値を例にしてビュークラスと更新可能な属性値を表にして見ると表1のようになる．

3. リポジトリのセキュリティ

リポジトリを利用できる人は，登録されたものだけに限定される．また登録された利用者ごとに利用の資格（データベース管理者，データ管理者，データ利用者）が与えられ，利用の資格によって利用範囲，データの利用形態が制限される．

4. リポジトリの利用の仕方

リポジトリのスキーマを定義したり，データを蓄積・更新・検索したりするために，辞書アクセスルーチンを利用する．このルーチンを「プリミティブ」とよぶ．利用者は利用者プログラムからプリミティブを呼び出してリポジトリをアクセスする（図7）．

5. 実　　現

リポジトリはDMS 1100 (Data Management System 1100) を利用して実現されている．UDS/DMSを使うのはやはりデータの堅牢さ，保全性やリカバリを意識した結果である．この構造はリポジトリスキーマを表現した汎用DMSスキーマを用意することで実現している．利用者のリポジトリスキーマは，インタフェースルーチンによ

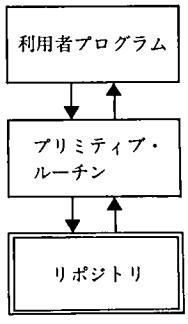


図7　リポジトリのアクセス

Fig. 7　Access of repository by primitive-routine

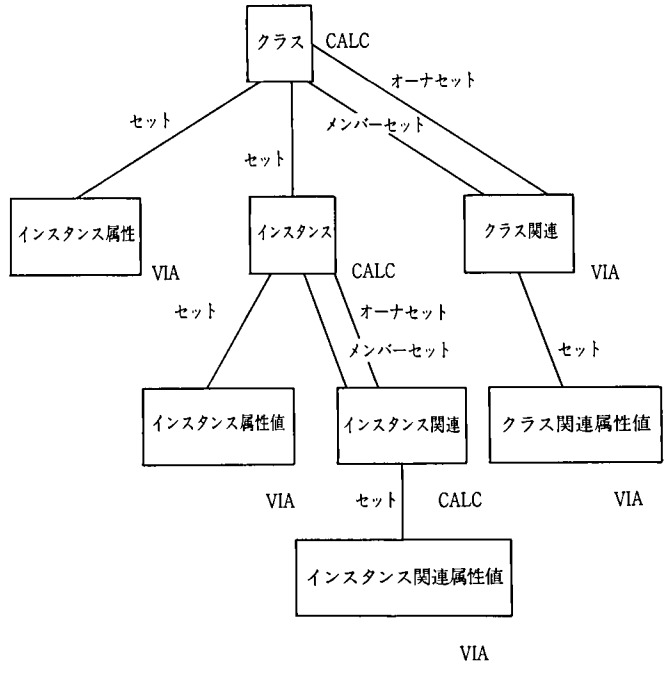


図8　汎用DMSスキーマ（一部）

Fig. 8　A part of DMS schema for IDES

ってDMSスキーマに変換され，実際のデータがアクセスされる．汎用DMSスキーマの骨子を図8に示す．ただしここではクラス，クラス関連，クラス関連属性値，インスタンス，インスタンス属性値，インスタンス関連，インスタンス関連属性値についてのみを選択してその概要を紹介する．

6. IDES用スキーマ

IDESリポジトリデータのビューは図9の通りである．点線で囲まれたクラスは階層化されており，ファイルクラスとレコード/項目クラスは図10のようになっている．

ここで重要なのは，ファイル，レコード等が階層化表現されることによってそのクラスの内容が一段と精密に表現でき，さらに利用者のビューに近づくことである．

属性定義においても共通属性は上位クラスで定義するだけで，下位クラス固有の属性は下位クラスにだけ定義すればよいので，冗長な属性定義をしなくても済み，より自然なものとなっている．

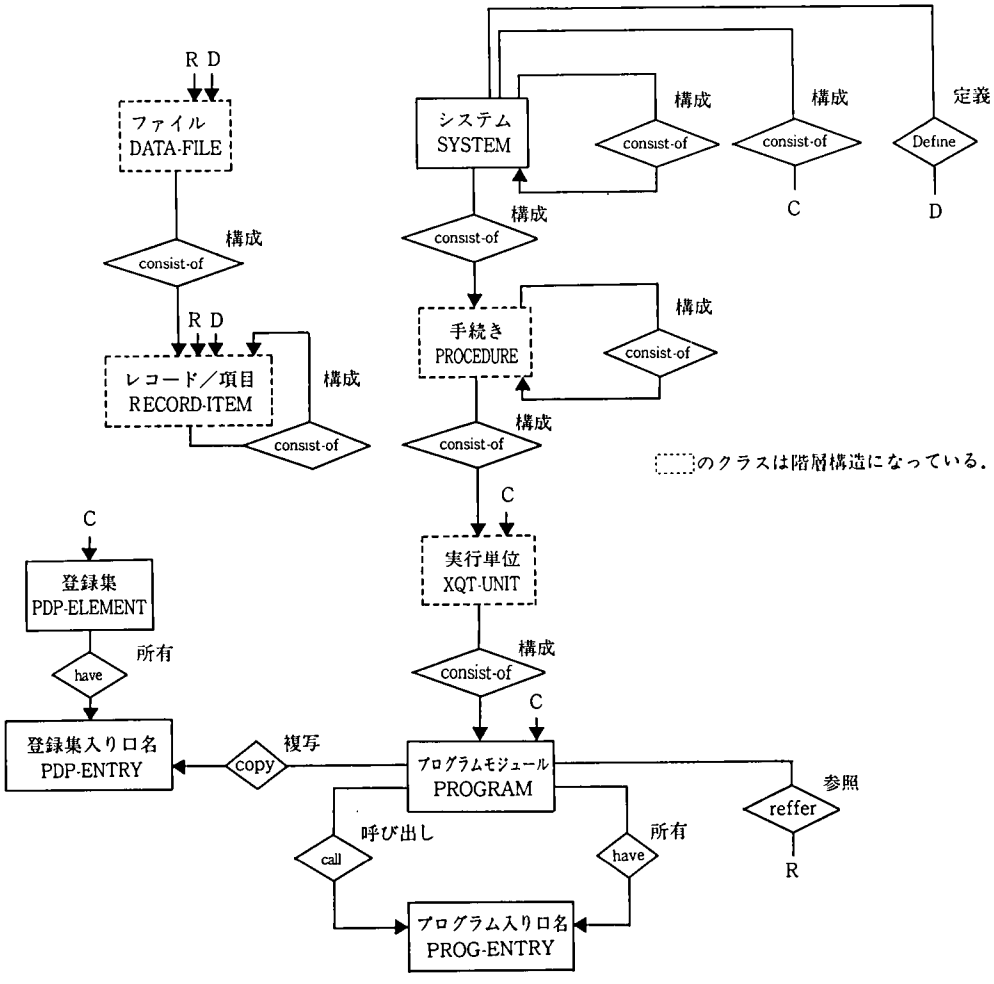
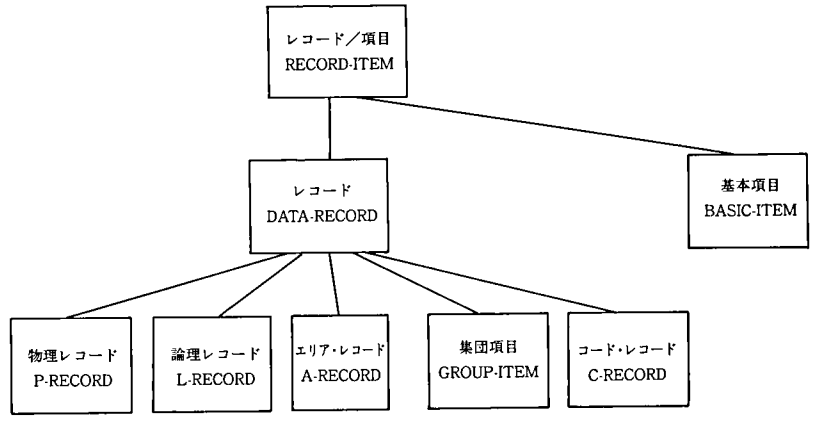


図9　IDESでのリポジトリスキーマ

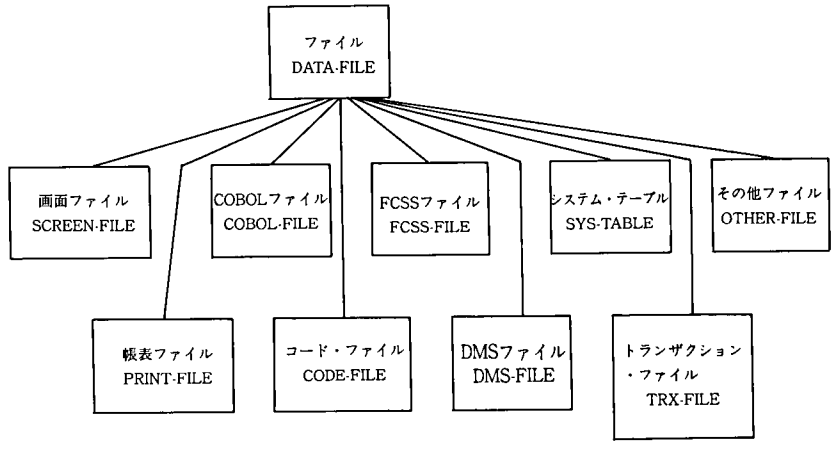
Fig. 9　Schema of IDES repository

ファイルの構成レコードと基本項目



(a)

ファイル型の定義情報クラス



(b)

図 10　ファイルとレコードのクラス階層

Fig. 10　Class hierarchy of file and record

7. おわりに

システム開発やソフトウェアの保守管理データはその種類が多く，またデータが相互に複雑に関係しているのが特徴である．そのようなデータをリポジトリとして自然な形で表現できるデータモデルが必要である．過去のいくつかのデータモデルは計算機指向が強かったが，最近は利用者指向になってきている．その意味でIDESリポジトリのデータモデルが，より利用者指向になったことが実際に開発に適用して，知ることができた．

参考文献　[1] CHEN, P. P. S. The entity-relationship model: Toward a unified view of data. ACM Trans. Database Syst. 1,1 (March 1976), 9-36.

執筆者紹介　板　倉　　教 (Satoshi Itakura)

昭和22年生．45年小樽商科大学商学部経済学科卒業．46年日本ユニバック総合研究所入社．52年日本ユニシス(株)に移籍．ソフトウェア工学の研究および開発支援ツールの開発に従事．現在，システム技術本部データベースソフトウェア部に所属．ソフトウェア科学会会員．

汎用情報検索システム RDIP

RDIP for General-purpose Information Retrieval Systems

山 崎 裕 明

要 約 情報系システムは多様で変化する情報要求に迅速に対応し，価値ある情報を提供できるものでなければならない．この要件を満たすためには，構造が柔軟であり，拡張性に富み，多様な情報加工機能を持った情報検索システムを構築する必要がある．しかしながら，このような特性を持った汎用的な情報検索システムを独自に開発し，利用を促し，維持管理するのは容易なことではない．

RDIP（Relational Dynamic database Interface Package）は，情報系システムに求められる諸要件に具体的に対応した汎用的な情報検索システムの短期間での構築，および豊富な利用手段の提供によるシステム活用の支援を目的とするミドルソフトウェアとして開発された．

本稿では，情報系データベースの構築から情報の活用に至る過程において，RDIP が支援する機能を順を追って紹介する．

Abstract The information processing system must be so built as to provide valuable information in response to a variety of changing user requirements. The fulfillment of such needs requires the construction of an information retrieval system which is flexible in structure, excellent in expandability and versatile in information processing functionality. However, a general-purpose information retrieval system of this nature is not easy to develop independently, to urge end users to adopt, and to continue maintaining.

The Relational Dynamic database Interface Package (RDIP) has been developed as a middle software product aimed at creating a general-purpose information retrieval system in fewer hours so as to meet the above-stated requirements, and at supporting the most efficient use of it with the help of a wide range of functions available.

This paper describes one RDIP-supported function after another according to the process from the construction of an information database to the utilization of information.

1. はじめに

各企業における情報システムへの課題の一つとして，エンドユーザ・コンピューティング(EUC)の推進と定着化がある．とくに，情報系システムにおける EUC の推進により，情報システム部門が抱えるバックログの削減が可能となり，エンドユーザにとっては，自らの意志で自由に情報の検索と加工を行う道が開かれることになる．EUC を実現するためには，情報系システムの標準化を押し進めると同時に，外部環境の変動に即応可能な柔軟性を持たせ，豊富な情報加工手段を提供できる仕組みを構築しなければならない．情報系システムの再構築による EUC 環境の基盤整備を前提とすると，その着手の第1歩として，現行の情報系システムの見直しを行う必要がある．ここで，情報系システムの見直しで顕在化すると思われる問題認識を整理すると次のようになる．

1)　情報系システムの位置づけが曖昧……情報システムの基幹業務系，管理系，情報系の切り分けが曖昧であり，長年にわたる開発の積み重ねにより，基幹業務システムに管理要素を持ち込んだままとなっている．これにより基幹業務システムの肥大化，複雑化を誘発させ，保守・維持の困難さを招いている．本来，基幹業務システムは業務を遂行する上で必要最小限の関連情報と仕組みから構成されるべきである．一方，情報系システムは異なる業務の事実情報を集約し，多様で変化する情報検索要求に機敏に対応できるものでなければならない．

2)　情報結合が困難……多種多様な情報伝達手段・加工手段が混在することにより，異なる体系から提供された情報の融合が情報の精度を保証する面で困難となっている．

3)　情報の個別要求……利用者にとって，提供される情報の状態を確実に把握することは困難である．たとえば，提供された情報のポジションが確定データか仕掛かりデータか区別が付きにくいといった点である．利用者はこの点が不安であり，個別の要求による一括したデータの提供を求めてしまうことになる．

4)　情報資源管理が困難……格納様式の相違，格納媒体の相違を含んだ情報の物理的な分散化が進み，情報の資源管理が容易でなくなってきている．

これらの問題・課題の解決のためには次に示すような対応が必要となる．

1)　情報系データの伝送・蓄積・格納方式の統一化……情報系データが格納されるデータベースの様式を統一し，十分に正規化されたデータ項目をデータディクショナリ・ディレクトリシステム（DD/DS）により一元管理する．また，業務系システムと情報系システム間のデータ移送手段と移送ルートおよび蓄積手段を統一する．

2)　情報系システム管理基準の設定……情報の管理基準とシステムの運用基準を設定する．情報の管理基準としてデータ構造体系とセキュリティがある．データ構造体系は情報の一貫性と利用者の利便性を考慮した冗長性を合わせ持つ必要がある．このためには，次のような物理的階層を持ったデータ構造とすべきである．

①　基礎データベース：複数の基幹業務システムより収集した基礎データに一定の集約・加工を施したデータ群であり，データ間の一貫性が要求される．

②　目的別データベース：部門管理者の要請に従って，基礎データベースより特定の条件に合致したデータを抽出し，編集・加工を施し，情報に付加価値を持たせたデータ群であり，特定部門に属する利用者を対象とした部門向け情報である．

③　個別データベース：利用者個人の判断で，目的別データベースよりデータを絞り込み，加工・編集を施し，保存した個人向けデータ群である．

3)　情報加工手段の統制……多様で変化する情報要求に対応可能な仕組みを構築し，共通の手続きに従った豊富な情報加工手段を提供する．

4)　システム管理者の負荷軽減……情報系システム管理者の管理範囲を，企業活動で発生した事実情報，または社外から取り寄せた業界動向情報・調査情報等を情報系データとして蓄積・維持することまでとする．利用者にとって，必要な時点で必要な情報を自由に検索でき，自由に加工編集が行える手段が提供されれば十

分であり，特定の情報提供を個別に要求する必然性がなくなる．

ここで述べたような問題認識を共有するユーザが情報系システムを再構築する際に，その対応で列挙した観点をコンセプトとした情報系システムの短期間での構築と，EUC の推進を全面的に支援するミドルソフトウェアとして RDIP が開発された．

2. RDIP の概要

RDIP の目的の一つに短期間での情報系システム構築支援がある．このために，必要となる作業の大部分を自動化しており，会話操作による簡易な作業指示で済むよう設計されている．また，基礎データベースの基盤ソフトウェアとして，情報系システム向きで維持管理が容易なリレーショナル型データベースを採用している．

EUC 支援のために RDIP 自体は 4 GL として使用実績の高い MAPPER で作成されており，目的別データベース，個別データベースは MAPPER レポートとして形式が統一されている．基礎データベースからのデータ抽出と加工編集は，会話操作によ

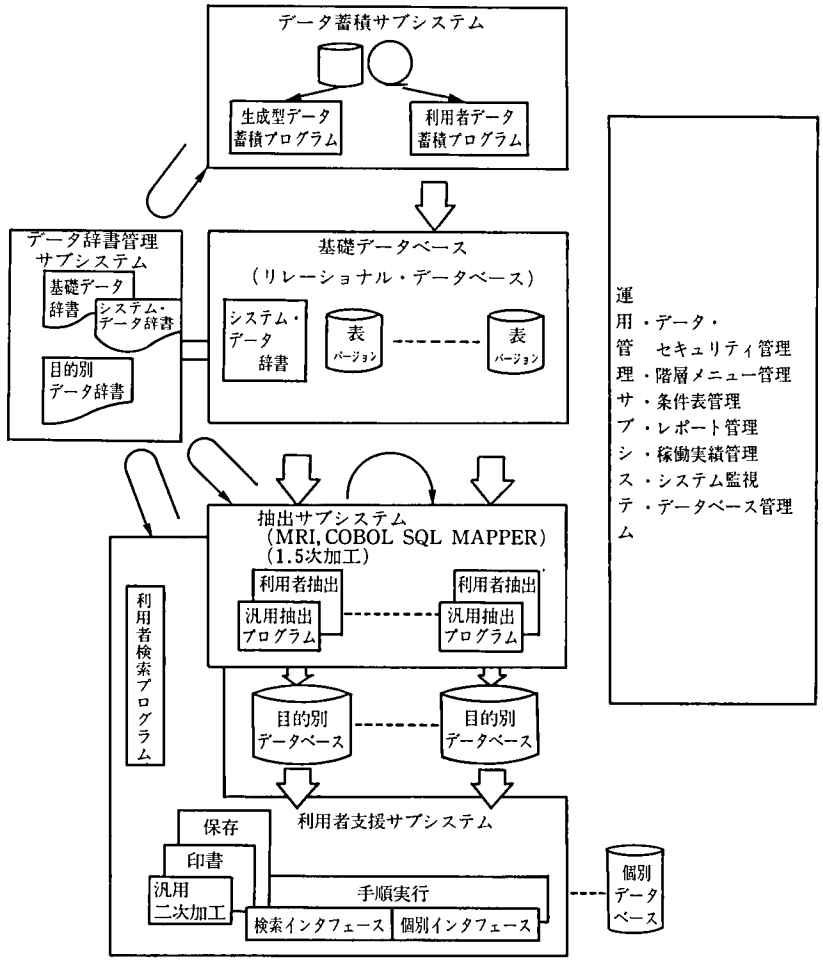


図1　サブシステムの構成

Fig. 1　Structure of subsystems

る手続きの登録だけで済むよう設計されており，試行錯誤的な検索や定型的な検索も行えるよう考慮されている．RDIPを採用することにより，利用者は必要の都度，見たい情報を，望ましい形で入手することが可能となる．

RDIPは，情報系システムの基盤となるデータ項目情報および，基礎データベースの構成情報を一元管理する「データ辞書管理サブシステム」，多様な外部データを統一形式に変換し，データ蓄積までの一連の手続きを極力自動化する「蓄積サブシステム」，目的別データベースの自動作成，および豊富なデータ加工手段による個別データベース作成を支援する「利用者支援サブシステム」，利用者支援サブシステムと連動し，基礎データベースからのデータ抽出・加工・編集をブラックボックス化して行う「抽出サブシステム」，および稼働実績管理，システム監視，流量制御，メニュー管理等の管理支援を行う「運用管理サブシステム」から構成される．

各サブシステムの構成を図1に示す．

3. 情報フレームの構築

情報系システム構築の第1段階は，情報系データベースの設計である．RDIPでは，情報系データベースの中核となる基礎データベースの基盤ソフトウェアとしてリレーショナルデータベース・システム（RDMS）を前提としている．リレーショナル型データベースは，データ構造が単純であり，構築と維持管理が容易であること，列の追加，エリア拡張等の物理的変動要因に柔軟に対応が可能であること，大量のデータを格納でき，多様な検索要求に対して一定の検索効率が保証されること，および情報間の結合・射影・制限操作が可能であることから，情報系システムに最適である．

RDMSを使用した基礎データベースのフレーム構築では，まず基礎データベースの基本単位であるデータ項目についてデータ辞書により一元管理を行い，次にデータベース構造を設計し，RDMSシステム辞書に構成要素（スキーマ，記憶域，表，列等）を登録する手続きが必要となる．『データ辞書管理サブシステム』はこの手続きに添って，必要となる管理情報を一元管理するとともに，フレーム構築のガイドラインを提供する．

3.1 データ項目の統制

基礎データベースの基本要素であるデータ項目については，情報システム全体で扱われるデータ項目と整合性をとる必要がある．このためには情報システム全体にわたってデータ項目がDA（情報資源管理者）によって統制され，データ辞書で一括管理されていることが前提となる．しかしながら辞書システムが実在しないことも想定し，RDIPでは次の2ケースの運用体系を設けている．

1) 統合化辞書システムからの情報入手……情報資源管理の達成レベルが高い企業では，辞書システムが実在し，一元的なデータ項目管理がなされている．この場合には，既存の辞書システムからデータ項目管理情報を入手し，RDIP特有の管理属性を付加し，RDIP辞書システムに一括登録する．

2) RDIP辞書システムでの一元管理……既存の辞書システムが存在しない場合は，RDIP辞書システムに管理情報を登録する．

データ辞書管理サブシステムのデータ項目に関する主な管理機能を次に列挙する．

1) データ項目識別管理……RDIPでは，登録されたデータ項目に対しデータ項目識別子を自動採番し，これをもって表の列名とする．データ項目の統制を図る際の標準化作業の一環として列名の命名を行うが，大量のデータ項目について有意なネーミングを施すには限界がある．RDIPでは，列名はあくまでデータ項目の内部コードとして扱い，利用者の識別手段は日本語名であるとしている．したがって，日本語名もRDIP辞書内ではユニーク性が求められる．

2) 桁数管理……データ項目単位に，データベース格納時の桁数と表示桁数のデフォールト管理を行う．

3) セキュリティ管理……データ項目単位に機密保護レベルの管理を行い，権利レベルとの対比により，データ開示か否かの判定を行う．

4) 数値の単位管理……数値の単位に関し，データベース格納時の数値の単位とデータ検索時に表示単位の指定を可能とする．

データ辞書管理サブシステムを利用したデータ項目定義情報の登録例を図2に示す．

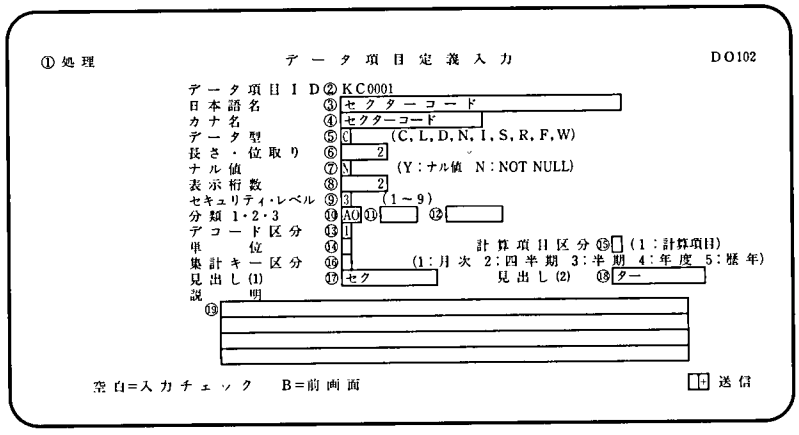


図2 データ項目定義入力画面

Fig. 2 Input screen of data item definition

3.2 データベース構築のガイドライン

RDIPではデータベース構造の設定規準を設けており，これをもってデータベース構築時のガイドラインとしている．RDMSが提供するリレーショナル操作を使用するには集合演算の考え方とデータ操作のルールを熟知しておく必要がある．しかしながら，このことは一般の利用者にとって大変困難な課題となる．RDIPは表間の結合方式を規定するとともに，RDIP独自の表の形態も規定することにより，データベース構造をパターン化し，結合操作によるデータ抽出手続きを統制する．

表の形態には，表に格納されるデータの性格により決定される主表と従表，同一形式データを物理的分散するか否かで決定される同配列表と個別表，時系列集計操作を適用する表か否かで決定される時系列表と非時系列表がある．

3.2.1 主表と従表

主表は情報系データベースの中核として位置づけ，主に企業活動の事実情報をトラ

ンザクションデータとして格納する表である．従表は主表の情報を補足する表であり，マスタデータとして実体情報を格納する表である．

1) 主表と従表間の情報結合……主表と従表を結合方式で結び付けることにより，主表に存在する実体識別コード（例：部署コード）で従表の実体属性項目（例：部署名称）を取り付ける情報操作が可能となる．結合方式には PARTIAL-JOIN と INNER-JOIN がある．

① partial-join：主表に存在するコードで従表に対応するコードが存在しない場合，検索結果としてスペースまたはゼロを代入する（図3の **a**～**j**）．主表に存在するコードで従表に対応するコードが存在すれば，指定された従表の項目を取り付ける（図3の *kl*）．

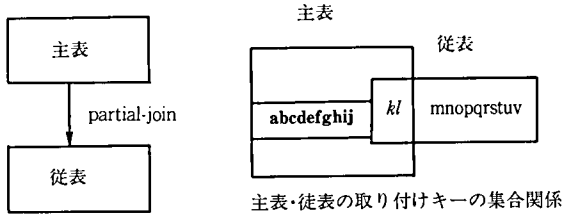


図3 部分結合

Fig. 3 Partial-join

② inner-join：主表・従表間で両者に同一のコードを持つ行が検索対象となる（図4の *kl*）．

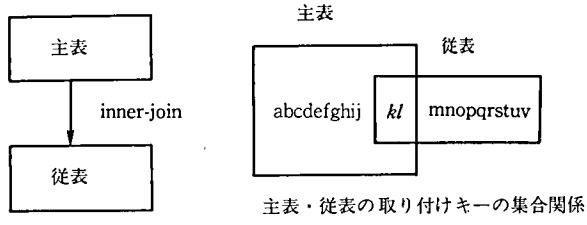
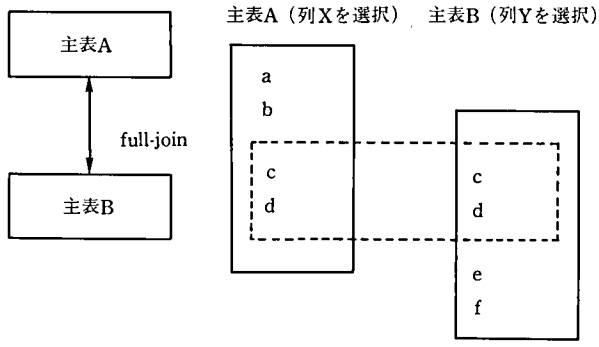


図4 内部結合

Fig. 4 Inner-join

2) 主表と主表間の情報結合……情報系システムを構築する際，分散する複数の基幹業務システムよりデータを収集し，これらを有機的に結合する試みが行われる．このようなケースで問題となるのは情報間の結合度合いである．企業活動で発生するトランザクションは非同期に発生し，各システムで保存するトランザクションデータ間で有機的な結合を保証することは困難である．このような状況下で情報系システムを構築し，複数の表より列を選択する情報結合を図ると，複数の表に共通に存在するデータのみが検索対象となってしまい，検索データが極端に絞り込まれてしまうことになる．RDIP はこのような状況を想定し，主表・主表間の結合方式を次のように定めている．

① full join：表内のインスタンス（列に対する条件設定に該当するもの）はすべ



検索結果	突き合わせキー	列xの内容	列yの内容
	a	100	スペース
	b	2000	スペース
	c	150	250
	d	230	14000
	e	スペース	500
	f	スペース	750

図5　完全結合

Fig. 5　Full-join

て検索対象データの取扱いとする．ただし，相手側に結合可能なデータが存在しない場合はスペース表示とする（図5）．

3.2.2　同配列表と個別表

リレーショナル・データベースの表には大量のデータを格納することが可能であり，一定の検索効率の保証が得られるが，月次データのように一定の間隔で発生するデータを逐次格納する場合は，そのつど記憶域のガベージを実施しないと検索効率の低下を招く恐れがある．RDIP は大量データを同一形式の複数表に物理的に分散化させながら，論理的には同一の表として扱える機能を提供する．

1)　同配列表……大量のデータを格納する主表について，論理的には一つの主表とみなし，物理的に次の何れかのパターンで分割・配置することを可能とする．これにより，データの蓄積は物理的分割の単位で可能となる．

・日別　・月別　・四半期別　・半期別　・暦年別　・年度別　・部門別

2)　個別表……物理的・論理的にも単一の扱いを受ける主表である．

同配列（月別）の抽出例を図6に示す．

3.2.3　時　系　列　表

情報に価値を持たせる加工手段で多用されるものに，情報の時系列推移展開がある．RDIP ではこれを基本機能の一つに位置づけし，時系列横展開機能として提供する．このために，時系列データを保有する表で，検索データに対し時系列横展開操作または時系列集計操作（月次データを半期単位，年次単位で集計する等）を行う場合に，主表に与える属性として次に列挙するパターンの何れかで時系列表の指定を行う．

・日別　・月別　・四半期別　・半期別　・暦年別　・年度別

時系列表に対しては，次の例1，例2に示すような抽出・編集が可能となる．

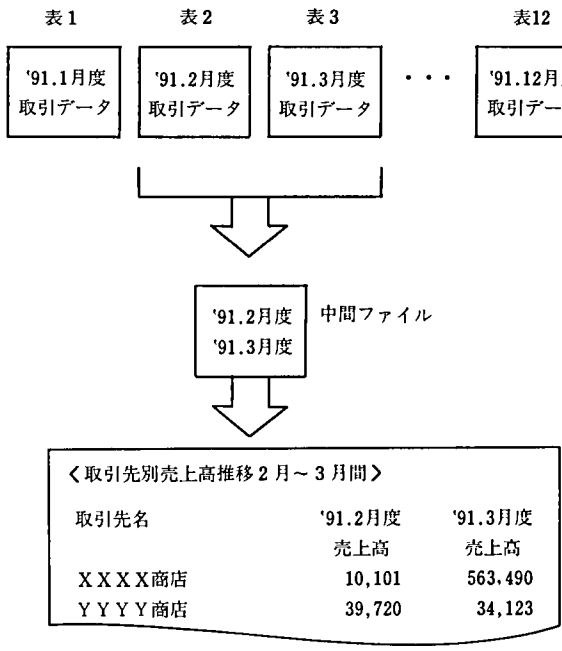


図6 同配列（月別）の例

Fig. 6 Example of DOUHAIRETU(monthly)

例1 時系列横展開（時系列横型）

目的 部店別に直近3か月の売上推移を見たい．('91.3時点)

指示 ① 抽出項目の特定 売上金額（主表より），部店名称（従表より）

② 集計指示 集計キー項目の指定 部店コード，取扱い月

集計項目の指定 売上金額

③ 横展開指示

相対月	表示欄
0	A
1	B
2	C

④ 範囲指定 from ('91.1) ～to ('91.3)

検索結果

	(表示欄A)	(表示欄B)	(表示欄C)
部店名称	前々月売上金額	前月売上金額	当月売上金額
本店	90,234,789	101,561,900	100,230,590
支店A	367,190	570,810	671,490
支店B	1,901,260	2,000,780	2,268,300

例2 時系列集計（時系列縦型）

目的 部店売上表（時系列…月次型）より，過去4期に渡る半期別の売上実績を本店，支店1，支店2について見たい．('91.3時点)

指示 ① 抽出項目の特定 時系列集計キー（半期），売上金額

② 集計指示 集計キー項目の指定 時系列集計キー（半期）

部店コード

集計項目の指定　　売上金額

③ 横展開指示　部店コード　　表示欄

H 01　　A

S 01　　B

S 02　　C

④ 範囲指定　　from ('89.4) ～to ('91.3)

検索結果

	(表示欄 A)	(表示欄 B)	(表示欄 C)
売上期	本店売上金額	支店1売上金額	支店2売上金額
8904	190,530,100	56,105,900	30,230,590
8910	201,367,190	55,570,810	60,671,490
9004	240,901,260	60,000,780	55,268,300
9010	235,892,000	101,900,810	40,390,500

3.3 基礎データ辞書による一元管理

スキーマ，記憶域，表，列，ビュー等，RDMS システム辞書に登録すべきデータベース定義情報，および主表，従表，同配列表，時系列表，結合方式等，RDIP 独自の管理情報はすべて RDIP 基礎データ辞書で一元管理される．RDIP 利用者にとって RDMS システム辞書は完全にブラックボックスであり，RDMS システム辞書の維持管理はデータ辞書管理サブシステムが内部で代行する．したがって，RDIP 利用者はデータベース定義情報に関する基礎知識を習得するだけで，データベース定義情報を RDMS に登録することができる．

基礎データ辞書の構成を以下に示す．

・RDIP 独自の管理情報

情報グループ定義（主表/従表，結合方式）

同配列表管理（表またはビュー表との関連，時系列単位）

データベース管理（データポジション管理，蓄積状況管理）

・データベース定義情報

スキーマ定義，記憶域定義，バージョン定義，表定義，列定義，主キー定義，外部キー定義，2次インデックス定義，ビュー定義

・データ項目管理情報

日本語名称，カナ名称，データ型，長さと位取り，セキュリティ・レベル，見出し，単位，ディスクリプション

4. データのロード

基礎データベースの設計情報を基礎データ辞書と RDMS システム辞書に登録した後に，情報系データの初期ロードを行うことになる．RDMS を直接使用したロード作業では，データ編集作業，データロード作業およびデータベース管理が必須となる．

1) データ編集作業……RDMS が提供するロード・ユーティリティ（HRLOAD または RDMSLOAD）の入力フォーマットに合わせたデータ編集を行う．

2) データロード作業……ロード・ユーティリティの入力パラメータを表単位に作成し，これを実行する．

3)　データベース管理……ロード処理に伴って情報系データベース全体にわたり，データ格納状態を管理する．

基礎データベースが大規模であると，これらの作業負荷は確実に増大する．RDIPの「蓄積サブシステム」は，これら諸作業を極力自動化することによるロード作業の軽減を目的として，次の機能を提供する．

1)　プログラム作成……ロードデータの加工・編集プログラムをパラメータ入力により自動生成する．

2)　データロード支援……ロード・ユーティリティ用のパラメータを自動生成し，ロード処理を自動起動する．HRLOAD, RDMSLOADの使用選択が可能である．

3)　データベース管理支援……ロードの結果はRDIPシステム内の表管理テーブルに反映され，データベース全体の蓄積状況を一元的に把握することができる．

4)　データポジション管理……ロードデータの確定日付をパラメータ指定することにより，データ検索時に『データ日付一覧表示』として，利用者に確定日付を伝達することが可能である．

5. データの抽出

基礎データベースから目的とするデータを抽出する手続きは，制限・射影・結合等のリレーショナル操作で行うことになるが，RDIPでは利用者の会話指示（条件指示，列の選択等）情報を内部でSQL文に変換し，これを自動発行する．データ抽出の経路は次の2パターンがあり，検索要求の程度，抽出対象の表構成等の諸条件をRDIPが識別・判断し，経路を自動選択する．

- MRI経由での抽出：単一主表とこれに従属する複数の従表が検索対象となる比較的単純な抽出条件の場合に適用される．
- バッチ経由での抽出：複数の主表や同配列表等，検索対象が多岐にわたる場合や，RDIP独自の編集処理を加味させる場合に適用される．

① 制限操作：データの絞り込みは，条件入力画面からデータ項目単位の条件やデータ項目間の複合条件を指定することにより可能となる（図7）．

② 射影操作：列の選択は表単位に行うが，選択された列の並び順を自由に指定することができる（図8）．

③ 演算操作：計算指示画面から，項目識別番号（#nn），四則演算子および正の定数を用いた計算指示を行うことにより，演算式を含んだSELECTリストをSQL文の一部として自動生成する．計算結果は予約語CALmn$として自動割り当てした項目に代入される（図9）．

④ 集計操作：集計キー項目，集計項目の指示により，SUM関数とGROUP BY句をSQL文の一部として自動生成する．集計結果は予約語SYUmn$として自動割り当てした項目に代入される（図10）．

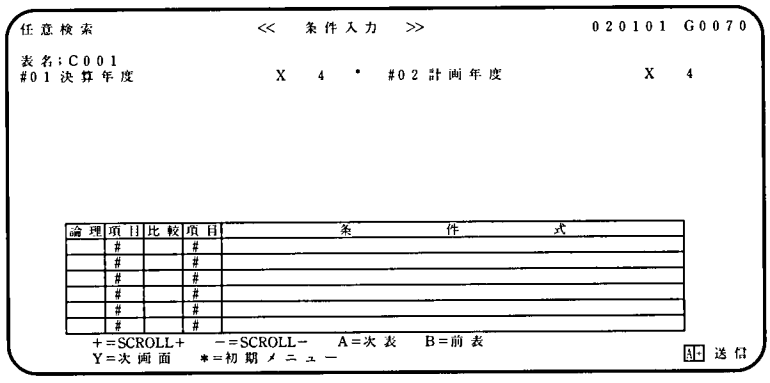


図7 条件入力画面

Fig. 7 Input screen of a condition

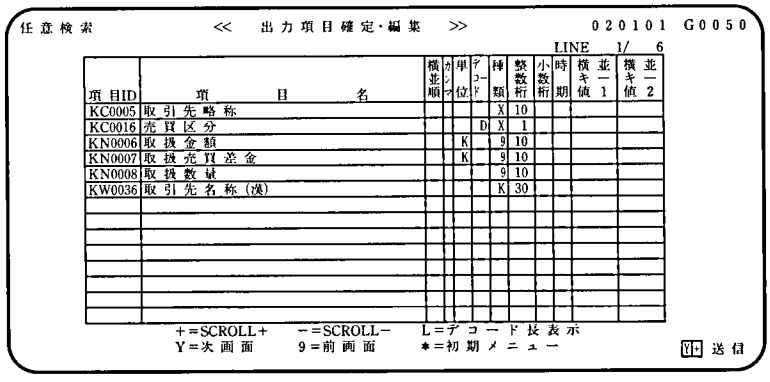


図8 出力定義画面

Fig. 8 Screen of output definition

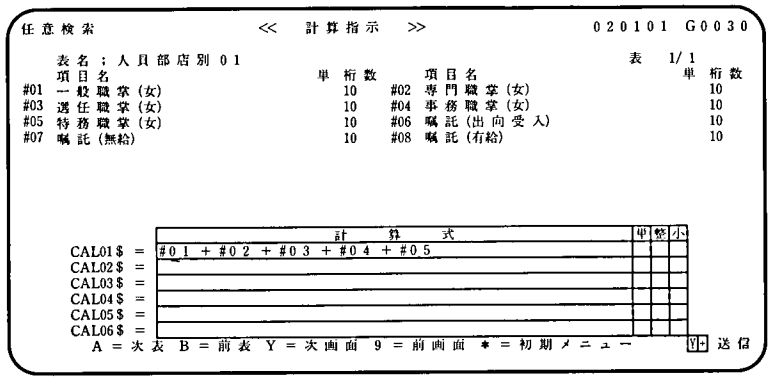


図9 計算指示画面

Fig. 9 Screen of indication for calculation

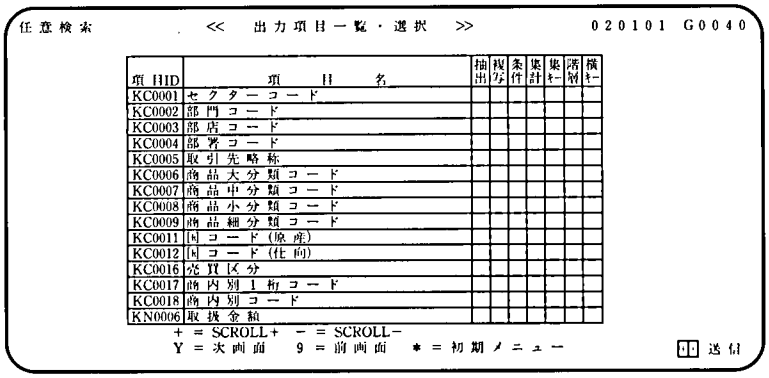


図10　出力項目選択画面

Fig. 10　Screen of output-item selection

6. 情報の編集・加工

情報の編集および加工処理は手順化され，自動処理されるのが望ましい．RDIP では，あらかじめ編集指示を行うことにより，次に列挙するデータ操作を自動的に行う．

1) 単位変換……数値項目について，データベース格納時の単位より，編集指示で指定された表示単位に変換を行う．

2) カンマ挿入

3) 表示桁編集……データベース内の項目サイズに依存せず，指定された表示サイズに自動編集する．

4) デコード……抽出したキー項目より名称を取り付け，同一フィールドに上書きする．

5) 縦横変換……集計操作と連動し，集計キーの取り得る値により集計データを特定の表示欄に出力する．

6) 特殊帳票の作成

① 階層集計表：特定項目の値域により，指定されたレンジ単位に，出現度数とレンジに含まれる数値項目の集計結果を一覧表として作成する．

② 順位表　：特定項目の集計値の順位に従って，下位の集計項目の明細を含んだ順位づけの一覧表を作成する．

7) 後処理計算……抽出された数値項目または数値項目間での自動演算処理

①縦計算，②横計算，③百分率計算，④項目間の四則演算処理

RDIP は会話操作による二次的なデータ加工・編集機能を以下に示すような「汎用二次加工」として提供する．

- 検索（SEARCH）　：データの絞り込み
- 整列（SORT）　：データの並び替え
- フォーマット変換　：項目の配置替え
- 縦横変換　：行，列の置換
- 計算　：項目間の四則演算，小計，中計，大計処理(図 11)

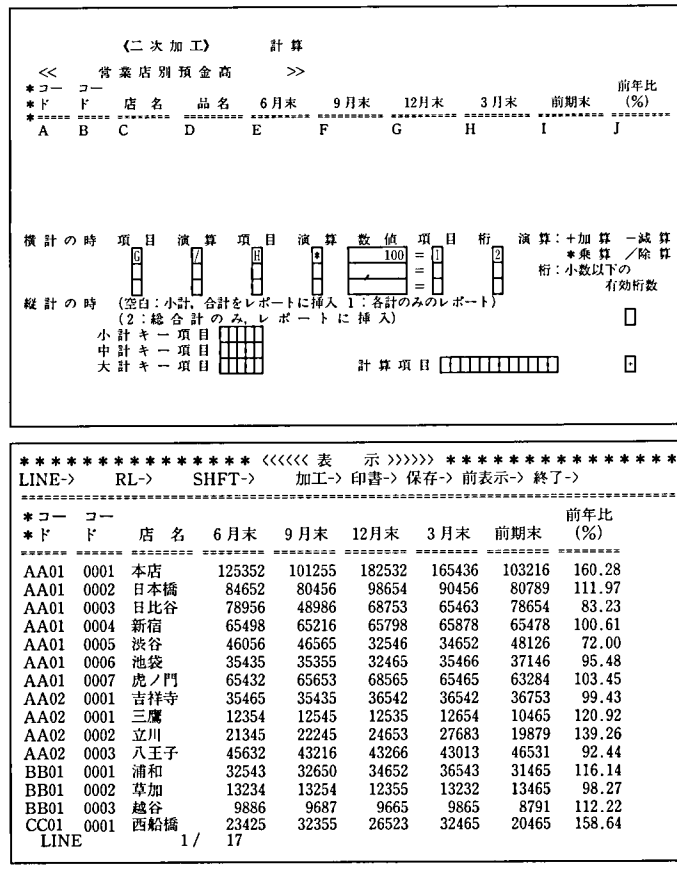


図11 計算指示と計算結果の画面

Fig. 11 Indication and display screen of calculation

- 併合 ：複数レポートの併合
- 見出し確定 ：帳票名，見出しの再編集
- 項目再編集 ：項目の順序付け，カンマ挿入，カンマ削除
- 見出し項目追加 ：特定位置に項目を生成
- 桁調整・列複写 ：項目の桁数変更，項目の内容を複写
- 会話型処理 ：MAPPER の会話操作による編集・加工

同じく，「汎用二次加工」のグラフ表示および SUFICS (SUper Financial Integrated Control System，意思決定支援システム) 簡易分析機能の提供により，情報分析支援を行う．

- グラフ表示：円グラフ，棒グラフ(横，縦，積み上げ，3次元)，折れ線グラフ，混合グラフ(図12)，レーダーチャート，ターゲットチャート，散布図
- SUFICS 簡易分析：簡易分析の種類と出力項目の関連を表1に示す．

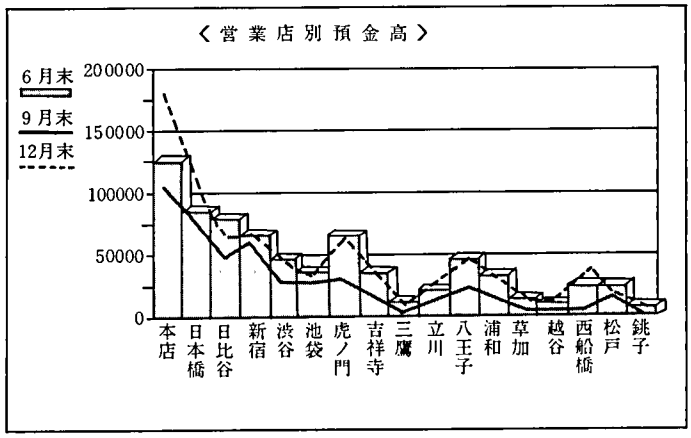


図12　混合グラフ表示画面

Fig. 12　Display screen of mixing graph

表1　分析の種類

Table 1 Type of an analysis

処理名	出力項目
基本統計量の算出	合計，最大数，最小数，平均，標準偏差，分散，歪度，尖度
相関係数の算出	相関係数
曲線あてはめによる算出	8種類の曲線あてはめによる統計量(自由度調整済み寄与率，F値，ダービン・ワトソン比，F分布(99%)，F分布(95%))およびパラメタ(a，b，c)
曲線あてはめによる予測(縦型)	実測値，推定値，予測値，残差
曲線あてはめによる予測(横型)	実測値，予測値
回帰モデルによる統計量の算出	統計量(自由度調整済み寄与率，F値，ダービン・ワトソン比，F分布(99%)，F分布(95%))，T分布，回帰による(自由度，平方和，不偏分散)，回帰からの(自由度，平和方，不偏分散)，全体の(自由度，平方和，不偏分散)，説明変数のT値，標準偏差
指数平滑法による予測値の算出	平滑化定数，残差分散，予測値，実測値
キャッシュ・フローの現在価値の算出	キャッシュ・フロー，割引率，現在価値
キャッシュ・フローの内部収益率の算出	キャッシュ・フロー，減債基金率，内部収益率

7. 情報の保存と再利用

情報系の検索パターンは，使用頻度の高い定型的な検索と，試行錯誤による非定型検索に大別される．これらの検索手段を実現するには検索手続きのパラメータ化と保存および再利用の機能が必要となる．また，加工済みデータの一時保存とセキュリティを十分考慮した第三者による再利用機能が情報系システムに求められる要件でもある．RDIPはこの認識のもとに次に列挙する機能を提供する．

1)　検索の手順化……情報の抽出と編集手続きは会話操作による指示で順次行うが，指示内容はパラメータ化され，手順レポートとして保存対象となる．

2)　手順再実行（試行錯誤的な検索）……手順レポートは再利用可能であり，会話操作による手続きの部分変更（抽出項目の変更，条件の変更等）で再検索が可能

である．

3) 手順の定型化……特定の手順レポートを定型手順として登録することができる．定型登録されると，メニュー画面からの選択で，抽出・編集処理が自動実行されることになる．定型検索時に特定の画面を表示させることも可能である(例：条件入力画面の表示)．

4) 目的レポートの保存……検索結果レポートの期限付きの保存と再利用が可能である．保存領域は共通領域と部門領域に大別される．

共通領域：利用者を限定しない全社共通の情報を格納する．

部門領域：特定部門内の利用者に提供すべき情報を格納する．

5) 検索の自動スケジュール……定型化された手順について，起動日時を指定することにより，月次，週次，日次単位の定型レポートを夜間に無人で作成することが可能である．

6) 目的レポートの閲覧……レポート選択機能により保存レポートを再表示することが可能である．さらに，表示データを「汎用二次加工」機能を用いてさらに加工を加え，自己のレポートとして再保存が可能である．

7) セキュリティ機能……手順レポート，保存レポートとも第3者による再利用が可能であるが，種々のセキュリティ検査が実施される．

8. 情報の機密保護

RDIPは表の論理的な集合単位である情報グループ，および機密保護レベルの概念によりきめ細かい機密保護管理を行う．

8.1 情報グループ

情報グループとは，利用者が検索可能な表を限定する目的で管理者が設定する論理的な表の集合体である．情報グループの設定例を図13に示す．

8.2 機密保護レベルと権利レベル

機密保護レベルは基礎データ項目，表，情報グループ，手順レポートおよび目的レポートに適用される情報の保護レベルである．

- 基礎データ項目　：機密保護レベル設定の基礎単位である．
- 表　：表を構成するデータ項目(列)の機密保護レベルの最大値が自動的に適用される．
- 情報グループ　：情報グループを構成する表の機密保護レベルの最大値を設定する．
- 手順レポート　：選択データ項目の機密保護レベルの最大値が自動設定される．
- 目的レポート　：選択データ項目の機密保護レベルの最大値が自動設定される．

権利レベルは利用者単位に設定されるデータアクセスの権限レベルである．同一利用者でもアプリケーションNo単位に設定が可能となっている．アプリケーションNoとはRDIP独自の概念であり，基礎データベースの管理単位に設定する識別番号である．通常は情報発生元の業務別に設定するが，セキュリティを考慮して細分化す

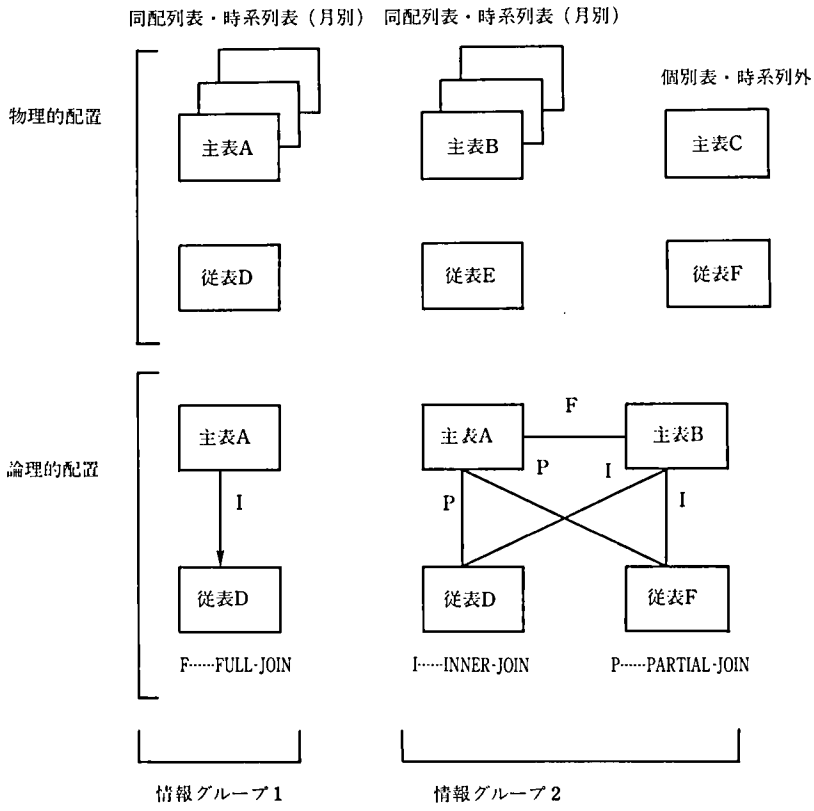


図13 情報グループの設定

Fig. 13 Settlement of information group

ることも可能である．

[アプリケーションの設定例]

データベース	アプリケーション No
人事情報［一般向け］	1
人事情報［人事部専用］	2
経理情報［一般向け］	3
経理情報［経理部専用］	4

[権利レベルの設定例]

検索者	アプリケーション No の別の権利レベル			
	ap 1	ap 2	ap 3	ap 4
人事担当者	9	9	9	5
経理担当者	9	5	9	9
人事・経理以外	9	5	9	5

人事情報・経理情報の機密保護レベルを一律6と設定すると，検索者と検索可能な情報の関係は次のようになる（権利レベルが機密保護レベル以上であれば検索可能）．

検索者	検索可能なデータベース
人事担当者	人事情報［一般］，人事情報［人事部専用］，経理情報［一般］
経理担当者	人事情報［一般］，経理情報［一般］，経理情報［経理部専用］
人事・経理以外	人事情報［一般］，経理情報［一般］

セキュリティの機能概要を図14に示す．

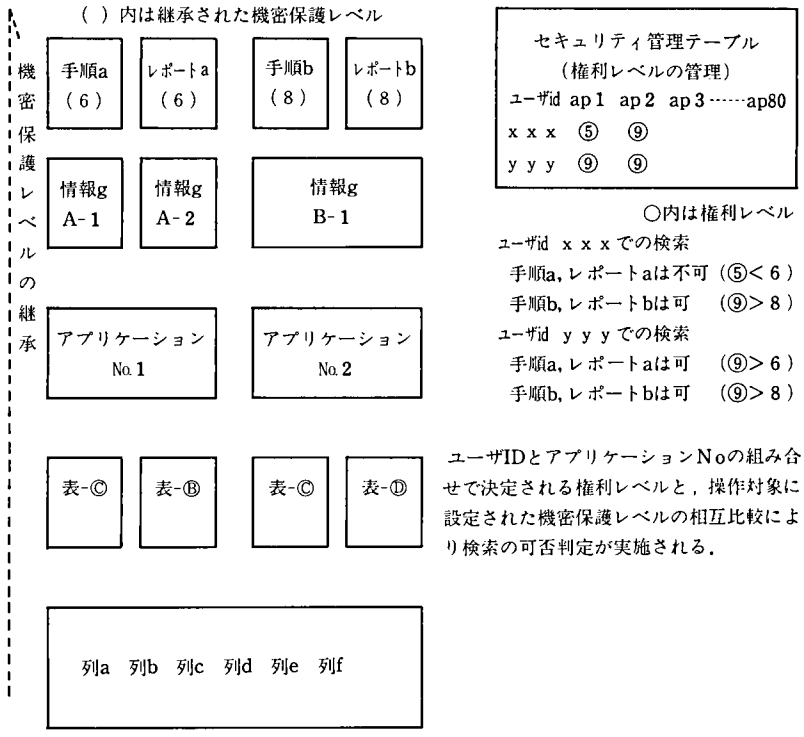


図14 セキュリティ機能の概念

Fig. 14 Concept of security

9. 情報の伝達

情報として付加価値を持った検索加工データが, 一定のセキュリティのもとに外部に伝達されることも情報系システムに求められる要件である. RDIP が提供する情報伝達手段を次に列挙する.

1) テキスト形式変換……PC 上のディスケットにダウンライン・ロードし, サード・パーティ・ソフトウェア独自の特殊編集を可能とする.
2) 他システムとの連動……ディスケットを介した EXOS システム(EXcellent Office System, ユニシス統合オフィス・システム) との連動や他の MAPPER システムへのレポート転送が可能である.
3) 帳票出力……特定端末への罫線付き出力, センタ・プリンタへのオーバレイ情報を付加した大量出力が可能である.
4) 外部ファイル出力……抽出・編集データを SDF (System Data File) 形式ファイルとして作成し, 分散環境下のサーバにファイル転送することが可能である.

10. おわりに

RDIP は, 本稿で述べてきたように情報系システムの構築からその活用までを支援するソフトウェアであり, RDIP の適用により, エンドユーザ・コンピューティングの推進を円滑に行うことができる. しかしながら, RDIP は商品化されて間もないソフト

ウェアでもあり，今後適用されるユーザの環境・背景を考慮しつつ，ユーザの要望を積極的に取り込み，進化発展させるものと考えている．また，近年顕著となったダウンサイジング化の方向に対応すべく，RDIP の思想を踏襲した分散マシン環境下での情報系ミドルソフトウェアの提供も計画中である．

参考文献
［1］「シリーズ 2200・1100　総合情報システム構築支援プログラム　RDIP 概説書」，ユニシス・マニュアル　481263603.
［2］「シリーズ 2200・1100　総合情報システム構築支援プログラム　RDIP 使用ガイド　利用者支援サブシステム編」，ユニシス・マニュアル　481265626.
［3］「シリーズ 2200・1100　総合情報システム構築支援プログラム　RDIP 使用ガイド　汎用二次加工機能編」，ユニシス・マニュアル　481265631.
［4］「シリーズ 2200・1100　総合情報システム構築支援プログラム　RDIP 解説書　データ辞書管理サブシステム編」，ユニシス・マニュアル　481265627.

執筆者紹介　山　崎　裕　明（Hiroaki Yamazaki）
昭和 24 年生．47 年横浜国立大学教育学部数学科卒業．同年日本ユニシス(株)入社．OS 1100 の保守サポートに従事した後，製造系中心のシステム開発に携わる．現在 I&C システム本部 MISA プロジェクトに所属．

オブジェクト指向データベースにおける問い合わせ式構造的最適化技術

A Query Structural Optimization Technique for the Object-oriented Database

山口裕久

要約 本稿では，オブジェクト指向データベース管理システムにおいて必須となるであろう問い合わせ式の意味的経路最適化について述べる．多くの問い合わせ式最適化はリレーショナルデータベースの研究と共に進んできた．しかし，オブジェクト指向データベースではクラス，クラス階層，関連表現といったセマンティックの表現能力の拡張がなされている．さらに，オブジェクト指向データベースでは，オブジェクトの振る舞いがオブジェクトの構造と一緒に定義できるカプセル化の機能を持っている．オブジェクト指向データベースの問い合わせ式最適化は，これらの構造的意味や振る舞い的意味を失うことなく最適なアクセス経路を見つけなくてはならない．

ここでは，SIM (Semantic Information Manager) 上で実現されている問い合わせ式最適化を例に意味的経路最適化を説明する．またSIMで実現されている最適化を例として，オブジェクト指向データベースで必要であるオブジェクトの等価性を問い合わせ式で解決するための一方法を示す．

最後に，オブジェクト指向データベースにおける問い合わせの最適化の今後の課題を述べる．

Abstract This paper describes the semantic path optimization in query language, which is regrarded as becoming essential for an object-oriented database management system (ODBMS). Studies of query optimization technique have been done, for the most part, in parallel with those of the relational database. However, extended semantic representations involving classes, class hierarchies and relationships are supported in the object-oriented database, and additionally available is the encapsulating capability which allows object behavior to be defined together with object structure. What is required for ODBMS query optimization is to generate an optimal access path without losing structural and behavioral semantics.

Taking, as an instance, the query optimization implemented in the semantic information manager (SIM), the author intends to discuss one of the most sophisticated semantic path optimization techniques and also one of the ways to solve object equability (needed by the object-oriented database) with the help of query structural optimization. In the end, some mention is made of problems left unsolved yet regarding ODBMS query optimization.

1. はじめに

1980年代後半からコンピュータが単なる事務処理の道具から，戦略的情報処理システム等に代表される企業の利益を生むための道具という見方へと変化してきた．コンピュータ化の対象範囲も，事務処理分野からCADやCASE等のように複雑な情報構造を必要とする業務へと拡大してきた．これらの変化に伴い，データベース管理シス

テムへの要求も変化し，ただ単にデータを格納するだけではなく，複雑な構造を持ったデータを表現する能力に対する要求が急速に高まってきた．

データ構造が複雑になるにしたがって問い合わせ式で要求される検索も複雑になり，一つの問い合わせ式でアクセスされる情報量も増加してきた．情報に対するアクセスも定型的なバッチ式アクセスから，情報が必要とされる場所で必要な形でアクセスするというように変化してきた．そのため，複雑な構造を持った情報を簡単な問い合わせ式で効率よくアクセスすることがデータベース管理システムに対して要求されるようになった．

本稿では弊社のデータベース管理システムの一つである SIM (Semantic Information Manager) を例に，複雑なデータ構造に対して効率よくアクセスし結果を得るための問い合わせ式最適化技術 (query optimization) を紹介し，オブジェクト指向データベースで要求されるであろう最適化技術を考察する．

2. 最適化の必要性

問い合わせ式最適化は，リレーショナルデータベースによって最初に提供された．リレーショナルデータベース以前のネットワーク型データベースは論理構造が物理構造に大きく依存していたが，リレーショナルデータベースによって論理的データ独立と物理的データ独立が提供された．論理的データ独立によって，データベース利用者は物理構造への波及を最低限におさえて論理構造を変更できるようになった上に，物理構造が論理構造設計に影響を与えなくなった．物理的データ独立によって利用者プログラムから物理構造の記述が排除できるようになり，物理構造の変更が利用者プログラムに与える影響を最小限にできるようになった．

ネットワーク型のデータベースでは利用者が必要な情報を得るためにデータ構造間を航海 (Navigation) しなければならなかったが，利用者はアクセス効率を考えたアクセス経路指定ができた．しかし，SQL では結合演算により必要な情報を得るため，利用者はデータ構造間を航海する必要がなくなり，データベースの操作性は格段に向上した半面，利用者によるアクセス効率を考慮したアクセス経路の指定は行えなくなった．

データ独立とデータ操作言語によって，利用者は物理的なアクセス経路を意識することなく容易にデータをアクセスすることが可能になった．しかし，アクセス経路の指定ができないことによってアクセス効率が低下した．このアクセス効率の低下を補うために，問い合わせ式ごとに最適なアクセス経路を選択する問い合わせ式最適化機能がデータベース管理システムに必要な機能の一つとされるようになった．

3. 問い合わせ式最適化の一般的な意味

問い合わせ式最適化とは，論理的なスキーマに対する利用者の問い合わせ式を，物理構造に対するアクセスに置き換えると同時に，最も効率の良いアクセス方法やアクセス経路を選択することである．

効率の良いアクセス方法の選択とは，特定の物理構造を最も早くアクセスする方法を選択することである．主キーを用いてある特定のレコードをアクセスするには，ハ

ッシュ機能を持ったインデックスを用いることが，最小の物理ディスク入力で物理構造からレコードを得る方法である．レコードをある順番で並べたり，ある順番で物理構造をアクセスするには，B-Tree インデックスが効率よくアクセスする方法である．

効率の良いアクセス経路の選択とは，複数の物理構造をアクセスするために最も効率の良い経路を選択することである．n 個の物理構造を効率的にアクセスするためには，$n!$ 個の経路の中から最も効率の良い経路を選択する必要がある．

最適化には，問い合わせ式の選択条件や属性を基本とした規則による最適化方法と，統計情報や物理的要素である構造体のレコード数やインデックスのレベルの深さ等を考慮したコストによる最適化方法がある．

4. SIM の問い合わせ式

SIM は，セマンティックデータモデルの代表的なモデルである SDM を基礎に開発されたセマンティックデータベースである．

SIM は，同じ特徴を持つエンティティの集合をクラスとして表現するクラス化の機能を持つ．さらに，クラス内のエンティティ集合の部分集合をサブクラスとして表現する汎化階層構造の機能も持っている．サブクラスは上位構造であるスーパクラスの属性を継承することができる．クラス間の関連は構造の一部として EVA(エンティティ値属性) で表現でき，EVA で関連づけられているクラスの属性は，EVA を通して EVA が定義されているクラスの属性の拡張として参照できる．したがって SQL のような結合演算の必要がない．SIM の問い合わせ式は視点クラスという考えを持っており，あるクラスを利用者の視点として論理スキーマを検索することができる．視点クラスから EVA を通しての拡張属性や汎化階層における継承属性によって利用者の論理スキーマに対する見方を指定できる．

拡張属性や継承属性で関連をたどり，複数のクラスにアクセスする複雑な検索が簡単な問い合わせ式で表現できる．アクセスするクラスの数が増加すれば，アクセスに時間がかかることは当然である．しかし，問い合わせ式が簡単に記述できることの代償として検索に時間がかかるのであっては，実用に耐えるデータベースということはできない．

SIM は，リレーショナルデータベースと同じ手法を用いて最適なアクセス方法の選択を行う．物理的な検索を開始するクラスの選択は内部的にはリレーショナルデータベースの手法を継承している部分もあるが，リレーショナルデータベースと SIM の意味表現能力の相違や視点クラス，拡張属性，継承属性等によって以下に述べるようなアクセス経路選択方法を採用している．

5. スキーマ図

問い合わせ式は，構文検査の段階で論理的なアクセス経路として木構造のスキーマ図 (Schema Graph) に展開される．スキーマ図は問い合わせ式に含まれるクラスとクラスの関連を論理スキーマ上でたどったものであり，問い合わせ式に必要な論理スキーマの部分を抽出したものである．

スキーマ図の節となるクラスは，まず第一に視点クラス，第二に問い合わせ式にお

ける属性の並びに現われる属性や選択条件内に現われる属性が直接属性となるクラス，第三に問い合わせ式には直接現われないがEVAをたどることによって生じるクラスからクラスへの経路の途中にあるクラスである．EVAや汎化階層はクラス間の関連を表現するものであるので節と節を結ぶ経路となる．スキーマ図は問い合わせ式の視点クラスを始点とし，関連や汎化階層等の意味を経路として表しているため，意味経路木構造（semantic path tree）とも呼ばれる．

それぞれの節は他の節へFather, Child, Siblingという三種類の経路と1：1や1：Mといった関連等級（degree）を保持している．節から節へのEVAがSV（単一値）で，その逆関連のEVAもSVの場合に1：1となる．両方のEVAがMV（複数値）の場合にはM：Nとなる．節AからChildで結ばれている節BでのFatherは節Aを示している．したがって，FatherとChildは逆関連を示しており，どちらの方向でも節をたどることができるようになっている．図1に示すスキーマを用いてスキーマ図を説明する．

From B Ret b_1, c_1 of B_C;

上記の式はクラスBを視点クラスとし，属性b_1とEVAであるB_Cで関連づけられているクラスCの属性c_1を検索する問い合わせ式である．スキーマ図は図2のようにBを始点にB_C(1：M)をたどりCにいたる木構造で表現される．節BのChildが節Cで，節CのFatherが節Bである．

汎化階層内のクラスも他のクラスと同様にスキーマ図で表せる．問い合わせ式に現われるサブクラスから，継承属性が直接属性として定義されるスーパクラスまでのす

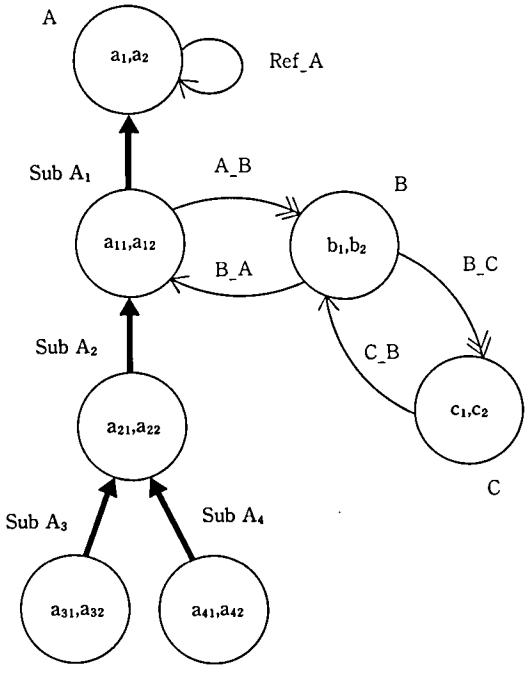


図1　スキーマ例
Fig. 1　Sample schema

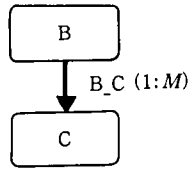


図2　単純なスキーマ図
Fig. 2　Simple schema graph

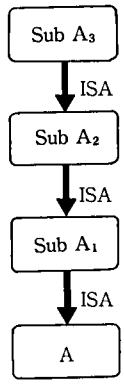


図 3　汎化階層のスキーマ図
Fig. 3　Schema graph of generalization hierarchy

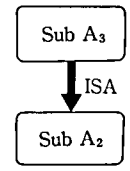


図 4　汎化階層の単純なスキーマ図
Fig. 4　Simple schema graph of generalization hierarchy

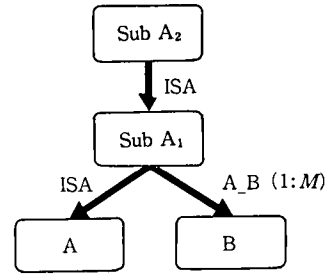


図 5　分岐スキーマ
Fig. 5　Branched schema graph

べてのクラスが節となる．サブクラスの節からスーパクラスの節へは，ISA (汎化) 関連としてスキーマ図で表現される．ISA 関連の逆関連は AS (特化) 関連であり，スーパクラスからサブクラスへの関連は AS 関連となる．

From Sub A_3 Ret a_{31}, a_{21}, a_1;

この式の属性の並びに現われる a 1 は，基底クラス A の直接属性が Sub A_3 クラスに継承された属性である．スキーマ図は，図 3 のように Sub A_3 を始点とし，Sub A_2, Sub A_1, A が ISA 関連で結ばれた木構造で構成される．上記の式の属性の並びから a_1 を除くと，図 4 のように Sub A_3 を始点とし Sub A_2 を節とする簡潔なスキーマ図となる．

一つのクラスから複数のクラスへアクセスする以下の式では，一つの節から複数の節への分岐が発生する．

From Sub A_2 Ret a_{21}, a_1, b_2 of A_B;

この式の場合には図 5 のように Sub A_2 を始点として Sub A_1, A, B が節となる．今までのスキーマ図と異なるのは，Sub A_1 から二つの節へ関連が結ばれていることである．節 Sub A_1 の Child は節 A であり，節 A の Sibling が節 B となる．節 A と節 B の Father は節 Sub A_1 である．

From A Ret a_1, a_1 of Ref_A;

クラス A からクラス A への再帰的アクセスである上記の式は，節 A から節 A への 1：M の関連等級を持ったスキーマ図で表現される．論理スキーマでは同一のクラスであっても，スキーマ図では異なる節と認識されるため，同一クラスを再帰的にアクセスする問い合わせ式は他の問い合わせ式と区別なく表現される．

6. SIM アクセスの論理モデル

あるクラスのエンティティ集合を表現するために“{ }”を用いると，クラスAのエンティティ集合は $\{a_1, a_2, \cdots, a_n\}$ で表現できる．図6のクラスAのエンティティ集合は $\{a_1, a_2, a_3\}$ であり，クラスBのエンティティ集合は $\{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8\}$ で表現できる．

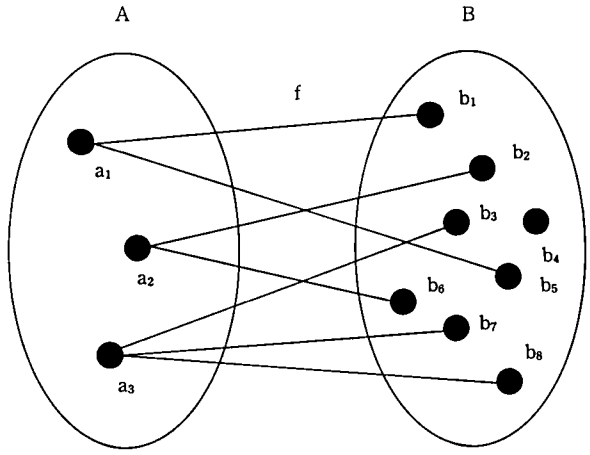


図 6　2項間のエンティティ
Fig. 6　Entities among binary classes

SIMのEVAは，関数型モデルのようにエンティティと関連している他のクラスのエンティティの部分集合を得ることのできる関数であると考えることができる．エンティティ a_i からEVAを通してアクセスできる他のクラスのエンティティ集合 $\{b_{i1}, b_{i2}, \cdots, b_{in}\}$ を $\{a_i \rightarrow \{b_{i1}, b_{i2}, \cdots, b_{in}\}\}$ として表現する．

クラスAとBの間のEVAを関数 f として考えると，Aのエンティティ a_i から f を通してアクセスされるクラスBのエンティティの集合 $\{a_i \rightarrow \{b_{i1}, b_{i2}, \cdots, b_{in}\}\}$ は式(6-1)で表される．

$$f(a_i) \equiv \{b_{i1}, b_{i2}, \cdots, b_{id_i}\} \equiv \sum_{j=1}^{d_i} b_{ij} \tag{6-1}$$

ここで d_i はあるエンティティにおけるEVAの持つ定義域の要素の最大数である．クラスAのエンティティ総数を e とすると，クラスAのすべてのエンティティから関連づけられているクラスBのエンティティ集合は式(6-2)のように表すことができる．

$$\sum_{i=1}^{e} f(a_i) = \sum_{i=1}^{e} \left(\sum_{j=1}^{d_i} b_{ij} \right) \tag{6-2}$$

したがって，クラスAのすべてのエンティティとそのエンティティと関連づけられているクラスBのエンティティは，式(6-3)で表すことができる．

$$\sum_{i=1}^{e} (a_i + f(a_i)) = \sum_{i=1}^{e} (a_i + \sum_{j=1}^{di} b_{ij}) \tag{6-3}$$

EVAの定義域の最大数 d_i はEVAのオプションによって異なる．EVAの対象となるクラスのエンティティ総数をDEとすると，EVAの定義域の要素の最大数は以下の

ようになる．

SV（単一値）	：$d_i=1$
MV（複数値）	：$d_i=\infty$
MV，MAX	：$d_i=$MAXの値
MV，DISTINCT	：$d_i=DE$
MV，UNIQUE	：$\sum_{i=1}^{e} d_i \leqq DE$

オプション句なしのMV EVAは重複したエンティティを含むBAGと呼ばれるエンティティ集合を表現し，DISTINCT句を持つMV EVAはSETと呼ばれる重複しないエンティティ集合を表現する．この論理モデルでは定義域をSETと定義し，一つのエンティティにおけるEVAの定義域の最大値はEVAの対象となるクラスのエンティティ総数として考える．

現在のSIMの最適化では，エンティティの論理アクセス回数を問い合わせ式のコストとして考慮しており，実際のエンティティ数や統計情報は考慮していない．したがって，物理的な係数はすべて定数として最適化の論理モデルを簡潔化している．

仮定1　すべてのクラスのエンティティ総数は同じEとする．

仮定2　すべての複数値EVAの持つ定義域の最大数をDとする．

n項間の関連をたどる場合のアクセスエンティティ総数は，式(6-4)で表現できる．

$$AE=\sum_{j=0}^{n-1} ED^j \quad (n\geqq 1) \tag{6-4}$$

したがって，二つのクラスAとBの2項間をEVAを経由して検索する問い合わせ式での論理アクセスエンティティの総数AE_{AB}は式(6-5)で表現できる．

$$AE_{AB}=E+DE=E(1+D) \tag{6-5}$$

上記のモデルではアクセス方法を考慮せず，クラスのエンティティをすべて逐次にアクセスすることを前提としている．ここでクラスのアクセスにインデックス検索を考慮し，以下の仮定を設定する．

仮定3　インデックス検索はクラスの全エンティティのT％を直接検索できる．

問い合わせ式にクラスBのエンティティを選択する条件があり，クラスBのインデックスがエンティティを選択するために利用可能とする．クラスAを始点とした問い合わせ式では，クラスBのエンティティはEVAを通してアクセスされるため，クラスBのインデックスは利用できない．またクラスBのエンティティが選択条件を満足するかどうかはエンティティを検索しなければ評価できない．したがって，クラスBに利用可能なインデックスがあっても，クラスAを始点としてアクセスする場合には，クラスBのアクセスエンティティ数はインデックスがない場合と同じである．

クラスBを始点とする問い合わせ式では，クラスBのアクセスにインデックスを利用できるので，クラスBのエンティティの論理アクセス回数は減少する．式(6-6)はクラスBを始点とした問い合わせ式で，インデックス検索を用いたアクセスエンティティ総数を示す．tはインデックスを用いることによって直接アクセスできる割合とする．

$$AE_{BA}=tE+DtE=tE(1+D)\ (0<t\leqq 1) \tag{6-6}$$

式(6-5)と(6-6)を比較すると，インデックス検索を採用した式(6-6)の方が論理アクセスエンティティ回数は少ないので低コストなアクセス経路であると言える．図6を用いて確認してみる．二つのクラスAとBをアクセスする問い合わせ式に，クラスBの奇数番号のエンティティが対象となる選択条件があるとする．クラスAを始点とした場合の論理アクセスエンティティの集合は，式(6-7)のように表すことができる．

$$\{\{a_1\rightarrow\{b_1, b_5\}\}, \{a_2\rightarrow\{b_2, b_6\}\}, \{a_3\rightarrow\{b_3, b_7, b_8\}\} \tag{6-7}$$

クラスBを始点とした論理アクセスエンティティを表すと以下のようになる．

$$\{\{b_1\rightarrow a_1\}, \{b_3\rightarrow a_3\}, \{b_5\rightarrow a_1\}, \{b_7\rightarrow a_3\}\} \tag{6-8}$$

式(6-7)と(6-8)を比較すると，式(6-7)では論理アクセスエンティティの総数は10であるが，式(6-8)では8となり，インデックス検索を用いてクラスBを始点としたアクセスが低コストな経路であることがわかる．EVAを基本とするSIMのアクセス経路選択では，スキーマ図の中でインデックス検索を用いて一番論理アクセスが少ない節を物理アクセスの始点とした経路が低コストである経路であり，効率の良いアクセス経路であると判断している．

7. 物理アクセス経路選択

物理的に効率よくアクセスするための経路選択では，以下の二点を考慮しなければならない．まず第一に各節の中から一番効率よくアクセスできる節を選択し物理的な始点として選択すること，第二に物理的な始点を中心とした物理的なアクセス経路をスキーマ図を基本にして構築することである．

7.1 物理アクセス始点の選択

スキーマ図の各節の中から物理アクセス始点を選択する際に評価の対象となる節は，その節の直接属性が問い合わせ式の選択条件に含まれる節で，かつB-treeかハッシュインデックスを持っている節である．

物理アクセス始点の選択はスキーマ図を論理始点から順に各節を評価し，評価の対象となる節ごとに最適なインデックスを選択し，それぞれの節の候補インデックスとする．それぞれの節の候補インデックスが決定されると，各節の候補インデックスの中から一番最適なインデックスを選択し，そのインデックスが定義されている節を物理アクセス始点とする．

7.2 最適アクセス経路の構築

スキーマ図の節の中から最適な物理アクセス始点選択の後，新しい節を始点として最適なアクセス経路を構築する．この新しい節を始点としたアクセス経路を問い合わせ図（Query Graph）と呼ぶ．論理アクセス始点と物理アクセス始点が同じ場合にはスキーマ図のアクセス経路が最適なアクセス経路であり，スキーマ図と問い合わせ図とは同じである．

新しい始点からスキーマ図の関連をたどり，新しい木構造を構築することで問い合わせ図を完成する．関連を逆にたどらなければならない節間は，関連と関連の等級を逆転しなければならない．関連の逆転は，EVAに逆関連が定義されている場合にはその逆関連を使用し，関連等級も逆関連の等級を用いる．

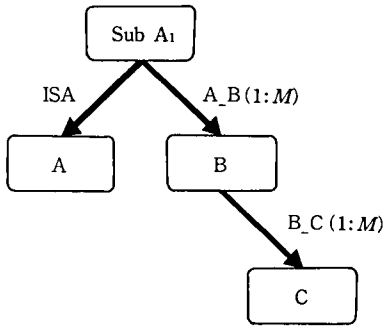


図 7　重複エンティティが発生するスキーマ図
Fig. 7　Schema graph of duplicate entities

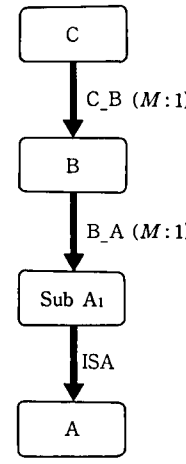


図 8　重複エンティティが発生する問い合わせ図
Fig. 8　Query graph of duplicate entities

たとえば図 2 のスキーマ図において節 C を新しい始点とした問い合わせ図を構築すると，節 C から節 B へ B_C の逆関連 C_B を通して関連を結び，関連の等級を 1：M から M：1 に変更する．したがって，節 C の Child は節 B に，節 B の Father を節 C に変更される．図 7 は下記の問い合わせ式のスキーマ図である．

From Sub A_1 Ret a1 where c_1 of B_C of A_B=10;

クラス C に c_1 をキーとするインデックスがあり，新しい始点として C が選択された場合の問い合わせ図が図 8 である．

問い合わせ式は視点クラスを持っているので，問い合わせ式の結果はスキーマ図のアクセス経路でアクセスした結果と同じ結果を保証する．スキーマ図での結果と同じ結果を保証するために問い合わせ図のアクセス経路で得た結果をスキーマ図の論理アクセス経路を通してアクセスした結果に変換して問い合わせ式の結果とする．

8. エンティティの等価性

クラス内のエンティティ集合は，オブジェクト指向データベースの概念であるオブジェクトの等価性を持っている．“すべての属性値が同じ二つのエンティティ“と“同じエンティティが二つ”は異なる意味を持っており，等価性とはすべての属性値が同じ二つのエンティティの存在を許すことである．すべての属性値が等しい複数のエンティティがクラス内に存在することは許されるが，同じエンティティがクラス内に重複して存在することは許されない．SIM は各エンティティごとに一意な値を持つサロゲート識別子によってオブジェクトの等価性を保証している．

SIM の問い合わせ式はエンティティ集合を結果として返すと考えることができる．したがって，問い合わせ式の結果として返されるエンティティの集合にも等価性が当てはまらなければならない．

問い合わせ図でアクセスされるエンティティの集合は SIM 内部で評価されるエンティティ集合であり，それが問い合わせ式の結果として評価されるかどうかは，問い

合わせ式に現われる属性の並びによって決定される．図6におけるクラスBを始点とした論理アクセスエンティティを示す式(6-8)において，問い合わせ式に現われる属性の並びにクラスBの属性が現われないとする．式(6-8)の論理アクセスエンティティは，SIMの内部で評価されるエンティティ集合であり，問い合わせ式の結果として評価されるエンティティ集合ではない．評価されるエンティティの組を一つのエンティティとして“[ ]”で表現する．式(6-8)によって評価されるエンティティ集合を論理アクセス経路に変換すると，式(8-1)のように表せる．

$$\{[a_1, b_1], [a_1, b_5], [a_3, b_3], [a_3, b_7]\} \tag{8-1}$$

式(8-1)はSIM内部で評価されるエンティティであり，ここから問い合わせ式の結果としては評価されないクラスBのエンティティを削除すると，式(8-2)のようになる．

$$\{[a_1], [a_1], [a_3], [a_3]\} \tag{8-2}$$

式(8-2)は問い合わせ式の結果として評価されるエンティティ集合である．a_1とa_3がともに2回現われており，同じサロゲート値を持つエンティティが複数存在することがわかる．したがって，問い合わせ式の結果のエンティティ集合の等価性が損なわれてしまっている．

SIMは最適なアクセス経路である問い合わせ図に従ってエンティティにアクセスし，スキーマ図のアクセス経路でアクセスしたように結果を返す．この結果，SIM内部で評価されるエンティティ集合では等価性が守られていても，結果として評価されるエンティティ集合では等価性を損なってしまうことがある．これはSIM内部で評価されるエンティティ集合と結果として返されるエンティティ集合が異なる問い合わせ式で，スキーマ図から問い合わせ図を構築する際に1：MまたはM：N関連等級を持つ節間の関連を逆転させることによって生じる．図6ではAからBへの1：Mの関連を逆転させ，BからAへM：1の関連になり，評価の対象となるエンティティ集合が異なるため等価性が損なわれる．

9. エンティティ等価性の保証

問い合わせ図の始点からスキーマ図の始点までの節を変換節と呼ぶ．等価性の損失は先にも述べたように，スキーマ図から問い合わせ図に変換した時に重複するエンティティが評価の対象となるエンティティ集合内に存在するために発生する．エンティティの重複は変換節でのみ発生する可能性があり，変換節以外の節では発生しない．図8の問い合わせ図では節Cと節B，節Sub A_1が変換節である．

すべての変換節が問い合わせ式の属性の並びに現われる場合には，SIM内部で評価されるエンティティ集合と問い合わせ式の結果として評価されるエンティティ集合が同じである．したがってエンティティは重複せず等価性は守られる．しかし，変換節内の節でFatherへM：1またはM：Nの関連等級を持ち，属性の並びに現われない節が変換節内に存在する場合には，エンティティの重複が発生する可能性がある．

変換節の中に同一汎化階層内のクラスが複数ある場合には，汎化階層内のクラス間ではエンティティの重複は発生しない．これは汎化階層がエンティティの部分集合を構成するもので，あるサブクラスに存在するエンティティはすべてのスーパクラスに

も存在する同じエンティティであるという理由からである．

SIM は変換節が発生し，変換節の一つの節から他の節へ $M:1$ または $M:N$ の関連等級がある場合には，重複エンティティが発生する可能性があると判断している．しかし問い合わせ図の視点におけるエンティティが唯一決定でき，変換節の物理視点から論理視点までのすべての節の関連等級が $M:1$ である場合は例外になる．この場合には問い合わせ図の視点のエンティティが1個存在し，それぞれの節にも1個のエンティティしか存在しないため，結果として評価されるエンティティも1個しか存在しない．

SIM は，重複エンティティが発生する可能性があると判断した場合には，問い合わせ式の実行を二つのフェーズに分けて実行する．第一フェーズでは問い合わせ図の始点から変換節のエンティティを評価し，変換節のすべてのエンティティのサロゲート識別子を作業用ファイルに書き込む．作業用ファイルには重複したサロゲート識別子が含まれるので，変換節のすべてのエンティティの評価が終了した段階で作業ファイルにある重複したサロゲート識別子を削除する．第二フェーズでは作業ファイルからサロゲート識別子を抜き出し，変換節のクラスのエンティティをサロゲートインデックスを用いてアクセスし，変換節以外の節にあるエンティティも評価する．変換節のエンティティの選択条件評価は，第一フェーズで行うことによって評価外のエンティティサロゲートを作業ファイルに書き込まないようにしている．

10. SIM の問い合わせ式最適化の今後の課題

現在の SIM の問い合わせ式最適化にはまだまだ解決しなければいけない問題点が残されているが，現在の問い合わせ式最適化の技術から考えると高いレベルの最適化機能を持っていると言える．しかし，より高度な問い合わせ式最適化機能を装備するためには，以下の機能に取り組む必要があるだろう．

先にも述べたように問い合わせ式最適化には，ある規則を規準にした最適化とコストを規準にした最適化がある．SIM は規則を規準にした最適化の方法を多く取り入れているが，コストを規準とした最適化はあまり考慮されていない．アクセス論理モデルではコストを考慮しているが，各クラスのエンティティ総数と定義域の最大数に定数を使っているため，静的な規準に従ったコストになっている．またここでは述べていないが，B-Tree インデックスの選択規準もインデックスのエントリ数やインデックスのレベルの深さ等が考慮されず，規則だけを規準にした選択を行っている．

スキーマ図と問い合わせ図では汎化階層内のクラスをそれぞれ独立した節として評価しているため，冗長な物理アクセスが発生する可能性がある．汎化階層内のエンティティは論理的には一つのエンティティである．したがって，問い合わせ式で評価されるクラスの中で最下層のサブクラスを節とすれば論理的には十分である．問い合わせ図では一つの節が物理アクセスの単位であるので，汎化階層内のエンティティが複数の物理ファイルに分散して格納されている場合には例外として物理ファイルごとに節を設定すればよい．

エンティティの等価性保証の方法における変換節での重複エンティティの可能性を判断する場合に，変換節内の節が問い合わせ式の属性の並びに現われるかどうかの判

断規準に曖昧さがある．また変換節が発生した場合に，評価後のエンティティ集合内におけるエンティティ順序とスキーマ図の経路でアクセスした場合の順序が同じ順序になるようにしている．しかしクラスにおけるエンティティ集合は順序のあるリストではなく，単なるSETもしくはBAGであることから視点クラスのエンティティ順序を保証する必要はない．

11. オブジェクト指向データベースにおける問い合わせ式最適化の課題

オブジェクト指向データベースの多くが構造面での論理データモデルとしてセマンティックモデルの持つ豊富な抽象化の概念を導入している．したがって，SIMは構造面ではオブジェクト指向データベースということができる．SIMの問い合わせ式最適化は構造面のアクセスにおける最適化であり，オブジェクト指向データベースでの構造面の最適化技術の一つの方向であることには間違いない．

オブジェクト指向データベースでは，オブジェクトの構造とそのオブジェクトに対する処理方法（メソッド）をカプセル化して一つのクラスに閉じ込め，利用者にはオブジェクトの持つ属性値を直接参照できないようにする情報隠蔽の機能がある．C++やSmallTalkから発展した初期のオブジェクト指向データベースでは，この情報隠蔽の機能を大切にしている．そのためデータベース管理システムでは，本来必須である操作言語を重視していない．しかし，純粋にデータベースから発展した最近のオブジェクト指向データベースでは操作言語を重視している．操作言語を用いる場合には，問い合わせ式最適化の段階で情報隠蔽の機能が壊れてしまう．オブジェクト指向データベースにおける操作言語の必要性は議論の段階であるが，筆者は操作言語が必要であるという見解に立ち，操作言語を用いる場合には問い合わせ式最適化の段階で情報隠蔽の機能が損失してもよいと考える．

オブジェクト指向データベースの操作言語では属性と同様にメソッドも扱えるようになる．しかしメソッドが問い合わせ式内で用いられると構造における関連以外に，メソッド間の関連も評価の対象になる．ここで問題になるのは，第一にメソッドが問い合わせ式の構文検査段階で評価可能かどうかということ，第二にメソッドで参照されるクラスや関連が物理アクセスの経路や節として評価可能であるかということ，である．

オブジェクト指向データベースではオブジェクトの状態変化を版（Version）として管理する機能がある．したがって，一つのオブジェクトでも複数の版が存在することになる．問い合わせ式で版の選択が可能であるとすると，一つのエンティティの複数の状態をたどるための最適化も考えなければならない．

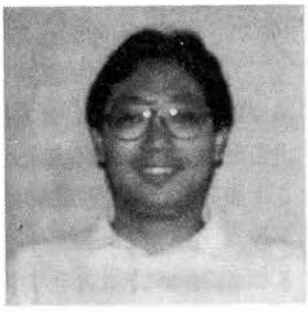
12. おわりに

本稿では，SIMの問い合わせ式最適化技術を基に複雑な論理構造における構造面での最適化を紹介し，オブジェクト指向データベースにおける問い合わせ式最適化の一つの方向を示した．オブジェクト指向データベースは機能面での要求が整った段階であり，機能を満足することがオブジェクト指向データベースの評価につながっていた．しかし，問い合わせ式最適化技術の議論が活発になり始めたこともあり，今後はオブ

ジェクト指向データベースを実装する段階での最適化技術の進歩が望まれるであろう.

参考文献 [1] Accredited Standards Committee X3, Information Processing Systems DBSSG/OODBTG Final Report, Sep., 1991.
[2] R. G. G. Cattell, *Object Data Management Object-Oriented and Extended Relatio-nal Database Systems,* Addison-wesley Publishing Company, 1991.
[3] C. J. Date, *An Introduction to Database Systems,* Vol 1, Fourth Edition. Addison-wesley Publishing Company.
[4] M. Hammer and D. McLeod, "Database Description with SDM: A Semantic Data Model", ACM TDS, Vol. 6, No. 3, 1981.
[5] J. G. Hughes, *Object-Oriented Databases,* Prentice Hall International Ltd.
[6] W. Kim, *Introduction to Object-Oriented Databases,* The MIT Press.
[7] Object Management Group (OMG), The OMG Object Model, draft 0.9, Sep., 1991.
[8] H. Pang, H, Lu and B. Ooi, "Query Processing in OODB", DASFAA'91
[9] G. M. Shaw and S. B. Zdonik, "Object-Oriented Queries: Equivalence and Optimization", DOOD'89.
[10] D. W. Shipman, "The Functional Data Model and the Data Language DAPLEX", ACM TDS, Vol. 6, No. 1, 1981.
[11] S. B. Zdonik and D. Maier, "Readings in Object-Oriented Database Systems", Morgan Kaufmann Publishers, Inc. 1990.
[12] S. B. Zdonik, "Query Optimization in Object-Oriented Databases", IEEE, 22nd Annual Hawaii Int. Conf. on System Science, 1989.

執筆者紹介 山 口 裕 久 (Hirohisa Yamaguchi)
1980年日本ユニシス(株)入社. 1985年よりデータベース開発保守担当. 現在システムプロダクト本部 ソフトウェア四部所属. Unisys Lake Forest 在中.



営業支援システムへの意味型データベースSIMの適用

An Application of the Semantic Database SIM to a Sales Activity Support System

森　良行, 小野寺　裕

要　約　情報系システムの構築が一般化しつつある中で，システムの柔軟性や生産性にどう応えるかが一つの焦点になっている．A社では，情報系システムとしての営業支援システムが開発され，すでに本稼働している．

本稿では，この営業支援システムで実現されている蓄積サブシステムと検索サブシステムに焦点を当て，蓄積サブシステムが，プログラミング言語を使用しない記述で開発できたこと，また検索サブシステムが利用者からの，その都度の多様な検索要求に応え，さらにスキーマの変更に影響されない汎用的なシステムとして構築できたことを述べる．

さらに，このシステム構築では営業支援システムの基盤となるデータベースに，UNISYS Aシリーズの意味型データベースであるSIM (Semantic Information Manager) が有効であったことを述べる．SIMを使用することにより，一貫したデータ主導型システム構築ができ，生産性の高いシステム構築/再構築を可能とする仕組みを提供してきたことを強調したい．

Abstract　Amid the on-going general tendency to create data processing-oriented systems at financial institutions, one of the greatest user interests has been how much flexibility and productivity those systems provide. The compny A has developed its own sales activity support system positioned as a data processing-oriented system.

As its core database system, this sales activity support system has adopted the semantic database for UNISYS A-series products, called the Semantic Information Manager (SIM). The use of SIM has made it possible to build a thoroughly data-oriented system and to create/recreate a highly productive system.

Focusing on the inquiry subsystem and updating subsystem both implemented in the sales activity support system, this paper emphasize the effectiveness of the semantic database; that is, the use of SIM has led to a successful development of the updating subsystem without using a programming language for process description and also of the inquiry subsystem whereby any data required for transactions handling which occurs at any moment can always be made available to end-users through inquiry according to their needs.

1. はじめに

情報系システムは，利用対象部門，利用目的および利用方法等の拡大に伴い，容易なシステムの構築/再構築を可能とする情報基盤が必要になっている．

情報系システムの必要条件の一つとして“変化する多様な情報要求に対する迅速な対応”がある．これは，ある情報が必要であると認識されてから，目に見える結果として出力されるまでの時間がきわめて短いことを意味し，そのような要求に対して，タイムリかつ適切に応ずることのできるシステム作りが必要となる．すなわち，多様な情報要求をあらかじめ定義し，それを満足するシステム構築で対応する従来型のシ

ステム作りではなく，要求自体が絶えず変わることを前提とし，それに応えうる容易なシステム作りが必要となる．

また，もう一つの必要条件として，“情報の価値への考慮”がある．これは，情報システムが社会・経済環境の変化に応じて，柔軟かつ機敏な対応が必要となっていることを示している．とくに，情報系システムでは基礎となるデータそのものが変化する．すなわち，データベースとして蓄積されるデータの意味や関連も変化し，絶えず鮮度の高いデータを迅速に取り込めるシステム作りが必要となる．

A社における営業支援システムは，情報系システムの一環として全社ベースで稼働している．このシステムは，基礎となるデータベースシステムに，UNISYS Aシリーズの意味型データベースSIM（Semantic Information Manager）を取り入れ，このような要件に応じたシステムを実現している．

2章では，営業支援システムの概要を紹介し，3章では，営業支援システムのデータベースに関して，SIMを用いることで，どのような考えや機能を使ってデータベースを構築しているかを紹介する．4，5章では，3章で紹介するデータベースを基盤に，営業支援システムのサブシステムである検索サブシステムと蓄積サブシステムについて，システムの構築/再構築における高生産性を達成するために実現した仕組みに焦点を当てて紹介する．

2. 営業支援システムの概要

2.1 システムの必要条件

営業支援システムは，営業部員が日々の営業活動を行うにあたって，見込み顧客の選定や顧客の取引状況の調査といった営業活動で必要とされる情報を，営業部員やそのスタッフ（以後，「利用者」という）がワークステーションを用いて，自由な視点で照会することをねらいとしたシステムである．

このようなねらいのもとで，システムに求められているのは，顧客情報や営業実績情報等を格納したデータベースをもとにして，その都度，必要に応じて利用者からの自由なデータ検索，加工に対応できるシステムの構築が迅速にできることである．さらに，経営戦略や市場の変化に伴い，取り扱うデータの種類や意味の変化に対応して，基礎となるデータベースや，そのデータを取り扱うプログラム群を容易に短期間で再構築できることも必要条件となる．

このような要件を実現するには，利用者からの自由な検索・加工に対応でき，利用者向きの画面制御や画面作成が容易に行える利用者支援機能，データベースのスキーマ変更に左右されずに，データベースの検索が可能な汎用的な検索処理機能や，ユーザがCOBOL等によるプログラミングを行わなくても，簡便にデータベースの更新処理が指定できるデータベース蓄積機能を備えたシステム構築が必要と考えた．

2.2 サブシステムの構成

営業支援システムは，前記のような要件に対応して，利用者支援サブシステム，検索サブシステム，蓄積サブシステムの三つのサブシステムから構成される．利用者支援サブシステムは，利用者による照会，抽出データの加工といった利用者操作を可能とする．検索サブシステムは，利用者の要求する情報をデータの種別や関係によりデ

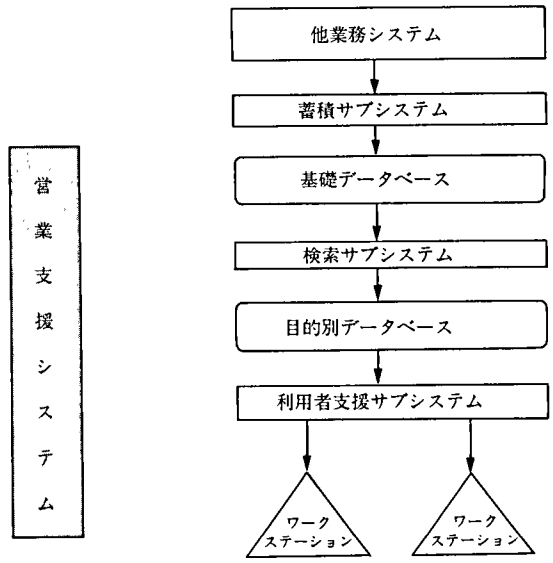


図 1　各サブシステム間のデータ関連図

Fig.1　Relation of data between each subsystems

ータベースから抽出して幅広い検索機能を可能にする．蓄積サブシステムは，データベースへのデータ蓄積を行うためのトランザクション・データを自/他ホストシステムより受けて，データベースに反映する．サブシステム間のデータ関連を図1に示す．つぎに，図2にもとづいて三つのサブシステムを説明する．

1)　蓄積サブシステム……これは，他業務システムより発生したトランザクションをリアルタイムで受信し，営業支援システムの基礎となるデータベース（基礎データベース）を更新するサブシステムである．トランザクション制御とデータベース更新プログラムからなる（図2）．

　他業務システムとのインタフェース処理を行うために，インタフェース形態を共通化して行えるユーザ・インタフェース・ライブラリプログラムを提供している．同一ホスト内であれば，このライブラリプログラムを経由して，トランザクションがこのサブシステムに取り込まれる．また，他ホストシステム上の他業務システムのトランザクションは，同一のライブラリプログラムからBNA(ホスト間通信のための制御系ソフトウェア）を経由してこのサブシステムに取り込まれる．一度このサブシステムに取り込まれたトランザクションは，このサブシステム内で保証し，障害時であっても再送する必要はない．このようなトランザクションデータの管理は，トランザクション制御プログラムが行っている．

　データベースの更新処理記述に対しては，データ蓄積という処理に限定した機能での簡易言語を提供している．この簡易言語による記述にもとづいて，実際にデータベースの更新処理を行うプログラム（データベース更新プログラム）が生成される．この更新処理記述については，5章で述べる．

2)　検索サブシステム……これは，利用者支援サブシステムから渡されるパラメタにより，利用者の指示した抽出項目や条件に従って基礎データベースからデータ

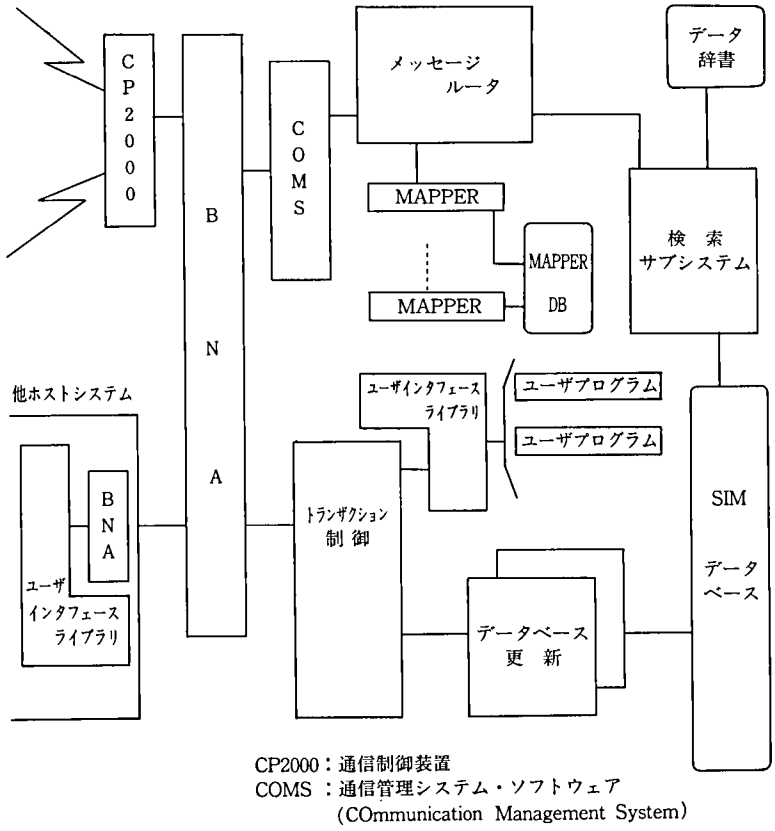


図 2 営業支援システムのソフトウェア構成図

Fig. 2 Structure of software in the Sales Activity Support system

を抽出し，利用目的別に設定されているデータベース（目的別データベース）に格納するサブシステムである．目的別データベースには，MAPPER データベースを使用している．利用者支援サブシステムとのインタフェースは，メッセージルータを通して行われる．

このサブシステムは，利用者支援サブシステムから渡されるパラメタをデータ辞書にもとづいて解釈実行することによってデータの抽出を行うので，利用者が抽出条件や項目を自由に設定できる非定形型照会に応えることができる．利用者から見れば，照会形態として，簡易操作による照会を可能にする定形型照会もあるが，検索サブシステムとしては，この区分けは存在せず，すべて非定形型照会として処理する．したがって，定形型照会画面の追加削除が発生しても，検索サブシステムには影響を与えない．

また，照会項目には，データベース上の実データ項目でない項目がある．これは，実データ項目より導出されるデータ（仮想項目）として，導出手順がデータ辞書上に登録されている項目で，利用者から見れば実データ項目と同様に取り扱える．

3) 利用者支援サブシステム……本サブシステムは，利用者のワークステーション

上の照会画面のインタフェース処理と，照会による検索要求での検索サブシステムとのインタフェース処理を行う．このサブシステムは，MAPPERを基にして構築されており，照会画面は，業務処理に適合した画面の組み合わせによって設定できる．

照会画面には，簡易操作で照会できる定形型照会画面と，自由な視点から項目を組み合わせて照会できる非定形型照会画面がある．定形型照会画面は，検索サブシステムに対して影響を及ぼすことなく，パラメタの設定により容易に作成できる．

また，このサブシステムでは利用者によって照会されたデータに対して，照会画面よりさらに加工・編集を行うことができ，すでに照会済みのデータと新たに照会されたデータを結び付けて一つの表を作成するといった操作がサブシステム内で可能となっている．

3. 営業支援システムのデータベース

3.1 データベースの階層化

A社における営業支援システムでは，二つのデータベースを階層的に使用している．一つは，利用者が必要とする情報の基礎となるデータで，主に他業務システムから発生するトランザクションデータによって，逐次更新されていくデータを持つ営業支援システムの基盤となるデータベースである．これを基礎データベースと呼ぶ．もう一つは，利用者の要求により，基礎データベースからデータを抽出した後に，利用者による加工・編集を可能にするために一時的に蓄えられるデータベースである．これを目的別データベースと呼ぶ．

基礎データベースは，データの本来の意味やデータ関連に重点をおいて管理され，業務処理に依存しないデータベースとして位置づけられる．これに対して，目的別データベースは，各利用者が操作して作成したデータを管理して，あたかも利用者固有のデータベースであるかのように取り扱われ，利用者支援の機能を有効に使用するためのデータベースとして位置づけられる．

このように階層的にデータベースを使用するためには，基礎データベースの各項目やデータ関連を目的別データベース上と同期して取り扱うことが必要となるし，コード値データや項目名等の漢字名表示や，データベース上にない仮想項目の設定も必要となるため，これらのデータを管理するためのデータ辞書（リポジトリ）が必要となる．営業支援システムではシステムの保守を容易にするため，処理手順やデータの管理を行うためのリポジトリとしてデータベースを使用している．これをデータ管理データベースと呼んでいる．

3.2 基礎データベース

3.2.1 基礎データベースの要件

基礎データベースは，営業支援システムの基礎となるデータベースであり，情報の提供を行う上で，業務処理が必要とする演算・加工等の基礎値として取り扱うことのできるデータを蓄えたデータベースとした．

利用者からの情報の見方は，データベース設計時には定まらず，したがって業務処

理に依存したデータベース設計をすることはできない．また，業務処理にとって必要なデータであるが，基礎値から導出されるデータは，データの整合性・冗長性からも基礎データベースに格納すべきでないと考えた．それぞれのデータ項目は，その意味と関連において，実世界（実務）での規則等にもとづいて分類され，データ間の関連が表現できるデータベースでなければならない．

基礎データベースは，蓄積サブシステムによって更新され，検索サブシステムによって照会される．蓄積サブシステムでは，開発・保守の観点からデータ関連の保守が容易に行え，かつデータ項目更新の整合性が守られるデータベース・システムを必要とする．また，検索サブシステムでは利用者の検索要求に従い，利用者支援サブシステムが生成したパラメタを解釈実行することで，照会された情報が得られなければならない．このために，データベースを操作する上で，処理対象を分類するための区分コードや，データ関連の意味を表す関連コードを取扱うためのプログラムロジックを組み込まなくても，データベース定義の情報を用いて処理できるようなデータベースが必要である．このような要件を満たすために，以下に述べるような特徴を持つ SIM を採用した．

3.2.2 SIM におけるデータ構造

SIM では，顧客とか商品といった同一の属性を持った実体（エンティティと呼ぶ）の集まりをクラスと呼ぶ．たとえば，顧客の集まりであれば顧客クラスと呼ぶ．この顧客の中でも，エンティティを分類して法人顧客や個人顧客に分けることができる．この分類された集まり（部分集合）を，SIM はサブクラスと呼んでいる．SIM では，データ項目をデータ値属性（DVA : Data Valued Attribute）と呼び，クラスとサブクラスにそれぞれ定義できる．一つのクラスが，複数のサブクラスに分類される構造をとる場合には，共通した属性に関してはクラス上で定義し，それぞれのサブクラスにはサブクラス特有の属性を定義することができる．クラス上で定義された属性は，その下に分類されたサブクラスにとっては継承される属性（継承属性）として取り扱われるので，そのサブクラスのエンティティを参照した時には，あたかもサブクラスの属性のように取り扱うことができる．たとえば，データベース操作の条件式にクラス名の修飾子なしに継承属性を指定できる．

また，このようなクラスとサブクラスが定義された時には，たとえば顧客全体が処理の対象となる場合には顧客クラスより参照し，個人顧客が処理の対象となる場合には個人顧客サブクラスより参照することになる．このように，参照する視点によって処理の対象となるエンティティの集合を限定することができる．このような構造を汎化階層構造と呼ぶ（図 3）．

汎化階層構造において，顧客から個人や法人に分類することを特化（Specialization）と呼び，逆に個人や法人から顧客としてまとめることを汎化（Generalization）と呼ぶ．この特化においては，とくに階層レベルでの制限はなく，サブシステムをさらに特化することができる．

SIM を用いることにより，データベースのデータ構造を汎化階層構造で定義できるので，参照する対象がデータベース操作段階で明確になり，プログラムのロジック上でこれらを判断する必要がなくなる．すなわち，データ参照における整合性が SIM に

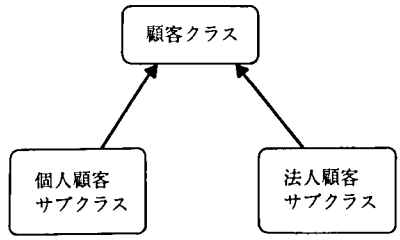


図 3　汎化階層構造例

Fig. 3　Generalization hierarchies

よって保証されているので，データベース操作に伴い参照されたエンティティが処理対象かどうかの判断ロジックが不要となる．また，これらの構造が業務処理に依存しないで，実世界の規則にもとづいて分類することができるので，データ指向の構造になりやすくなる．

3.2.3　SIMにおけるデータ関連表現

データの関連では，DMS IIのような従来型のデータベースは，ファイル間の関連付けを行うために種々の人工的な構造をスキーマ上に付加して，プログラムの手続きを用いてこの関連の取扱いを実現していた．従来型では，この関連についての必要な情報をユーザが作成し，データ辞書上に登録しなければならず，関連を取扱う手続きも作成しなければならない．SIMは，エンティティとエンティティの関連をエンティティ値属性（EVA：Entity Valued Attribute）によって定義できる．このため，スキーマ情報からデータの関連情報をデータ辞書上に取り込むことができ，プログラムの手続きも必要としない．この関連表現の違いを図4に示す．

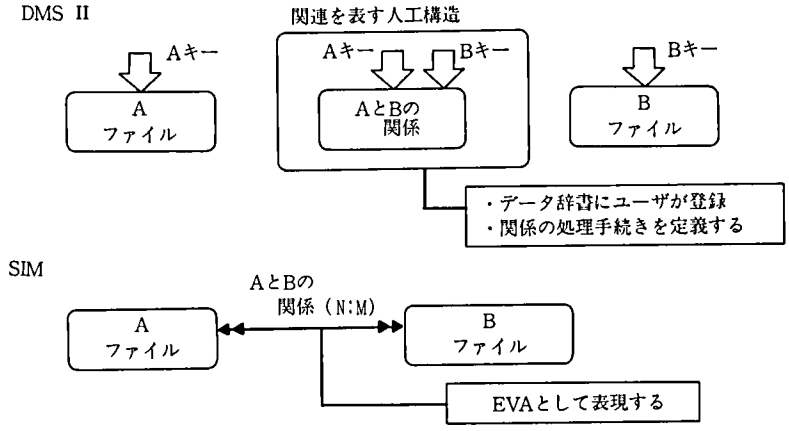


図 4　データ関連表現の違い

Fig. 4　Difference in the implimentation of data-relationship

これに加えて，同一ファイル内のレコード間の関連や，異なるファイル間の関連の中には，複数の異なる関係が存在する．たとえば，顧客の関係には関連取引先や世帯関係等がある．これらの関係はそれぞれ別の意味を持っている．データ指向を考えれば，このような関係は別々の関係として取扱うべきである．なぜなら，同じ関係とし

て取り扱うと，既存の関係が削除されたり，新しい関係が追加されたりした場合にデータベース上で定義する取り決めや，プログラム上の関係処理を行う手続きを変更することになり，その関連を取り扱うプログラムの変更が発生するからである．SIM を用いることで，このような関係を EVA としてスキーマ上明示的に定義できるので，データベース操作で指定する EVA を変更するだけで対応ができる．

エンティティとエンティティの関係表現は，データ参照の処理を行う場合の主要な要素となる．ファイル間にまたがるデータ参照がある場合に，そのファイル間の関係が定義されていないと取り出せないはずである．もし，このようなデータを取り出すことができるのであれば，処理プロセスの中でファイル間の関係処理を行っているからである．このようなデータは，処理プロセスが変わるとその関係が変わり，データの持つ意味も変化するのでデータ管理上の問題を起こす．それゆえに，あるデータ関連は，一意に関連が定まるように何らかの方法でデータ辞書上に定義されなければならない．SIM を用いない場合には，いろいろなデータ関連を定義すると，多くのプログラムで対応しなければならずプログラムが複雑になるので，主要な関連だけに限定して行っていた．SIM を使用することで，このようなデータ関連がデータベースシステム内で保証されるので，データベースのデータ関連操作は EVA の設定・解除を行うことで可能となる．

3.2.4 基礎データベースとしての SIM

基礎データベースは SIM を採用しているために，データ分類についてはスキーマ上で項目独立を前提とした汎化階層構造の表現が可能となり，特化によるデータ分類の記述がなされ，データの整合性の保証が得られている．さらに，データ関連については SIM の支援する属性を用いて関連表現されており，データ間の参照整合性も保証される．このため基礎データベースにおいては，スキーマ上でのデータ分類やデータ関連の定義によりデータのアクセス方法が判断できる．したがって，データアクセスをスキーマ情報にもとづいて行えるので，スキーマ変更に影響されない汎用的なアクセスが可能となる．このような SIM の利点を活かすことで，このデータベースはデータ中心を基本としたスキーマ設計がなされ，処理等により左右されないデータ指向のシステムを可能としている，

データベースの更新は，SIM のデータベース操作が項目独立を前提としているため，レコードや集団項目の単位での内部形式の再定義（COBOL の REDEFINES）がされないので，項目の配置を意識する必要がなく，項目処理は独立項目として取り扱える．また，データやデータ関連の参照整合性の保証が SIM によってなされていることにより，データ更新における処理を簡素化することができる．すなわち，データ関連やデータの参照整合性の判断処理記述を除外し，処理対象の項目に対する条件や演算，移送を基本とした簡易な記述によるデータベース操作を付加することで更新処理が記述できる簡易言語の提供が可能となった．この仕組みについては 5 章で述べる．

データベースの検索においては，スキーマ定義を情報として取り込むことにより，クラス/サブクラス間のデータ関連やデータ分類が明確となるので，それにもとづいてデータをアクセスすることが可能となる．とくに，検索条件に指定する項目が，DMS II のように索引テーブルのキー項目でなければならないといった構造上の制約はない

ので，データベース操作上与えられた条件項目の属するクラスを意識するだけでよい．このことにより，データ検索では，照会する項目の属するクラスと，複数クラス間の関連を判断することで，与えられた条件を指定してデータを検索することができる．したがって，スキーマ情報と仮想項目定義等の補助情報を取り込むことで汎用的な処理が可能となっている．この仕組みについては4章で述べる．

このように基礎データベースにSIMを用いることで，システムの変更や開発に容易に応えられる蓄積サブシステムと，利用者からの検索要求にいつでも応えられ，スキーマ変更に影響されない汎用的な検索サブシステムの構築が可能となった．

3.3 目的別データベース

目的別データベースは，利用者の要求により基礎データベースから抽出されたデータを一時的に保存しておき，その保存されたデータに対して，利用者が二次加工および編集を行うために提供しているデータベースである．すなわち，利用者支援という観点から，各利用者が照会した抽出データをこのデータベースに格納して，利用者のデータ操作のために，あたかも個々のデータベースが提供されているかのように取り扱われてるデータベースである．このデータベースには，MAPPERデータベースが使用されているので，利用者はMAPPERの利用者インタフェースの機能を使用して，これらの二次加工・編集等の処理を独自に行える．

3.4 データ管理データベース

基礎データベースと目的別データベースで取り扱われる項目は，データ項目の属性という点で考えれば，双方のデータベースで一貫している必要がある．このデータ属性の一元管理を実現しているのが，データ管理データベースである．目的別データベースは，基礎データベースの付随的なデータベースであるので，基礎データベースに対するデータの属性変更をデータ管理データベース上で行うことで，目的別データベースに反映する仕組みを用意した．さらに，データ管理データベースは，システムの開発・保守といった作業に対する生産性向上を図るための仕組みへのデータ提供にも使用される．

たとえば，蓄積サブシステムにおいては，蓄積処理記述上で使用されるトランザクション項目等のデータ項目の一元管理や，処理ロジックの部品化を行うためのモジュール管理に使用される．検索サブシステムでは，検索処理を行う上でのアクセス経路分析や仮想項目処理を行うために，基礎データベースのスキーマ情報，データ項目の属性，仮想項目情報をこのデータベース上より参照する．また，関連キーを使用したデータの絞り込みが必要とされる関係において，結合キー（JOIN KEY）を用いて関連表現しているので，その関連情報をこのデータ管理データベース上で管理している．データ管理データベースでは，以下のような情報が管理されている．

1) 蓄積サブシステムで用いる辞書情報
   - データベース更新ライブラリのテーブル定義記述に関する情報
   - テーブル定義記述で用いるデータ項目の情報
   - 日付のような変動パラメタ項目の情報
   - 定数値の情報
   - データベース更新ライブラリで使用する条件式情報

2) 検索サブシステムで用いる辞書情報
- デコード項目（ある数値と表示文字の変換，たとえば店番と店名）の情報
- 端末のヘルプ要求に対するヘルプ・メッセージの情報
- 項目のグループ管理情報
- 仮想項目の情報
- 結合キーとクラス関連情報

3) 全システム共通に用いる辞書情報
- 基礎データベースの構成要素の定義に関するスキーマ情報

4. 検索サブシステム

4.1 検索サブシステムのねらい

検索サブシステムの主な機能は，利用者が欲しい情報を指定された条件で適時に参照できることである．このような機能を実現する上で，本サブシステムは，利用者の要求にもとづいて個別のプログラムを作成することなく，汎用的なデータの抽出を可能にした仕組みを持ったシステムでなければならない．

検索サブシステムは，利用者支援サブシステム内において定形型画面処理（決められた業務画面）となっていても，汎用的な抽出処理を行っている．このため利用者支援サブシステムにおいて，新たな定形型画面処理が追加されたり変更されたりしても，検索サブシステムではプログラムの変更を必要としない．

4.2 検索サブシステムの処理の流れ

検索サブシステムは，MAPPER 環境(利用者支援サブシステム)から渡される検索要求のパラメタにより，基礎データベースに対する検索制御や，検索処理（抽出処理および編集）を行っている．すなわち，あらかじめ決められた検索処理が作成されているのではなく，パラメタによって動的に検索方法が決められ，それに応じて抽出・編集の処理を行っている．この処理の流れを図5に示す．

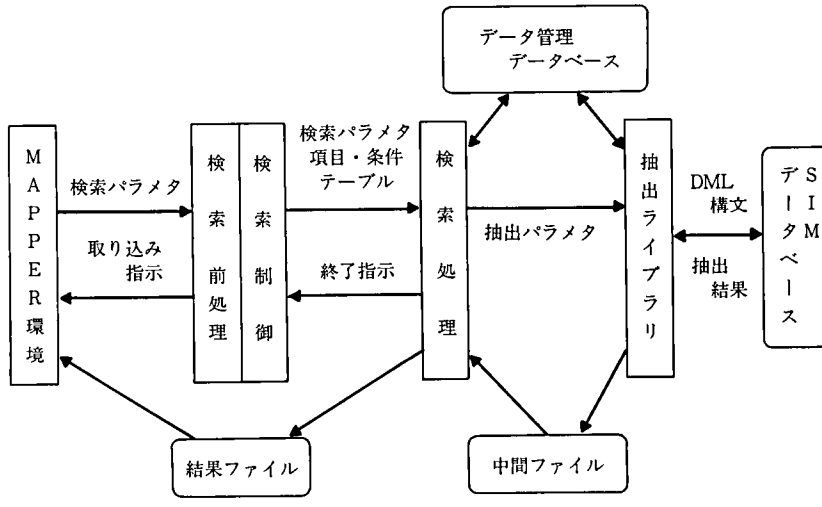


図 5 検索サブシステムの処理の流れ

Fig. 5 Process flow of inquiry subsystem

図5の処理の流れにおいて，検索前処理は，MAPPER環境より起動されるプログラムで，MAPPERより受けた検索パラメタを検索制御に渡す．検索制御は，検索パラメタの仲介を行って検索処理のプログラムに渡す．検索処理は，検索パラメタより抽出するためのパラメタを作成して抽出ライブラリを呼び出す．抽出ライブラリは，渡されたパラメタをもとにしてデータベース操作言語（DML：Data Manipulation Language）を生成して，SIMに対してのデータベース操作を行い，データを抽出してファイルを作成する．このSIMへのアクセスは，あらかじめ決められた構文で実行するのではなく，動的にその操作を決定してデータベースをアクセスする．ここで抽出されたファイルが中間ファイルとなる．検索処理は，この中間ファイルをもとにして，要求された書式に編集して結果ファイルを作成する．このとき，仮想項目があれば，その演算処理等を行い結果ファイルを作成する．結果ファイルが作成されると，検索前処理のプログラムに終了指示が渡され，MAPPER（目的別データベース）上への取り込み操作が行われる．

このような処理の流れの中でパラメタとして渡される情報は，出力項目と検索条件だけである．この二つの要素を基にして，検索処理のプログラムは抽出パラメタを作成している．抽出パラメタは，視点クラス（どのクラスを基にして参照するか），アクセス経路，抽出順序を設定している．

4.3 抽出パラメタの作成

MAPPER環境から送られてくる検索パラメタの中で，SIMインタフェースに使用できる情報は出力項目と検索条件である．この二つの要素は，それぞれ項目テーブルと条件テーブルに展開される．

抽出パラメタ作成過程では，視点クラス，アクセス経路，抽出順序を自動的に決定する．この決定には，アクセス経路の分析に使用されるクラス間の関係が一つしか存在しないことと，検索条件の多いクラス/サブクラスを優先するということが前提となっている．

4.3.1 抽出経路選択

出力項目，もしくは条件に指定されたクラス/サブクラスの中から，どのクラス/サブクラスよりアクセスを開始するかを決定する．データ辞書に登録された内容からEVA，結合キー（JOIN KEY）でのデータ関連表現を調べて，必要となるクラス/サブクラスへの関係が存在しているか否かを検査する．一つでも孤立してしまうクラス/サブクラスが存在した場合には，その旨を結果として返して処理は中断する．これは，照会データの中に関連を持たない項目が含まれていることを示している．

アクセスを開始するクラス/サブクラスを選択するために，そのクラス/サブクラスに対する条件の数を使用する．たとえば，クラスAに該当する条件が三つ，クラスBに該当する条件が二つであれば，アクセス順は，AからBになる．条件の数が等しい時は，アクセスの際に索引テーブルを使用できるもの（アクセス効率の良いもの）を優先する．条件の多い順にクラス/サブクラスを並べ，早い段階ですべての条件が満たされるような，クラス関連の木構造を選ぶ．この結果を蓄えたテーブルを経路テーブルと呼んでいる．経路テーブルは，抽出パラメタ作成処理を通して使用される最も重要なテーブルであり，処理の流れを決定する．テーブル上に存在するクラス/サブクラ

スがそのままDML上の視点クラスとして使用される場合も多い．

4.3.2 抽出パラメタ

抽出パラメタは，項目テーブル，条件テーブル，経路テーブルをもとに作成される．これは，DMLの構造に対応する情報で構成されている．すなわち，視点クラス情報，結合キー情報，目的属性情報，条件（大域選択）情報の四つに分けられる．

1) 視点クラス情報……視点クラスは，経路テーブル上に存在するクラス/サブクラスと，抽出対象属性の存在するクラス/サブクラスの中から選ばれる．そのサブクラスは，汎化階層上での下層（特化）レベルが優先されるが，2方向以上の特化されている情報が抽出される場合には，各サブクラスの共通した親クラス（スーパクラス）が選択される．

たとえば，図6の構造で抽出対象属性が，クラスA，サブクラスA2，A4の属性であれば，視点クラスはサブクラスA4となる．また，これに加えてサブクラスA5の属性も抽出される時は，サブクラスA2が視点クラスとなる．

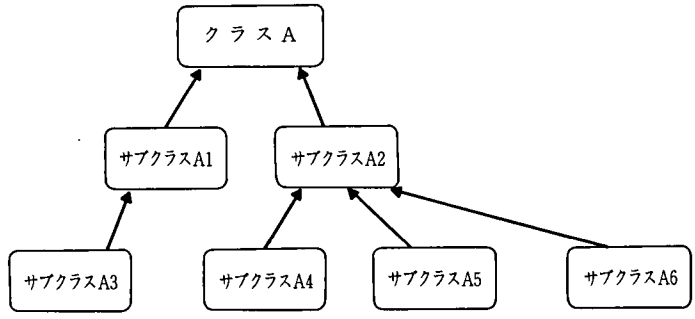


図6 汎化階層構造例

Fig. 6 Example of generalization hierarchies

2) 結合キー情報……抽出対象となるクラスの相互関連情報に，結合キーが存在する場合には，その結合キーの属性をクラスの属性参照時に含める必要がある．このため，その属性情報を付加する．

3) 目的属性情報……抽出対象となる属性が，視点クラスから見て上層の属性（汎化）であれば，直接属性か，あるいは継承属性なので修飾は不要である．しかし，視点クラスよりも下層のサブクラスの属性ならば，サブクラス名の修飾が必要となる．また，EVA経由で他の汎化階層のクラスの属性を抽出するのであれば，その経由するEVA名の修飾が必要となる．これらの修飾情報を目的属性情報として付加する．

4) 条件情報……視点クラスからアクセスできる範囲に存在する属性の条件をDML構文に変換しパラメタとする．

4.4 抽出処理

抽出ライブラリは，抽出パラメタよりDMLを作成し，これを使用してSIMデータベースのアクセスを行う．SIMによるデータの参照形式には，構造化形式，表形式，混成形式（構造化と表の混合）があるが，MAPPERによる出力形式が表形式となって

図 7　抽出操作例のクラス関係

Fig. 7　Relationships of classes in the example of extraction

いるので，抽出ライブラリでは表形式による参照を行っている．

たとえば，図7のような関係のクラスの属性を抽出する場合に，A_B，B_Cの関係は，それぞれ1：Nとして，クラスA，B，Cの順で抽出が行われるとする．これらの関係が，EVAで結び付けられている場合には，クラスAの属性を参照するDML構文にEVAの修飾を付加してクラスB，Cの属性も指定できるので，表の行単位にクラスA，B，Cのそれぞれの属性が並列に並べられた形式で参照される．

しかし，これらの関係が，結合キーで結び付けられている場合には，クラスA，B，CそれぞれにDMLを作成し，それぞれに対しての参照を行う必要がある．このため，まずクラスAの一つのエンティティを参照し，次にそのエンティティの属性値をもとにして（結合キー）クラスBを参照する．クラスBの参照されたエンティティの属性値をもとにして，クラスCを参照する．この条件値に一致したクラスCのエンティティがなくなると，クラスBの次のエンティティを参照し，それに関連するクラスCのエンティティを参照する．クラスBの参照すべきエンティティがなくなると，クラスAの次のエンティティを参照する．このような形式で，各クラスの情報が抽出される．また，この抽出方式においても，プログラム的には表形式でクラスA，B，Cの属性を並べることになる．この場合に，クラスB，Cのエンティティが存在しなくても，クラスAの属性は抽出され，クラスB，Cの属性はナル値として取り扱う．

抽出ライブラリは，各表の1行を1レコードとして中間ファイルに出力している．

4.5 編集加工処理

検索処理プログラムは，抽出ライブラリによって作成された中間ファイルをもとに，編集処理と仮想項目の算出処理を行い，MAPPER環境へのファイルに変換する．このとき，同時に見出しの作成も行っている．

5. 蓄積サブシステム

5.1 蓄積サブシステムのねらい

蓄積サブシステムは，他業務システムからのトランザクション・データをもとにして，SIMデータベースを更新している．このために他業務システムの変更にともない，トランザクション・データの変更があれば，データベースの更新処理の変更が必要となる．また，データベースのスキーマ変更・拡張からも影響を受ける．蓄積サブシステムは，このようなデータベースの更新処理の変更に対して，柔軟かつ迅速な対応ができるようにする必要がある．

このような要件に対して，蓄積サブシステムでは定義型の簡易言語を提供して，基礎データベースを更新するための記述をプログラミング言語を使用せずに記述できるようにし，この定義よりデータベース更新プログラムを生成する仕組みを構築した．

この簡易言語の記述は，デシジョン・テーブル形式で表現され，テーブル定義と呼ぶ．テーブル定義は，蓄積処理に対するデータベース操作の定式化にもとづいて手順を指示するものである．このテーブル定義を用いることにより，データベース操作の前提知識がなくても更新処理記述ができる．

テーブル定義で記述された手順は，データ管理データベース上に登録される．ここに登録されたテーブル定義は業務処理ごとに指示された単位で，データベース更新ライブラリとしてALGOL言語のプログラムに変換される．

5.2 データベース・インタフェース

テーブル定義をもとにして作成されたプログラムは，データベースのインタフェースに，親言語インタフェースではなく，データベース操作言語(DML)を使用している．これは，親言語を使用した場合には，プログラム生成時にSIMのデータベースを必要とするため生成環境の問題が発生するからである．

テーブル定義によって生成されるプログラムは，データベース操作言語を用いるために，以下の三つのフェーズでSIM操作のための処理を行っている．

1) データベース更新ライブラリ生成時……テーブル定義記述が，作成・変更された時に，データ管理データベースを参照しながらデータベース項目，データベース選択条件および操作種別等の一覧テーブルを作成して，生成されるALGOL原始ファイルに取り込む．図8に示すように，この時点ではSIMソフトウェアとのインタフェース処理は行わない．

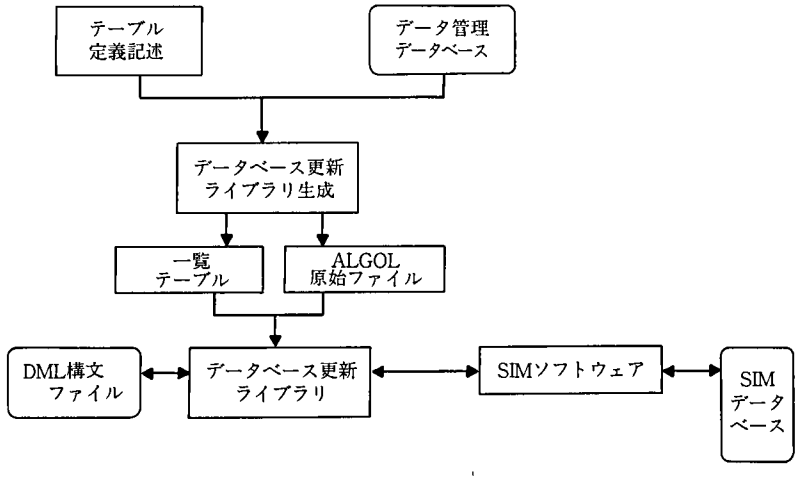


図 8 生成の流れ図

Fig. 8 Generation flow

2) ライブラリ呼び出し時のDMLの初期化……ライブラリの立ち上がり時に，1)で作成した一覧テーブルよりDML構文ファイルを作成する．ライブラリ呼び出し時に操作されるDMLが初期化されていない場合には，該当のDML構文を取り出し，SIMとのインタフェース処理を行いDMLを初期化する．このときSIMによって操作番号が割り当てられる．これは，使用される構文単位に行われる．

3)　初期化済みのDMLの使用……ライブラリ呼び出し時に操作されるDMLがすでに初期化されている場合には，そのDMLに割り当てられた操作番号を使用してSIMデータベースの操作を行う．

テーブル定義上でのデータベース操作は，プログラム生成時に，指定された処理指令が，一つまたは複数のデータベース操作文（DML構文）に置き換えられる．また，これらのDML構文を取り扱う処理は共通した手続きによって行われるので，テーブル定義上で記述されるデータベース処理指令は，この手続きを呼び出す指令に置き換えられることになる．

テーブル定義上で記述できるデータベース操作を以下に示す．

処理指令の略号	処理内容
CRT	エンティティの作成指示
REF	エンティティの参照
STO	エンティティの格納
CHS	サブクラスの変更
DEL	エンティティの削除（単一）
DLG	エンティティの削除（グループ）
INC	EVAの作成（単一）
EXC	EVAの削除（単一）
ING	EVAの作成（グループ）
EXG	EVAの削除（グループ）
CHE	EVAの変更（グループ）

5.3　テーブル定義の概要

テーブル定義では，手順だけを記述し宣言は存在しない．宣言に相当する部分については，すべてデータ管理データベース上に登録されていることが前提となる．このため，テーブル定義上で使用される項目は，すべてデータ管理データベース上に登録し，このとき採番された項目番号を使用して記述する．

手順の記述には，データベースのエンティティを確定させる処理(前処理)，データベースの各項目への移送に関する処理(項目処理)，データベースの更新指示を行う処理（後処理）がある．これらの手順は，トランザクション・データの処理種別単位（たとえば取引種別単位）に記述し，データ管理データベース上に登録する．

5.3.1　テーブル記述

各手順の記述は複数のテーブル記述より構成される．一つのテーブルは複数の記述によって構成される．テーブルは文章表現の段落に相当し，記述は文にあたる．記述については，条件記述と動作記述の2種類がある．

一つのテーブル内には，ある条件に対してある動作が行われ，条件と動作の組み合わせによってテーブルが構成される．表1にテーブルの構成例を示す．

表1では，条件Aを満足する場合には動作Cを実行し，条件Bを満足する場合には動作Dを実行する．条件A，B共に満足しない場合には動作Eを実行する．テーブル記述上，条件記述の組み合わせによりその動作を振り分ける．各条件の組み合わせに論理識別欄を用いて行い，組み合わせは1テーブル当たり5通り指定できる．

表1　テーブル構成例
Table 1　Example of table structure

	論理識別					
条件　A	T	T	F	F		(条件記述)
条件　B	F	T	T	F		(条件記述)
動作　C	○	○				(動作記述)
動作　D		○	○			(動作記述)
動作　E				○		(動作記述)

5.3.2　モジュール登録

二つ以上の処理種別の手順の記述において同じテーブル記述がある場合に，そのテーブル記述をモジュールとして登録しておき，各手順の記述よりこれらのモジュールを呼び出すことができる．また，モジュールから別のモジュールを呼び出すこともできる．

共通処理のテーブル定義をモジュールとして登録することにより，さまざまな処理モジュールが作成できるので各処理の部品化が可能となる．このことにより，新たな手順の記述が発生しても，部品化された処理のモジュールを組み合わせることで比較的短時間に対応できる．表2にテーブル定義の記述例を示す．

表2　テーブル記述の例
Table 2　Example of table description

区分	処理	項目番号	記号	項目番号	記号	項目番号	継続	論理フラグ
条件	REF	125.00.		125.00.1001	=	980.00.0211	1	. . . . . ①
条件				125.00.1002	=	980.00.0212	1	. . . . . ②
条件				125.00.1003	=	980.00.0213		.1.2. . . . ③
動作		990.00.0210	=	"1"				.1. . . . . ④
動作	CRT	125.00.						. .1. . . . ⑤
動作		990.00.0210	=	"2"				. .1. . . . ⑥

各欄の説明：
区分　：条件記述か，動作記述かを示す．
処理　データベースへの処理指令．上記の例では，REF はクラスからの読み込み，CRT はクラスへの新規エンティティの作成指示を示す．
項目番号：〈クラス番号〉．〈サブクラス番号〉．〈項目連番〉からなる．クラス番号の上二桁が 98 ならトランザクション項目，99 は作業項目，その他はデータベース項目を示す．
記号　演算子を示す．(,), +, −, *, /, =, >, <, L(LEQ), G(GEQ), N(NEQ)等
継続　：記述が次の行にまで継続するなら，1．
論理フラグ：1 は論理値の真，2 は偽を示す．

上記表の説明：
① 125.00 のクラスを対象に，参照条件としてデータベース項目の 125.00.1001 がトランザクション項目の 980.00.0211 の値と等しく，かつ，
② データベース項目の 125.00.1002 がトランザクション項目の 980.00.0212 の値と等しく，かつ，
③ データベース項目の 125.00.1003 がトランザクション項目の 980.00.0213 の値と等しいか．条件を満足するエンティティがあれば動作④を，なければ動作⑤，⑥を行う．
④ 作業項目 990.00.0210 に"1"を移送する．
⑤ クラス 125.00 に，新規のエンティティの作成指示を行う．
⑥ 作業項目 990.00.0210 に"2"を移送する．

6. おわりに

本稿の営業支援システムの紹介は，意味型データベースSIMを基礎として，自由な検索を可能にするMAPPERを組み合わせたデータ主導型システムの構築例である．この営業支援システムに使用されたSIMは，日本ではじめての実システムへの適用であったが，ここで述べたようなシステム構築支援ツールが作成できたのは，SIMの提供する機能によるところが大きいと考える．SIMは，オブジェクト指向の考えを取り入れたプロダクトであり，オブジェクト指向データベースへの今後の拡張が期待される．本稿は，高度に発展したリポジトリにもとづき，システム開発・保守の生産性を高めようとする今日のシステム作りの参考になるものと思う．

参考文献　[1] J. P. Thompson, "Data with Semantics", Van Norstrand reinhold, 1988.
[2] D. R. Howe "Data anlysis for Database Design" Edward Arnold, 1983.
[3] "Research Foundation in Object-oriented and Semantic Database systems", ALFONSOF, CARDENAS/DENNIS MCLEOD, Prentice-Hall, 1990.

執筆者紹介　森　良行 (Yoshiyuki Mori)
昭和28年生．50年上智大学理工学部数学科卒業．同年日本ユニシス(株)入社．金融機関の客先サービスに従事した後，Aシリーズの利用技術サービスに従事する．現在，システム技術本部利用技術二部に所属．

小野寺　裕 (Yutaka Onodera)
昭和36年生．59年慶応義塾大学商学部卒業．同年日本ユニシス(株)入社．金融機関の勘定系システムの導入・保守に従事した後，営業支援システムの開発・保守に従事する．現在，金融システム第二本部システム六部に所属．

真のソリューション(解決策)を目指したASDF

1. はじめに

日本ユニシスは90年代の情報システムの方向性を『複数の異機種コンピュータが持つ情報を，ネットワークを通してエンドユーザが容易に利用できる環境』であるととらえ，これをインフォメーション・ネットワークと定義している．そして，それを実現するための指針としてUA（Unisys Architecture）を提唱した．今回このUAにそって，具体的な情報システムの構築環境を実現するための体系（フレームワーク）であるASDF（Advanced Solution Development Framework)を発表した．

2. ビジネス活動と情報技術の融合

今企業を取り巻く環境には，国際化，競争の激化，規制の緩和，高齢化等，各種の大きな変化が起きている．このような環境の変化に迅速かつ柔軟に対応できることが，企業経営の大きな課題になってきている．そしてこのような環境変化は，ビジネスの一線での迅速な判断の重要性を増大させており，組織の活性化の意図も含めた権限委譲が進むなか，組織活動が集権型から分権型へと移行してきている．

一方，企業経営において情報システムが重要なインフラストラクチャになってきている今日，その情報システムを支える情報技術の基盤を確立することが経営を左右する戦略的課題になりつつある．情報システムが企業のなかで真に効果的であるためには，企業組織の構造を濃密に反映した情報基盤の整備が求められている．企業組織が分権化の方向に進むなか，企業全体としての整合性と一貫性の保障を前提とした上での情報基盤のより一層の分散化が必至の方向となってきている．

3. 情報技術の変化

80年代はマイクロ・コンピュータの急激な発達の時代であり，オフィス環境に数多くのパソコンやワークステーション，LAN等が導入され，情報基盤の分散化が本格化した．90年代に入ってこの傾向はさらに加速化すると同時に，メインフレームでの基幹処理を含めた企業全体にわたる情報システムの統合性・一貫性が求められてきている．分散処理を前提にしつつも，相互運用性，透過性，共通性を基盤として，全体として統合化された情報処理環境が必要になってきている．そこでは分散配置されたコンピュータがネットワークで接続され，あたかも単一システムであるかのように協調・連携することが必要である．

日本ユニシスはこのような情報基盤の進展の中での指針・提案として，1990年に情報アーキテクチャUAを発表した．UAでは真にユーザの立場に立った90年代の情報基盤を，エンドユーザからシステムの構造を意識させないオープンでかつ統合的な情報処理環境『インフォメーション・ネットワーク』であると考えている．そこでは分散環境を実現するハードウェア・プラットフォームを，インフォメーション・ハブ，分散サーバ，ワークステーションの三つの役割に整理する．そしてその環境を構成するソフトウェアの機能を五つの分野（アプリケーション・サービス，インフォメーション管理サービス，システム間接続サービス，分散システム・サービス，システム管理サービス）に体系化し，そのサービスの機能とインタフェースを規定している．

また，日本ユニシスが提供する具体的なソフトウェアを，順守する規約が国際標準に基づくもの，日本ユニシス固有のもの，業界標準のものにより，それぞれオープン・クラス，プレミアム・クラス，共存クラスの三つのクラスに位置づけている（図1)．

これらのことは，統合化された分散処理のための情報基盤を，長期的な展望に立ちつつ確実かつ容易に構築できることを意味する．

4. 次世代のシステム構築技術

UAは90年代の情報基盤として統合化された分散システム環境を提唱しているが，その上に築かれるユーザ・システムの開発に当たってどのようなアプローチが必要であろうか．次世代のシス

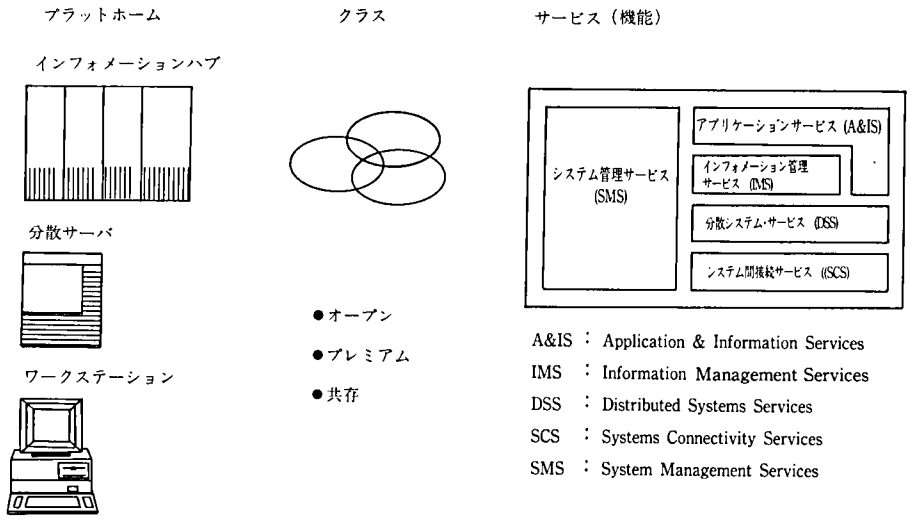


図1　ユニシス・アーキテクチャ

テム構築技術には，従来目標とされていた高生産性といったことに加えて，次のことが求められている．

1) 経営戦略と情報技術の連携……経営戦略の中で導き出された課題と解決策を，最新の情報技術を用いて速やかにかつ確実に実現できなければならない．

2) 最小限の開発……戦略的に最重要な解決策の実現に集中することが重要であり，そのためには可能な限り開発しないで済ませることが必要である．このことは大きな負担となっている保守作業を軽減することにもつながる．

3) 変化への即応……後の変更・拡張に即応できるよう，柔軟性の高い簡素なシステムとして構築すること，また課題を小さい単位に分けて構築することが大切である．そうすることにより，全体的な完成を待たず，重要なものから部分的に実現することも可能になる．

4) 優先順位の決定……開発の優先順位を明確にし，経営戦略上の急ぐ部分から着手する．また常に本質的な核となる部分からモデル化し，詳細は可能なかぎり実現時期に近い時点（ジャスト・イン・タイム）で仕様化することが重要である．

5) エンドユーザの参加……品質の高い効果的なシステムを構築するためには，構築のすべての段階への情報の最終的活用者であるエンドユーザの参加が重要である．また「最小限の開発」や「変化への即応」を達成するためにも，エンドユーザ自身が主体性を持って課題の解決を図れるよう，エンドユーザ・コンピューティング環境を整備することも必須である．

6) 既存システムとの共存・共生……以上のことを実現するには，すでに稼働している既存システムを有効に活用することが大変重要である．ユーザ・インタフェースの改良等の部分的な手直しも含めて，既存システムを積極的に活用することによって，始めて戦略的に重要な部分への投資の集約が可能になる．

5. ASDF (Advanced Solution Development Framework)

日本ユニシスは，時代の求める前記のような要求に応えるため，ソリューション構築における新たなフレームワーク『ASDF』を提唱する．

ここで言うソリューションとは，ユーザが情報システムに対して抱える課題の包括的な解決を意味するものであり，従来の「アプリケーション」の枠組みを超えるより広汎な「解決策」を指している．

ASDF は新しい統合分散環境の指針である UA の考え方にもとづき，統合 CASE とエンドユーザ・コンピューティングを基軸として，ソリューション実現のためのツール，サービスおよび方法論を統合化した体系（フレームワーク）である．

ASDF は，UA の描くシステム像を具体的に実

現するためのものである．すなわちUAが理想とする新しい統合分散環境，インフォメーション・ネットワークを実現するために，日本ユニシスが推奨する具体的なアプローチである．ASDFは企業における情報技術の戦略的意義の明確化と，企業情報システムの「戦略的構築」のための一貫したツール，サービス，方法論の提供を目的としている．

5.1 ASDFの構成

ASDFは図2に示す三つの層（原則，統合プラットフォーム，ツールとサービス）から構成される．

5.2 原　　則

原則はASDFを提供・発展させていくに当たって，その有効性・継続性を保証するために適用する基本的な方針である．すべてのツール，サービス，方法論はこれらの原則にそったものが提供される（図3）．

1) ビジネス指向……ASDFはソリューションを構築する際，技術面のみからとらえるのではなく，『ビジネスのための』という面からとらえるべきと考えている．そしてASDFは多種多様な企業活動の中でも，とくに戦略的なビジネス活動の支援に狙いをあてている．つまり，技術指向にとらわれるのではなく，経営戦略上必要なソリューションを，経営戦略が求める時期に，タイムリに提供できることを目指している．

2) 柔軟なソリューション……経営環境の変化に迅速に対応する柔軟性を備えるべきと考えている．長期的な展望に基づいて情報技術のインフラストラクチャを構築し，時代の要求にあったソリューションをモジュール化して構築することで，要求に即応できる環境を整えることを目指している．

3) エンドユーザの参加……経営者，管理者等も含め，情報活用の主体者であるビジネスの専門家の直接的または間接的な開発への参加によって，ユーザニーズをソリューションに正確に反映させるべきと考えている．

4) 標準への準拠……国際標準・業界標準に準拠すべきと考えている．これによりユーザの開発したアプリケーション自体の相互運用性が向上すると共に，第三者やユーザが開発したツールも統合できるオープンな環境が提供できる．

5) 既存システムとの共存……既存のデータやプログラムを生かすべきと考える．既存のアプリケーション資産を継承し，ユーザ側の投資を有効活用するため，徒来の技術で開発された既存システムとも共存できるようにする．

6) マルチベンダ環境の支援……標準への準拠や他メーカの機械へのインタフェースを通して，マルチベンダ環境においてもASDFがその効果を発揮できることを目指している．

5.3 統合プラットフォーム

統合プラットフォームはASDF全般にまたがって使われるソフトウェア群である．これらは5.4節で述べる各種のツールを共通に支援し，環境を統合的にまとめあげるためのソフトウェア・プラットフォームであり，三つの要素で構成される（図4）．

1) ユニシス・リポジトリ……開発・運用・保守に関わるすべての情報を一元的に管理するためのデータベースである．このリポジトリ

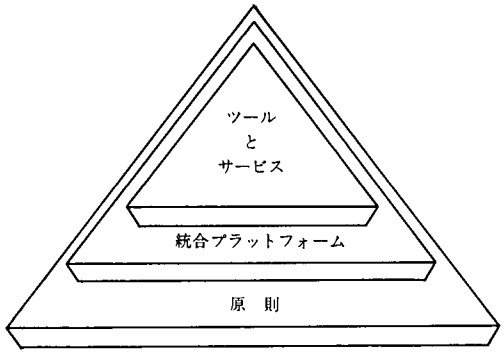


図2　ASDFの構成

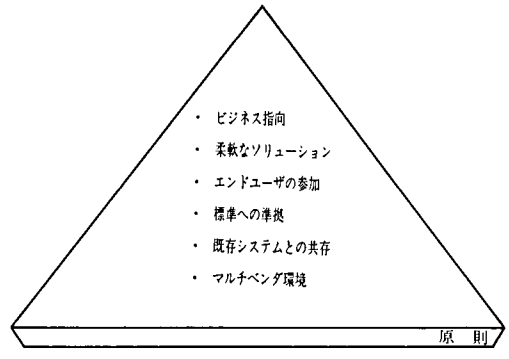


図3　原　　則

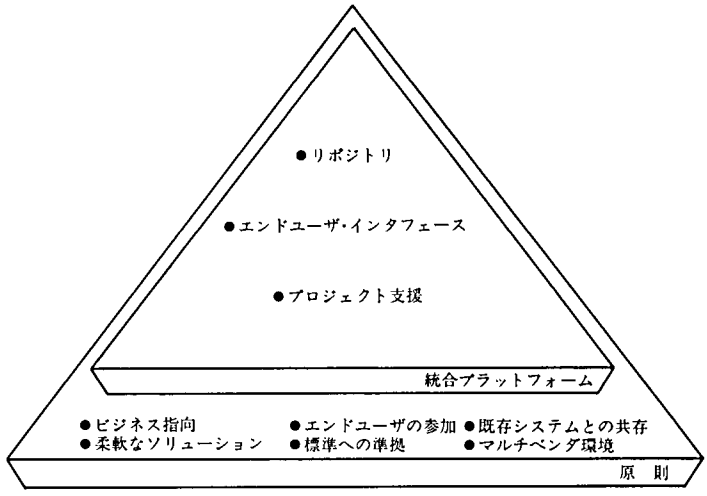


図4　統合プラットフォーム

により，ビジネス分析段階から保守段階まで，一連のソリューション構築における情報を，構築方法の違いによらず一元的に管理でき，そのため一貫性のある統合的なソリューションを確実に構築・保守することが可能となる．

2)　エンドユーザ・インタフェース……ASDFでは，提供するすべてのツールおよびユーザ・ソリューションの双方に，業界標準にもとづいたUA準拠の最新のGUI（グラフィカル・ユーザ・インタフェース）を利用できるようにする．

3)　プロジェクト支援……高品質なソリューション構築を実現するために，リポジトリを効果的に活用し各種ツールを統合化する働きを持つプロジェクト支援機能を提供していく．

5.4　ツールとサービス

原則にもとづき，かつ統合プラットフォームに支えられて，ビジネス指向のソリューションを構築するための，次のようなツール，サービス，および方法論を提供する（図5）．

ASDFでは，「ビジネス分析段階」，「情報ソリューション計画段階」を経て，「ソリューション構築段階」へ至る．ASDFはビジネスの真の課題の解決に最大の焦点を当て，その迅速な実現を提唱するものであり，ソリューション構築の中核に「迅速なソリューション構築」を提案している．「迅速なソリューション構築」は，オンライン・リアルタイム方式およびバッチ方式で中核的な業務処理を行うための「トランザクション処理」向けと，情報を効果的に活用するための「経営者支援とエンドユーザ・コンピューティング」向けに区分される．そしてその「迅速なソリューション構築」を効果的に実現するためにも，既存システムを有効に活用することを提唱している．この面での具体的な提案が「既存システムとのインタフェース」および「リエンジニアリングとモダナイゼーション」である．

5.4.1　ビジネス分析

ASDFにおけるシステム構築の最上流の段階である．ASDFでは，この段階を支援するCASEツール，サービスおよび方法論を提供する．コンサルティング・サービスに関しては，日本ユニシスからだけではなく独立したコンサルティング会社からの提供も計画している．

この段階では，ビジネスの方向性，企業の戦略，情報システムの戦略，情報システム基盤の構想等を明確にする．

5.4.2　情報ソリューション計画

この段階では，ビジネス分析段階で明確化された情報システムの戦略と情報システム基盤の構想をもとに，情報システム基盤の具体的な整備計画およびソリューションの具体的な構築計画を作成する．

情報システム基盤の整備計画は，企業内のインフォメーション・ハブ，サーバ，ワークステーションの役割の明確化，それらのネットワーキング方法と具体的な配備計画，ソリューションの構築・実行のためのソフトウェアの導入計画等から

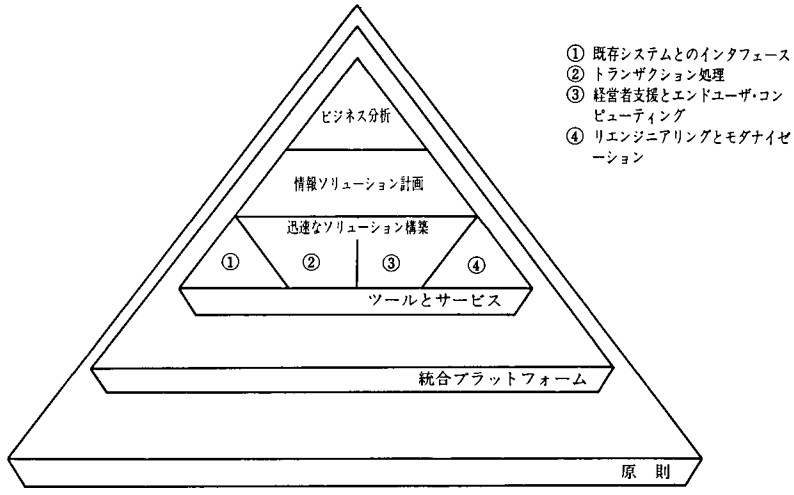


図5　ツールとサービス

なる．

ソリューションの構築計画は，開発するアプリケーションの選定，優先順位，実現時期，実現方法，実現体制等に関する提案である．ASDF ではソリューションの実現方法として，「トランザクション処理向けの迅速なソリューション構築」，「経営者支援とエンドユーザ・コンピューティング向けの迅速なソリューション構築」，「既存システムとのインタフェース」，「リエンジニアリングとモダナイゼーション」を提案している．効果的なソリューションをタイムリに実現するためには，これらの方法を適切に組み合わせることが大変重要である．

ASDF はこの段階を支援するために，CASE ツール，サービス，方法論を提供する．

5.4.3　迅速なソリューション構築（トランザクション処理）

オンライン・リアルタイム方式やバッチ方式によるトランザクション処理システムを，短期に構築するためのツールや方法論を提供する．

目的システムの設計，画面・帳表および処理ロジックの定義の簡素化，完全なソリューション生成等を実現するためのツールや方法論を提供する．代表的なツールには LINC, IDES, ALLY 等がある．

LINC は，上流 CASE ツールや方法論とともに，開発生産性を一層向上させる諸機能を提供することにより，強力な統合 CASE 環境へ発展させる．IDES は，シリーズ 2200 上の 3 GL 実行環境で稼働させるシステムを開発するための下流 CASE ツールであるが，これも上流 CASE の提供により統合 CASE 環境へと発展させる．ALLY は UNIX*環境での分散オンライン・トランザクション処理のための下流 CASE ツールである．

5.4.4　迅速なソリューション構築（経営者支援とエンドユーザ・コンピューティング）

経営者支援システムやエンドユーザ・コンピューティングを支援するために強力なツール群を提供する．ここでは柔軟性の高い情報検索・加工能力と使いやすいインタフェース（マウス，ウィンドウ，アイコン，グラフィックス，イメージ，タッチスクリーン等）の融合により，経営者や管理者の意思決定を支援するためのシステムを容易に構築することができる．また，業務の一線に携わる人々が自ら情報を検索・加工できるエンドユーザ・コンピューティング環境を容易に構築することが可能である．

この分野ではユーザ・インタフェース，ネットワーク機能，リレーショナル・データベースとの連携機能，エンドユーザ向け開発支援機能等が大幅に強化された MAPPER がベースとなる．

5.4.5　既存システムとのインタフェース

蓄積されたユーザ資産を継承するため，3 GL 等で開発された既存システムやデータベースとのインタフェースを提供する．ASDF では戦略的に重

* UNIX オペレーティング・システムは，UNIX Systems Laboratories, Inc.が開発し，ライセンスしている．

要な部分を小さい単位で迅速に開発し，それを段階的に積み上げていくといった進化型の構築アプローチを提唱しているが，この考え方を実現するためにも，既存システムを新たなソリューションの中に含められることが非常に重要となる．

具体的にはLINC, ALLY, MAPPERでは多様な方法でそのインタフェースをとることができる．またIDESでは3GL実行環境と大変親和性の高いシステムを構築できる．

5.4.6 リエンジニアリングとモダナイゼーション

既存システムを最新の技術や環境によって再構築するためのツールを提供する．既存のプログラムから仕様を生成するリバース・エンジニアリング・ツールや，さらにリバース・エンジニアリングされた仕様からASDFの最新技術によってシステムを生成することを目指したリエンジニアリング環境を提供する．

また既存の3GL/4GLアプリケーションを大幅に変更することなく，最新の技術を導入し，アプリケーションを活性化し，時代の要請にあったシステムにモダナイゼーション（近代化）するツールも提供する．具体的な例として，ユーザ・インタフェース部分だけをWindows等の最新のGUIに変更できるツールがあげられる．

6. ASDFの特徴

以上述べてきたように，ASDFはソリューション構築のための体系化されたフレームワークであり，図6に示す構成になっている．

そして，ASDFは以下のような特徴を持っている．

1) オープンとユニシス固有技術の融合……国際標準，業界標準を積極的に支援し，オープンな環境への対応を図っている．
2) 既存システムとの共存……COBOL等，3GLで開発されたシステムとの共存を目指し，既存アプリケーションの継承を図っている．
3) 経営戦略を即反映できる情報システムを迅速に構築……役員室の意思決定を直接プログラム（コード）に反映した戦略的な情報システムを，短期間で実現することができる．
4) 一貫したツール，サービス，方法論……上流工程における企業戦略作りや情報アーキテクチャ作りから保守の段階に至るまで，一貫したシナリオにもとづいたツール，サービス，方法論を提供している．
5) 実績のあるベース……すでに多くの所で実績のあるツール，サービス，方法論がベースになっているため，安全な導入と即時の効果が期待できる．
6) 導入段階の柔軟さ……ASDFの導入では，既存システムの現状やユーザニーズに応じて，ビジネス分析・情報ソリューション計画・ソリューション構築のどの段階からでもスタート可能である．そして徐々に範囲を拡大することによって，漸進的に一貫した体系に移

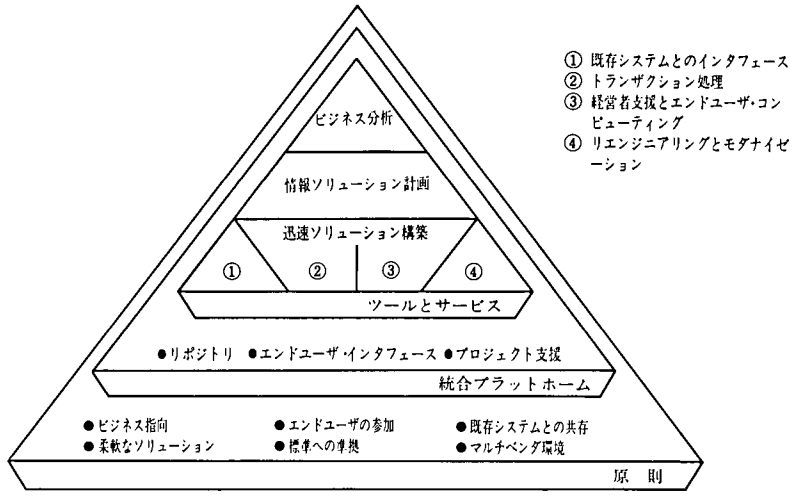


図6 ASDF

行することができる．

7) 最新技術の導入……ASDF の導入により，UA の考え方にそった最新の情報システム基盤を自動的に取り込むことが可能になるため，ユーザはビジネス上の課題に集中でき，結果として競合優位な高品質な情報システムを実現できる．

7. おわりに

ASDF は，単なる「アプリケーション開発のためのツール群」ではなく，ユーザが情報システムに対して抱える課題の包括的な解決をめざした「ソリューション構築のための統合的な環境」を提供するものである．

ASDF の採用により，次のような効果が期待できる．

1) ビジネスに集中……ASDF では，国際標準・先進技術が自動的に取り込まれる．ユーザは変化の激しい最新技術の動向に注意を払う必要はなく，ビジネス上の課題の解決に集中でき，変化の激しいビジネス環境に柔軟に適応できる情報システムを構築できる．

2) エンドユーザの積極的参加……エンドユーザの積極的な参加により，ユーザニーズを的確に反映する利用者主体のシステム構築が行える．

3) 整合性のある長期的なシステム構築……計画段階から実施段階まで一貫した考え方と方法論を採用することにより，長期に耐えられる柔軟な情報処理環境を実行できる．

4) 資産の継承……ソフトウェア，アプリケーション，ハードウェアにおける既存資産を継承し，現行の投資を保護することができる．また，新技術導入に対するリスクを低減し，新技術導入に伴う移行費用を最小限に抑えることにより，最大の効果を実現できるソリューションに集中投資が可能となる．

現在システム構築の世界は，かつてない大きな変革の時期にきていると言えよう．ASDF は 21 世紀に向かう時代における，システム構築のための日本ユニシスからの提案である．日本ユニシスは今後この ASDF にそって，ユーザのトータルなソリューション実現の支援をしていく．

掲載論文梗概

情報系システムは，多様で変化する情報要求に迅速に対応し，価値ある情報を提供できるものでなければならない．RDIP は，情報系システムに求められる諸要件に具体的に対応した汎用的な情報検索システムの短期間での構築，および豊富な利用手段の提供によるシステム活用の支援を目的とするミドルソフトウェアとして開発された．山崎裕明は**汎用情報検索システム RDIP** の中で，情報系データベースの構築から情報の活用に至る過程において，RDIP が支援する機能を順を追って紹介している．

コンピュータ化の対象範囲が複雑な情報構造を必要とする業務へと拡大し，データ構造が複雑になるにしたがって，問い合わせ式で要求される検索も複雑になり，そのため複雑な構造を持った情報を簡単な問い合わせ式で効率よくアクセスすることがデータベース管理システムに対して要求されるようになった．山口裕久の**オブジェクト指向データベースにおける問い合わせ式構造的最適化技術**は，ユニシスのデータベース管理システムの一つである SIM を例に，複雑なデータ構造に対して効率よくアクセスし結果を得るための問い合わせ式最適化技術を紹介し，オブジェクト指向データベースで要求されるであろう最適化技術を考察している．

情報系システムは，利用部門，利用目的，利用方法等の拡大に伴い，容易なシステム構築・再構築を可能とする情報基盤が必要になっている．また，情報系システムの要件として，変化する多様な情報要求に対する迅速な対応，情報の価値への考慮がある．森良行・小野寺裕の**営業支援システムへの意味型データベース SIM の適用**は，A 社の営業支援システムが情報系システムの一環として稼働し，基礎データベースに SIM を採用して，要件に応じたシステムを実現でき，生産性の高いシステム構築・再構築を可能とする仕組みが提供できたことを紹介している．

掲載論文梗概

ISSN 0914-9996

技　報
UNISYS TECHNOLOGY REVIEW
Vol. 12　No. 1　(No. 33)

発　行　日　平成 4 年 5 月 31 日
編　集　人　柳　生　孝　昭
発　行　人　富　田　和　夫
発　行　所　日本ユニシス株式会社
東京都港区赤坂 2-17-51 〒 107
TEL (03) 3585-4111 (大代表)
印　刷　所　三美印刷株式会社