55－S 002

マン・マシン・ユーザ・インタフェイス
に関する調査研究報告書

昭和56年3月

JIPDEC

財団法人　日本情報処理開発協会

この報告書は、日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて昭和５５年度に実施した「マン・マシン・ユーザ・インタフェイスの調査研究」の成果をとりまとめたものであります。

序

当財団は情報処理技術の調査研究の一環として，昭和５５年度より「マン・マシン・ユーザ・インタフェイスに関する調査研究」に着手いたしました。

今後の情報処理に要求される基本的条件の１つは，より広汎な分野およびアプリケーションに対応し得るマン・マシン・ユーザ・インタフェイスの改善であります。即ち従来のコンピュータ・システムの使用者は比較的専門家の比率が高く，コンピュータが機械であるための使用上の不自由さや不便さはある程度見過すことも可能でありました。しかしながら，今後期待される多様な且つ広汎な利用分野においては恐らく数多くの非専門家が直接システムの操作を行う可能性が極めて高いものと考えられます。

情報処理システムが多くの非専門家にとって使い易いツールとなるには，コンピュータ側のインテリジェンスを向上させ，人間の五感による情報伝達機能により近い形態で各種の情報の処理が容易に行なえることが必要不可欠となります。

そこで，本プロジェクトにおきましては，1985年頃を目途とした今後の新しい情報処理システムにおいて，実現されねはならぬ，マン・マシン・ユーザ・インタフェイスの革新的な向上を支える基本的諸技術についての調査研究を行うことと致しました。

本調査研究は昭和55年度より３ケ年計画で実施する予定であり，本報告書はその第一年度目の成果をまとめたものであります。

本報告書がこの方面に興味をお持ちの方々に広く利用され，今後の情報処理技術向上の一助として寄与することができれば幸いです。

なお，ここに本プロジェクトの研究会に於いて，ご熱心な討議を通して多分のご指導，ご協力を戴いた東京大学大須賀助教授をはじめ研究会メンバー各位，ならびに分散型データベース関連の研究開発成果に対し，貴重な評価および御意見をいただいた北海道大学田中譲講師に心から感謝の意を表します。

昭和５６年３月

財団法人　日本情報処理開発協会

会長　上野幸七

目　　次

はじめに

マン・マシン・ユーザ・インタフェイスは多様な側面からとらえることができる。例えば，従来はＴＳＳにおける会話型言語の機能や，コマンドの操作性，あるいは端末のキーボード配列やコントロール・キーの位置等，利用者が直接マシンあるいはシステムに触れる部分のみをクローズアップしたマン・マシン・ユーザ・インタフェイス論が中心であった。

本プロジェクトでは，コンピュータ・システムが今後更にすぐれた人間のアシスタントとなるべきであるという観点から1985年以降を目途とした，より将来的，且つ理想的なマン・マシン・ユーザ・インタフェイスに改革的な変化を与え得るために実現されねばならぬ基本的な情報処理技術要素自体に対する検討を行うことを主目標とした。

本調査研究は昭和55年度より３ケ年計画で実施する予定となっており，本年度はその第一年度目にあたる。

本年度の調査研究は，内容を大きく２つに分けている。その１つは，マン・マシン・ユーザ・インタフェイスを支える今後の新しい情報処理技術として，音声，画像，図形，文字，記号等各種情報メディアの統合処理を可能とするファイル，入力装置等の技術動向を中心に，従来のコンピュータ・システムが不得意としていたパターン情報処理や非数値情報が容易に処理可能になることにより始めてマン・マシン・ユーザ・インタフェイスの革新的な向上が図り得るという観点で調査検討を行ったものである。

もう１つは今後のオフィス情報システムをはじめ各種のアプリケーションにおけるベーシック・リソースとしての分散型データベースにつき，マン・マシン・ユーザ・インタフェイスの観点から見た望ましい実現形態およびそれを支える技術的要素の検討を行ったものである。本報告書ではこれらを第Ⅰ部，第Ⅱ部とに分けて構成した。

第Ⅰ部のマン・マシン・ユーザ・インタフェイスの調査では1980年代後半の実用化を目途として高度なマン・マシン・ユーザ・インタフェイスを支える基本的な情報処理技術として統合メディア，新しい入力装置等の動向ならびに人間工学的見地からのマン・マシン・ユーザ・インタフェイスに具備されるべき機能要件等について調査した。なお，本調査研究は東京大学大須賀節雄助教授を主査とするマン・マシン・ユーザ・インタフェイス研究会を設けて行った。

本報告書の第Ⅰ部第１章ではマン・マシン・ユーザ・インタフェイス概論として一般論を述べている。第２章では各種情報メディアの統合について情報メディアの概念，統合に対する要求，情報の蓄積・伝送，既存システム例，今後の動向等について述べている。第３章では新しい入力装置について入力装置の記述方法，現状，現状の評価と将来への展望等について述べている。第４章ではマン・マシン・ユーザ・インタフェイスの人間工学的考察としてマン・マシン・ユーザ・インタフェイスの現状と人間工学的考察，今後のマン・マシン・ユーザ・インタフェイスのあり方等についてユーザの立場からの要求機能に対するアンケート結果を含め述べている。

第Ⅱ部の情報システムにおける分散型データベースシステム技術の調査研究ではオフィス情報シ

ステムを始めとする今後の各種アプリケーションに共通する重要な技術としての分散型データベースシステムについてそのデータベースモデル，問合せ処理，システムアーキテクチャ等につき調査研究を行うとともに一部のインプリメンテーションも行った。

本報告書の第Ⅱ部第１章では序論を述べ，第２章ではデータベースシステムモデルについて既存のデータベースモデルの再整理を行った結果について述べている。第３章では新たなアプリケーションとして重要なオフィス情報処理システムとデータベースとの関連について検討し，このモデルとしてのフォームフローモデルを提案している。第４章から第７章にかけては，第Ⅱ期分散型データベースシステムＪＤＤＢＳ－Ⅱのモデル化とその開発内容について述べている。

即ち，第４章では，ＪＤＤＢＳ－Ⅱの全体アーキテクチャを論じ，第５章では問合せの処理概要を述べ，第６章ではローカル共有媒体ネットワークを用いた時の通信処理モデルと処理アルゴリズムの提案と評価を行い，第７章ではＣＯＤＡＳＹＬ　ＤＢＴＧデータベースシステムのリレーショナル・インタフェイスについての検討とそのインプリメンテーションと評価結果について述べている。最後に第８章では今後の課題を整理し第Ⅱ部のまとめとしている。また，付記としてデータベースと抽象化，同種化システム，ＱＴＰの例を掲げた。

なお，第Ⅱ部の調査研究にあたっては第３者の評価として北海道大学工学部の田中譲講師のご協力を得た。

第Ⅰ部　マン・マシン・ユーザ・インタフェイスの調査

マン・マシン・ユーザ・インタフェイス研究会委員名簿

（敬称略，順不同）

大須賀	節雄	東京大学宇宙航空研究所助教授
井関	義弘	三井造船㈱システム本部プロジェクト室課長
松岡	潤	㈱日立製作所システム開発研究所企画室主任研究員
押尾	勝平	富士通㈱開発事業部方式部第一方式課調査役
花木	真一	日本電気㈱中央研究所周辺機器研究部研究スペシャリスト
山本	欣子	(財)日本情報処理開発協会開発部長

第Ⅰ部　マン・マシン・ユーザ・インタフェイスの調査

1. マン・マシン・ユーザ・インタフェイス概論

1.1 情報処理系のインタフェイス

マン・マシン・ユーザ・インタフェイスは情報処理系における一般的なインタフェイスの一つである。情報処理系におけるインタフェイスとは，２つ（もしくはそれ以上の）独立の情報処理装置間に介在し，この両装置のそれぞれの独立性と同時に併存・協力性を保証するための機能もしくは（広義の）装置とする。また独立の情報処理装置とは，少くとも情報の記憶機能，制御機能，処理（変換）機能を備え，単独で一連の情報処理機能を果たすことのできる装置とする。これらの諸機能を遂行するための手段や物理的な装置の種類，形態は多様である。各独立の情報処理装置（以下，単に情報処理装置あるいは装置という）は，それぞれが固有の情報表現形式を有し，それに基づいて処理（変換）方式が定められている。すなわち，各装置ごとに表現できる情報形式のクラスと処理可能範囲は機構上定まっており，この範囲内のものについてのみ各装置は正常な動作が保証されている。

各装置が扱かい得る情報のクラスはそれぞれの装置の物理的な構造に依存する。このクラスは各装置に固有のものであるから，他の装置のものとは一般には一致しない。このような装置は単独で動作している間は問題ないが，異なった装置を協力させて同じ問題の処理を分担させようとする場合，この両者を直接結合することはできない。両者で情報の表現の形式，処理の方式が異なるため，このままでは処理不能に陥ったり，全く意図しない処理を行うことになってしまう。したがって，一方から他方へ，あるいはその逆に，情報が送られる際に，送る側の形式から送られる側の形式への変換機能が必要になる。これを行なうのが独立した情報処理装置間のインタフェイスである。

独立した情報処理装置間のインタフェイスは独立した情報処理装置の範囲をどのようにとるかにより異なり階層的な性質のものである。たとえば，計算機システム全体を一つの単位と考え，これと他の計算機システムとの間のインタフェイスを考えることができる。一方，各計算機システム内では，ＣＰＵと独立の入出力制御機構を備えた入出力装置をそれぞれ独立の情報処理装置の単位として，この間を結ぶインタフェイスも必要となる。人間も一つの独立した情報処理装置と考えることができるが，この時端末装置はその全体が人間と計算機システム間のインタフェイスとして考えられると同時に，その下のレベルでは端末装置を一つの独立した情報処理装置としてこれとＣＰＵもしくはメモリとの間のインタフェイスが必要になる。

上述のように人間自身も一つの独立した情報処理装置と考えられるから，人間と計算機あるいは他の独立した情報処理装置の間にインタフェイスが必要であり，これがマン・マシン・ユーザ・インタフェイスである。マン・マシン・ユーザ・インタフェイスは他のものに比し一般には複雑である。これは人間の情報処理機能の多様さ，複雑さに起因する。特に情報のセマンティクスの処理がインタフェイスに入り込んでくるからである。

I.2　情報のシンタックスとセマンティクス

人間を除く他の情報処理装置では，通常は情報の表現形式と処理機構すなわち情報表現の変換のメカニズムが厳密に定められている。したがって，入力形式が与えられた後，それを別の形式に変換したい時，それらがすべてこの装置に許されたクラス内のものなら，このメカニズムを用いて自動的に変換が実行される。この変換は機械的であり，個々の変換に含まれる意味は従来は強くは意識されてはいない。しかし各情報処理装置が受け持っているのは情報の発生から処理の終了に到る全過程の一部にすぎない。

情報処理の全過程は現実の世界に発生する問題をまず一定の形式化規則のもとで，形式化された情報によって記述し，しかるのち一定の変換規則によって情報を形式表現の枠内で変換し，その結果を再び現実の世界に戻して終了する。すなわち情報処理のプロセスの全体は実際には意味の処理を行なうのが目的であり，それを最初および最後の現実の世界と情報の形式的表現間の変換により，形式の変換におきかえている。これはいわば意味の世界と形式の世界との間の変換であり，セマンティクス処理あるいはセマンティクスーシンタックス変換と呼ぶことにする。

最初の変換すなわち現実世界から情報の世界への変換は特に重要であり，このプロセスが入ることによって以後の処理においてセマンティクス（現実の世界との対応）をその都度考慮せずに済む。この部分は様々な呼び方がされているが，ここでは単純にコード化と呼ぶことにする。現状では大半の場合コード化は人間によって行なわれている。コード化に際しては，情報の一つの形式化表現が現実の世界におけるある意味を表わすものとするという一定の約束が前もって存在し，それに基づいて変換が行なわれる。

このプロセスをもう少し詳しく知るために，現実の世界における現象あるいは問題が，まず人間によってどのように理解されるかを見てみよう。人間がある現象を認知するためには，まずその現象にかかわる対象の認知が基本となる。この対象は，人，犬，家などのように物理的な存在であれ，戦争，愛，神のような抽象的な概念であれ，その存在と範囲が認知され，それについて記述を行たい得るものに限られる。以下，これを実体と呼ぶことにする。すると現実世界における出来事，現象等は，この実体に生ずる変化あるいはいくつかの実体間の関係に分解することができる。これを簡単のため関係という言葉で総称することにする。関係とは述語で記述されることがらといった意味であり，したがって，"眠る"，"愛する"，"行く"，"美しい"，"親子である"，などはすべて関係であり，これら関係がどの実体について存在するかによって出来事や現象が認知されたとしてよいだろう。

２つの独立した情報処理装置が共に人間である場合，両者間の情報の伝達には各種の手段が用いられる。言葉，図形，画像，身振り，目くばせ，接触，その他である。これらは人間に備わった感覚機能を用い，したがって人間に最も適した自然な方法である。いずれの手段によるにせよ，人間が認知したことを他の人間に伝えるためにこれらの方法による表現に変換することもコード化の一形態と言える。言葉を例にとると，認知対象である各種実体とその間の各種関係に対応して語があり，またそれらの配置により，関係をより詳細に表現するための文法という構造規則がある。これらの

規則により，一つの形式とそれが表わす意味との対応が前もってつけられている。人間における発話，すなわち人間間におけるコード化はこの規則にのっとった意味から形式への変換といえる。この事情は言葉以外の手段による場合でも全く同様であり，この意味で以下，言葉という時にこれらすべての伝達手段を含む広義のものを表わすことにする。

このように，人間の場合，言葉への変換は意味ー形式の変換ではあるが，言葉の構造自体はかなり複雑なものである。これで十分なのは聞き手の側で言葉から意味への変換能力があるからで，計算機ではこのようにはならない。理解能力が不十分だからである。

情報処理装置が人間以外のものの場合の多くは，この意味ー形式間の変換を自からは行なわず，多くの場合，すでにコード化された情報が与えられることを前提として設計される。すなわち，情報のセマンティクスは扱かわれず，シンタックスのみによる処理（変換）が行なわれる。セマンティクスーシンタックス間の変換は意味の理解という機能をもつもののみが可能だからであり，従来の情報処理装置ではこのような機能を実現できなかったからである。

近年パターン認識装置への関心が高まり，文字認識，物体認識などの試みがなされ，特に文字認識は一部実用化レベルに達している。パターン認識は，従来，専一的に人間に任せられていたセマンティクスーシンタックス間の変換の一部を機械化しようとするものである。現実の世界の出来事や現象を，人間の眼を通さずに直接コード化しようとする点で，これはセマンティクスの領域に必然的に踏み込むことになるからである。

現実世界の出来事を認知するためには，人間もそれを表わす原始情報を取り込む機能が必要である。視覚・聴覚等五感はこのような検知機構であり，これを通して得られるものは情報である。この原始情報を分類し，パターン化し，その基本パターンにつけられた名前を用いて全パターンの構造を記述することがコード化の機能であるが，パターン化はそれ自体意味と形状の対応づけであり，このプロセスには過去の経験や法則等の知識を用いる点が通常の情報処理の考え方と基本的に異なる。現在のパターン認識技術はまだこれらを実現するだけのレベルに達していない。現在実現可能なのはパターンのコード化のためのアルゴリズムを前もって作成しておくことができ，意味的な知識を参照しないで処理し得る程度のもの，すなわちパターンが単純で分類も少数で済むものとか，文字のように，完全な原始情報ではなく，人間同志のレベルでは十分にコード化された情報として扱かわれているものなどに限られる。したがって，これらは厳密な意味でセマンティクスを扱かっているのではなく，２次元形状情報を１次元の情報に変換する技術である。

本題に戻り，マン・マシン・ユーザ・インタフェイスが他の装置間のインタフェイスと根本的に異なるのは，このように人間と機械との情報処理能力におけるあまりにも大きな差があるためである。このギャップを埋めるにはまだ相当の時間を必要とするであろう。そして，今後もなお当分の間，人間は計算機を利用するために，計算機が機能し得るような情報を準備すること，すなわち，人間のもつ表現形式から情報処理装置の固有の形式へのコード化を要請されるであろう。これに関しては，より現実的な問題として，情報処理技術の開発・改良により，出来るだけ人間の負担を軽減することが重要であり，また情報処理装置をより使いやすくするための使い方の技術の開発など

が必要となろう。

1.3 インタフェイスのレベルと標準化

独立した情報処理装置が同一言語を理解する人間同志である場合，コミュニケーションのためのメディア（言語，図形，その他）は共通であり，意味の言語表現としてのコード化（語り手）と，言語から意味の変換としての理解（聞き手）が必要な変換のすべてである。両方が同型の機械的情報処理装置である場合も状況は類似している。情報の表現形式の基本（たとえば語長）は同一であり，それを用いた構造表現，たとえばデータ構造による問題の表現レベルで変換が行なわれれば十分である。

これに対し，両装置が同型のものでない場合，さらに多くの変換が必要になる。これを以下，論理的な変換，形態的変換，物理的変換に分ける。

(1) 論理的変換

問題の意味ー形式間の変換を論理的変換と呼ぶことにする。これまで述べてきたセマンティクスーシンタックス変換のことである。なお1.2節ではセマンティクスを認知のレベルで論じたが，論理的変換はマン・マシン・ユーザ・インタフェイスにおいてのみ特徴的な変換であり，意味ー言葉の変換は人間の自然の活動であることから，以下ではこれを言語・図形・画像等，人間が自然に用いているメディアにより表現された意味と，その計算機内表現への変換とする。

(2) 形態的変換

各種情報処理装置にはそれぞれ固有のデータ表現形式や制御情報形式が与えられている。表現の単位である語長，語の集まりであるセル内の語（バイト）数，セル単位の情報の記憶・転送などの際の制御手順などである。これらの形式の異なる装置を結合する際には当然形態的変換が必要である。

(3) 物理的変換

情報は物理的信号で表わされるが，この実現方式も装置に固有である。信号のレベル，クロック周期やその形式，信号制御手順などが異なる装置の結合にはこれらの間の変換が必要である。

インタフェイスを論ずる時，これらレベルの異なる変換を区別して考える必要がある。なお，(3)については主として電子回路としてハードウェア設計の問題であり，このレベルのインタフェイスが常に問題とされるような状態では，今後，多様化が予想される情報処理のニーズに対応する上で障害となる。これはかなりの程度，標準化により解決されてゆく問題である。一般にインタフェイスに伴なう問題は，標準化が可能である時はそれによって軽減される。電子回路設計面での標準化は可能であり，現にそれが実現しつつあるが，今後も一層この方向の努力は必要である。

一方，(2)については，各種装置の多くが，要求されている機能を固有の物理的構造の特徴を利用することによって実現しているという性質上，必らずしも標準化が可能ではない。たとえば，ディスクやドラムは磁気の性質を利用して情報を記憶し，磁気の時間的変化により信号を取り出すために回転機構を用いるが，このような複雑な構造のものでは，これを実現する方式の選択や技術レベル

に幅があり，標準化には限度がある。

(1)，(2)の変換は主としてソフトウェアに関するものである（将来，ＬＳＩ化するにしても，(3)の意味でのハードウェアとは異なる）。したがって，将来(3)のハードウェア・インタフェイスの問題が標準化によって解消した場合，インタフェイス問題は主としてインタフェイス・ソフトウェアに重点がおかれることになる。この時(2)の問題は，装置固有の構造から来る制約や，製造者側のコスト評価などの点で標準化しにくい装置同志を結合するためのインタフェイスをソフトウェアによりいかに解決するか，それをユーザ・レベルで処理し得るようにするにはいかなる方式をとるかに向けられるであろう。以下ではこの意味でマン・マシン・ユーザ・インタフェイスに関し，主に(1)，(2)のレベルの問題について論じる。

1.4　マン・マシン・システム

マン・マシン・ユーザ・インタフェイスはマン・マシン・システムを形成する上で必要な機能もしくは装置であるから，それのあり方を考察するに際してはマン・マシン・システムというもののあるべき姿を十分考察して，そこからマン・マシン・ユーザ・インタフェイスとして要求される機能を導びき出すことが必要である。

マン・マシン・システムは元来人間と計算機とが協力して問題を解くことを目的とするもので，これ以外の方法では問題解決プロセスに計算機を導入しにくい設計，医療診断，意思決定，教育，研究などの分野に効果的であることが期待されているものである。ここで問題は常に人間の側から発するから，機械の役割はこの問題解決に際して人間を援助することである。人間は機械を利用するために，この問題の解決プロセスのどこかでこれを計算機に与えねばならない。問題のどの範囲を計算機に受け持たせるかは計算機利用のコスト，問題を計算機に入力するためのコスト（労力）等，計算機を利用するために必要となるコストと，計算機を利用することにより得られる利得（時間・労力等を含む）の兼ね合いで定まるものである。初期の計算機はコストも高く，計算機利用技術も未熟であったため，問題解決の全プロセスのうち，ごく一部を受けもつに過ぎなかった。たとえば航空機を設計するという問題において，構造計算の一部を計算機に行なわせるという程度のものである。これは現在の航空機のデータ収集と分析，基本設計方針の確立，初期概要案設計，原動機の選定，概案設計図面の作成とその検討，風洞モデルの作成と実験，より詳細な形状設計，線図等を含む図面の作成，翼型の選定，空気力学的特性の解析，構造設計と構造図面の作成，構造解析のためのモデル化，構造解析，構造強度，剛性試験，製造組立図面作成…等と続く複雑な作業のごく一部であり，しかしこのような人間中心の作業の一部を計算機化するためにモデル化と計算機用の情報表現（プログラムおよびデータ）の作成のための手間が余計にかかる。しかし敢えてそれを行なうのは，これらの手間をかけ，計算機使用料を費やしてもなお，その効力が人間が多数のパラメータを変えつつ構造計算を進めるより，安く（速く）結果が得られるからである。またこれ以外の作業まで計算機を用いることは，構造計算という一定の前もって定まった手順の繰り返しによる大量の機械的演算に比し，作業内容が多様であり，時々刻々と得られる各設計段階での結果やデータ

にもとずき，多くの基準を適用しつつそれ以後の作業の方向が改定されるといった動的な要求が多く，前もって作成したプログラムを繰り返し利用することによって結果が得られない種類のものであるが故に，これらを計算機化するのは技術的にも経済的にも決して妥当な方向ではなかった。

しかし，このような事情は計算機のコストの急激な低下，対話技術を中心とするマン・マシン・システムの技術の発展により，当然変更し，全問題解決プロセスにおける機械化の範囲の比率も大になってきているし，今後は人件費の一層の増大，計算機の費用の一層の低下から「この傾向は益々促進される」ことは疑いない。

I.5　論理的変換と知識ベース

ここで問題となるのがマン・マシン・ユーザ・インタフェイスの問題である。人件費の増大が，計算機利用範囲の拡大の原因の一つであるとするなら，計算機を利用するために人が行なわねばならない作業も当然短縮すべきことが要求される。従来は人間が問題のすべてを計算機流の表現に変換していたが，これを極力少なくすることが要求されてきている。これは二つの形で現われる。一つは人間が用いている表現をそのまま計算機に与えること，すなわち，セマンティクス－シンタックス変換を計算機自身に行なわせること，もう一つは必要な記述量を出来る限り減らすことである。いずれも，通常，人間同志が行なっている対話形式に近い形で計算機とも情報交換ができるようにしようとするものである。

注意せねばならぬのは，人間の言葉から計算機流の表現への変換を従来のごとく手続きとして行なうのでは決して人の負担を軽減しない点である。人間同志でも会話を成立させるために必要な情報量は，それを伝達するために人間が用いることのできる手段－目・耳・口・手などの器官－によって単位時間当たりに変換し得る量に比し十分大である。もし必要な情報量をすべてこれらを通して伝達せねばならないとしたら，伝達に時間がかかり，会話の興味は失なわれるのみならず，疲労のために会話が成立しなくなることもあるであろう。現実には，会話内容に関して，両当事者が一定の事前知識を有しており，その部分に関してはすべて伝達しなくても単に相手の知識を励起する情報のみを伝達すれば良いという条件のもとで，必要伝達量が減少し，会話が成り立っているといえる。たとえばある目的に関し，綿密な実行計画を作成し，それを知識として有している当事者同志の間では，いざ実行の合図は目くばせ一つで十分である。目くばせという１ビットの情報が，知識を励起し，計画内容に相当する情報量が伝達されたと全く同じ結果をもたらすからである。

この状況はマン・マシン・システムの場合でも同様である。上述の第２の傾向－必要な記述量を減らすこと－がこれに相当するが，このためには計算機側で問題に関連する多くの知識を保有し，適宜それを用いて人間から送られる情報を補いつつ，問題の記述を計算機自身が完成させるという機能が必要とされる。このようなシステムを知識ベース・システムと呼ぶ。近年，知識ベース・システムに対する関心は急速に増加しているが，マン・マシン・インタラクションという点でこの傾向は極めて重要なものといえる。

1.6　人間工学的配慮

しかし，このような論理的変換機能を計算機が完全な形で所有するようになるのは，可能としても実現はかなり先のことになるであろう。それまでは部分的に機械化可能なものから手をつけ，より良いマン・マシン・ユーザ・インタフェイスを実現してゆく他ない。

入力に関連して，これと同時に形態的変換を，人間の負担をできるだけ少くするように実行することも重要である。人間が計算機と情報交換する際，何らかの動作が必要である。一方，これを受ける装置ー人間と直接に接するこれら装置ーは単に機械的性能のみでなく，人間にとって，生理的にも心理的にもより使い易い，疲労の少ないものであることが必要とされる。このような人間工学的研究はマン・マシン・ユーザ・インタフェイスにおいても大きな研究課題となりつつある。

1.7　情報メディアの多様化

マン・マシン・システムにおいて計算機の受け持つ割合が増大するにつれ，もう一つの傾向が生ずる。従来は計算機の行なう処理はごく単純なものに限られていたものが，計算機の利用比率が増加するにつれ，複雑化し，それに必要な情報も多様化することである。それに伴って，従来はたかだか一種類か二種類程度の装置を用いて情報の入出力が行なわれていたのにたいし，もっと多様な装置を同時に用いて，多様な形式の情報を扱かうという必要も生じてくる。言語のみでなく，音声，画像その他多種の情報メディアを同時に用い，それらを統合して一つの目的に用いるといった複雑な処理も現実に可能になってきているし，要求も強くなってきている。このような多様な情報を計算機自身が扱かうために統合化の技術と共に，新しい入力装置への要求も生じる。従来，人手により計算機流の表現化が行なわれていた時には，情報が多様であっても，人間が理解できる表現のものあるいはそのための装置があれば以後は人手により入力することが（時間と労力を要するにしても）可能であった。しかし，それを自動化するとなると，そのための新しい装置が必要になる。このように処理範囲の拡大は必然的にメディアの多様化，ひいては装置の多様化，すなわち新装置，特に入力装置の開発要求に発展してゆく。

1.8　仮想（論理）装置

情報メディアが多様化し，同一の処理目的にこれら多様のメディアに対応する装置が使われる場合のインタフェイスに従来と異った機能を要求するようになる。主たる情報処理装置としては，その処理目的に適合した単一の装置が存在し，これが処理目的に必要な情報をすべて操作するものであれば制御が単純になり最も望ましいことであるが，現実にはそのようなことはあり得ない。それは計算機の処理目的は問題ごとに異なり，したがってそれぞれ要求する情報も異なるからである。一方，入出力装置側も，前述のように，固有の機能がその構造と結びついていることが多く，多様な情報を機構的に統合した装置を作ることは少くとも経済的とはいえない。結局，それぞれ固有の基本機能をもつ装置を群として用いる他ない。このためのインタフェイスは前述のようにソフトウェア面の問題として扱かわれるべきであるが，多様な装置を必要とする多様メディアの統合要求を満

たすためには，処理要求に適合した，これら装置の機能的組合せを作ることが必要になる。

この際，基本機能を標準化しておくことが出来ればその統合も簡単になるが，現実には装置のレベルの標準化は困難なことが多い。その結果，同一目的の機能でも装置としては全く異ったものが作られたり，同一種類の装置でも機能的に異ったものも作られる可能性がある。たとえばファンクション・キーという最も単純な機能を目的とした装置を考えてみよう。

ファンクション・キーに関しては以下のような機能の一つもしくはその組合せが可能である。

a. ファンクション・キーの位置をソフトウェアにより問合せ得る機能

b. キーを押した時，割込みを生ずる機能

c. キーを開放した時，割込みを生ずる機能

これらの基本機能の他に

d. ソフトウェア制御のもとで，キーの下のランプを点灯する機能

e. 各割込みと同時にキー番号を付して送る機能

なども可能である。

現実のファンクション・キーはこれらの任意の組合せ機能を有するものとして各種のものの設計が可能である。インタフェイスのレベルで望ましいのは，このような現実の装置を用いて，あたかも上述のすべての機能を備えた理想的なあるいは仮想のファンクション・キーが存在するかの如くに用い得るようにすることである。

例として，非常に単純なファンクション・キーがあり，割込み機能は与えられていないとする。一方，これと別にクロック割込みがあるとする。この時，この両者を組合せて，あたかも割込み機能付きのファンクション・キーを使っていると同じにすることが可能である。すなわち，一定時間ごとに時間割込みを掛け，そこでソフトウェアによりファンクション・キーのチェックをする。もし，キー・オンだったらファンクション・キー割込みルーチンに跳ぶようにすればよい。

これは二つの装置（単純なファンクション・キーとクロック）を統合して，問題解決のための処理上，必要な機能を備えた仮想的な装置があるかの如くにする最も単純な例である。

これを一般化し，複数の実在の装置を組合わせて，あたかもそれらの複合機能を持った一つの装置の如くに定義することができる。これを仮想装置あるいは論理入出力装置と呼ぶ。

もし，このような仮想装置の定義がユーザ・レベルで可能であれば，そしてハードウェア・レベルの物理的変換部分が標準化により容易なものとなっていれば，物理的構造や製作者の事情から様々な形態をとる実在の装置を用いながら，しかもそれらを組合せて，複雑化した情報処理が要求する複合機能を備えた装置を仮想装置として定義し，利用することができるであろう。このためには仮想装置定義言語とでも呼ばれる新しい言語があれば便利である。これは今後の課題である。

本報告書は以下で，近い将来このような各種機能を複合化する汎用手段が開発されることを期待し，より現実的な立場から情報メディアの統合，入力装置，人間工学的側面について考察する。

2. 各種情報メディアの統合

2.1 情報メディア

2.1.1 情報メディアの概念

マン・マシン・ユーザ・インタフェイスという観点から，情報メディアの概念は次の様に述べることが出来よう。

① 装置メディア…… 人間とコンピュータ間の情報伝達を行う際に利用される装置群。各種の装置が存在するが，人間の情報伝達時に使用される感覚に対応して分類が可能である。

② 感覚メディア…… 人間の持っている感覚に情報を伝達する際の情報表現形態。たとえば，聴覚への情報表現形態は音声である。

③ 形式メディア…… 感覚メディアの装置上または伝送路上でのデータ形式。ディジタル・データ表現とアナログ・データ表現の二形式があり，ディジタル・データ表現の場合はさらにコード化データ，バイナリ・データとベクトル・データから成る。

情報メディアの統合を考える場合は，装置メディアの様な物理的メディアよりも，感覚メディア，形式メディアを取扱う方が都合が良い。

各メディアと人間の五感（視覚，聴覚，触覚，嗅覚，味覚）との対応を表２－１に示す。

表２－１ 人間の五感とメディアとの対応

五感	装置メディア	感覚メディア	形式メディア
視覚	出力系 ○ ディスプレイ ○ プリンタ ○ プロッタ	文字	ディジタル
	入力系 ○ キーボード ○ ディジタイザ	図形	ディジタル
	その他 ○ FAX	画像	ディジタル
	○ マイクロフィッシュ	画像	ディジタル
	○ VTR	画像	アナログ
聴覚	出力系 ○ 音声応答装置 入力系 ○ 音声認識装置	音声	ディジタル アナログ
触覚 嗅覚 味覚	――――	圧力 臭 味	――

2.1.2 対象とする情報メディア

マン・マシン・ユーザ・インタフェイスにおいて対象とする感覚メディアは，表2－1からもわかる様に，文字，図形，画像，音声である。これは人間の情報伝達手段の主たる感覚が，視覚，聴覚であることから当然であろう。今後，マン・マシン・ユーザ・インタフェイスに人間の感情など心理的な情報伝達が必要とならない限り変らないと思われる。

形式メディアとしては，コンピュータとの接続を考えた場合，ディジタル・データ表現が主となるであろう。一部アナログ表現が残るのは，たとえば既存のマイクロフィッシュ検索装置の様に，蓄積されているデータが，コンピュータ内に取込まれず，マイクロフィッシュ（蓄積媒体）から直接光学的にハード・コピーを取る装置，あるいはアナログ的な出力デバイス（スピーカ）とのインタフェイスを持つ音声装置などにおいてであろう。

次に，これから述べる文字，図形，画像の定義付けを行なう（表2－2参照）。

表2－2　感覚メディアと形式メディア

感覚メディア	概　　念	形式メディア
文　　字	英数字，カナ，漢字 （→各国文字）	ディジタル・データ表現であり， かつコード化データである
図　　形	線　　画 （例：グラフ）	ディジタル・データ表現であり， かつベクトル・データである
画　　像	絵，　写　真	ディジタル・データ表現であり， かつバイナリ・データである

コード化データとは，情報がコード化されており，かつコンピュータ内で復号（デコード）化することによってコンピュータが情報内容を解釈できるデータである。ベクトル・データもコード化データの一種とも言えるが，線分表現用データで，始点，距離，方向からなるパラメータで表現している点が異なる。

バイナリ・データは，二次元情報を格子状に区切り，各セル対応にビット列で表現したデータである。

画像情報はデータ量が多いため，当該バイナリ・データをMH（Modified Huffman）方式等によってコード化し，データ圧縮を行なう。この場合，コード化はされているが復号化してもコンピュータで解釈できるわけではないので，先に述べたコード化データの範ちゅうには含めない。

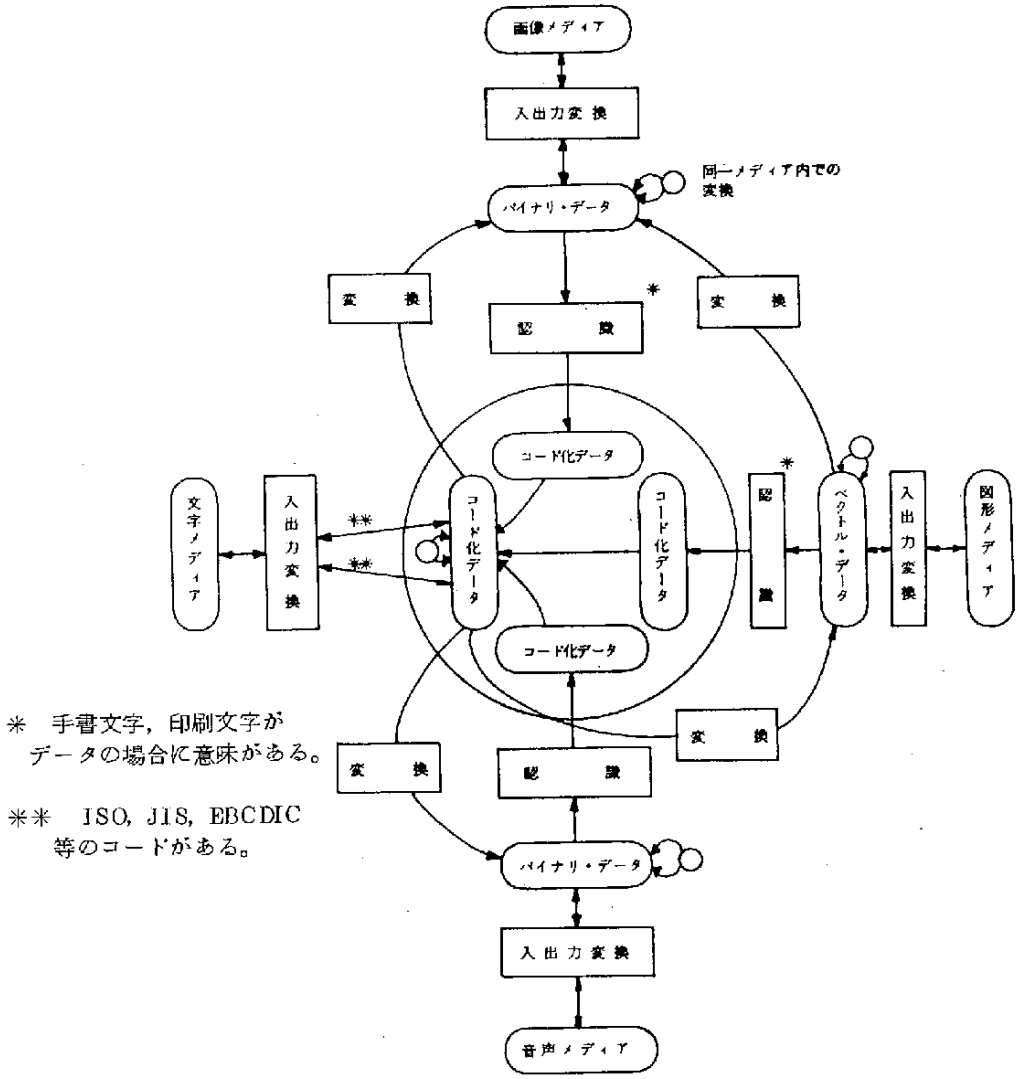
2.1.3 情報メディアの変換

対象とする情報メディアを統合して取扱おうとする場合，以下の理由でメディア変換が必要と

なる。

- マン・マシン・インタフェイスの向上を図る上で，特定の感覚メディアに変換したい場合
  例：文字情報→音声メディア
- 装置メディアの物理的な制限により，変換しないと出力できない場合　例：文字情報→ファクシミリ出力
- 装置内での取扱いの容易さから，形式メディアを統一したい場合　例：アナログ→ディジタル
- 形式メディア内でのコード変換を行ないたい場合　例：EBCDIC コード→ISOコード

図２－１に各メディア間の変換の概念を示す。



＊　手書文字，印刷文字がデータの場合に意味がある。

＊＊　ISO, JIS, EBCDIC 等のコードがある。

図２－１　メディア変換の概念

2.2　情報メディアの統合に対する要求

2.2.1　背　　景

従来コンピュータ・システムは文字メディアをコード化データ形式で表現し，数値データ処理，定型業務処理を主体として行なわれて来た。利用者も専門的な人間がほとんどで，マン・マシン・ユーザ・インタフェイスの観点からすると，コンピュータに対して人間の方が合せる"コンピュータ主体型"であった。この点で，利用者側としてはコンピュータを身近かなものとして使い切れず，今一つ満足出来ないところがあった。

ところが，オイル・ショックなど社会環境の変化から，また製造部門のＥＤＰが一段落してきたことなどからオフィスの生産性向上が注目されだした。この問題を解決すべく「オフィス・オートメーション」という言葉が標傍され，ファクシミリ，日本語ワード・プロセッサなど単独製品が世の中に出始めてきた。

この様な状況下で，"人間主体型"に対する要求は急激であり，画像，図形，文字，音声の各メディアが統合され，より利用者がコンピュータを意識しないですむシステムの出現が望まれている。

この動きは，オフィス内の作業だけでなく，設計部門における設計作業のＣＡＤ化，一般生活における情報提供サービス等々に波及している。

2.2.2　マルチ・メディア・サービスの例

前項で述べた様に，情報メディアの統合に対する要求は，オフィス分野をはじめとして，各分野から出始めている。

本項では，各分野において考えられる適用例，およびどの様なマルチ・メディア形態となるか，その概要を示す。

(1)　オフィス分野

オフィス内の業務としては，大きく分類すると，思考・判断業務，文書処理業務，連絡業務，計算処理業務となろう。このうち，思考・判断業務は人間が行なうべきものであり，それを補助する形でその他の処理が存在する（図２－２参照）。

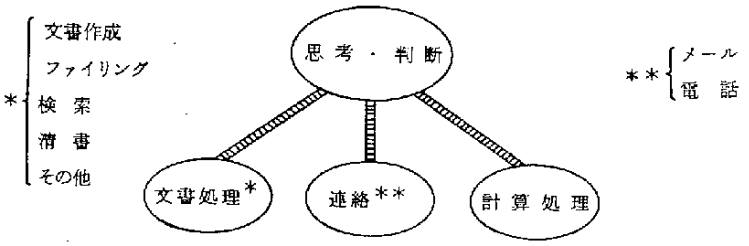


図２－２　オフィス業務

計算処理業務については，既にＥＤＰ化されており，さらに近年は漢字も取扱える様になり，より利用者に身近になっている。

したがって，文書処理業務，連絡業務にコンピュータを導入することが望まれているわけである。

a） 文書処理業務

本業務は，文書作成，ファイリング，検索，編集，清書等がある。文書自体は，図２－３に示す様に各メディアが混在しており，これらを統合して上記処理を行なえる必要がある。

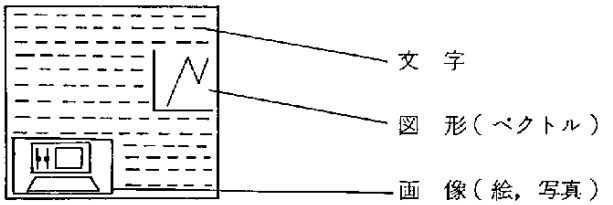


図２－３　文書の例

b） 連絡業務

連絡業務としては，郵便，電話，会議等がある。

以上の業務での適用例およびマルチ・メディア形態について，表２－３に示す。

表２－３　オフィス分野での適用例

適用例	概　　要	感覚メディア	主要装置メディア
電子ファイリング・システム	・既存の文書を画像入力し，蓄積する。 ・文字ベースで検索し，必要情報を得る。 （特許情報蓄積検索システム）	文　字 画　像	・イメージ・スキャナ／プリンタ／ディスプレイ
電子ドキュメント作成管理システム	・ＷＰにより文章作成，ホストにあるＤＢを使って，グラフ作成，人手作成の絵をイメージ・スキャナから入力する。 ・ディスプレイ上で，各情報の編集を行なう。 （拡大／縮少，切出し／埋込み，回転） ・同一形式メディアで蓄積する。	文　字 図　形 画　像	・ＷＰ（ワードプロセッサ） ・イメージ・スキャナ ・複合プリンタ／ディスプレイ
社内電子メールシステム	・郵送はファイル間で行なわれ，受信側は，適当な時間にディスプレイ上で読む。特に必要なものだけハードコピーをとる。	文　字 画　像 （図形） （音声）	・上記システムのサブセット（音声応答，認識）
インテリジェントＦＡＸシステム	・ＦＡＸの画像入出力機能，通信機能にインテリジェンスを付加したシステム。 （ＷＰ＋ＦＡＸ，またはＦＡＸ＋ＯＣＲ）	画　像 音　声 文　字	・ＦＡＸ ・ＷＰ／ＯＣＲ ・音声応答装置
電子ディクテーティングマシン・システム	・音声認識機能によって，口述から文書作成を行なう。清書も同時に行なわれる。	音　声 文　字	・音声認識／応答装置

(2) 設計分野

設計分野のなかでも，コンピュータの論理設計から，シミュレーション，実装設計については，設計方法の仕組が解明されており，ＤＡ（Design Automation）として15年以上前から実用化されていた。

しかしながら，構造設計の分野では，現在，自動設計は不可能で，コンピュータを導入する場合，設計者がコンピュータと会話を行ない設計を進めて行かなければならない。上記システムは一般にＣＡＤシステムと呼ばれ，各社開発が急である。

大型ディスプレイ，ライト・ペン，ディジタイザ等を用いて設計を行ない，三次元表示機能によって設計をまとめていくもので，感覚メディアとしては，図形と文字が統合される。マン・マシン・ユーザ・インタフェイスの向上のためには，より簡単なコマンドで設計を進められることが望ましく，そのためにはデータベースの充実が必要である。

(3) 一般生活分野

この分野において，マルチメディア形態が要求されるのは，一般の人々が各種サービスを受ける場合が主となるであろう。

利用者側から情報を入れるような場合としては，小売店等のオーダ・エントリ・システムが考えられる。

上記の形態が実現されることによって，利用者は必要な時に必要な情報を，音声あるいはハード・コピーの形で手に入れることが出来るようになる。

家庭用の場合は特に装置メディアの制限（電話，ＦＡＸなど）から，音声と画像メディアが主となるであろう。

表２－４に，具体的サービス例によって分類した適用例，概要を示す。

表２－４　一般生活分野での適用例

適用サービス例	概要	感覚メディア	主な装置メディア
不動産情報	センタに物件情報ファイルを持ち各支店のＦＡＸ，ディスプレイ端末に出力する。あるいは，家庭のプッシュフォンからの問合せに音声で答える。	画像 音声 文字	FAX ディスプレイ 音声応答 プッシュフォン
レジャー・催物・交通の案内，予約	レジャー施設，催物，交通の案内，予約。家庭のプッシュフォンからの問合せに音声で案内を行う。予約もできる。（FAXによる予約券の入手）	音声 画像 文字	ＦＡＸ プッシュフォン
証明書類の交付	住民票，印鑑証明等，比較的法的規制の緩い書類の在宅交付。	音声 画像	ＦＡＸ
オーダ・エントリ・システム	支店から本店に対し，オーダ品の内訳等の連絡。本店での在庫管理売筋情報の収集，支店への連絡。	音声 画像 文字	ＦＡＸ プッシュフォン
総合情報検索システム	現在試行されているCAPTAINシステムに，ハード・コピー機能を設けたシステム。さらに検索能力を高める。	画像 文字	ＴＶ ＦＡＸ

2.3　情報の蓄積・伝送

主としてコード化データ（数値データ，および英字，カナ文字，漢字などのコード化された情報），音声情報および画像情報について，それらの蓄積・伝送方法の概略を述べる。

2.3.1　情報の蓄積

(1)　コード化データ

コード化データは，よく知られているように，そのファイル媒体として磁気ディスクが最も一般に用いられる。保存用ファイルとしては磁気テープが使用されるが，日常の処理を受けるファイルは，そのアクセスがランダムに行なえることから磁気ディスクが利用されている。磁気ディスクへの記録は年々高密度化技術が開発されて来ており，したがって１スピンドル当りの容量が増大し，800ＭＢ（メガバイト）程度のものが出現している。記録の高密度化に伴って単位記憶当りの記憶コストの低価格化が実現されている。

さらに大規模な情報蓄積用としては超大容量記憶装置（ＭＳＳ）が利用される。ＭＳＳは一般にそれ自体が階層化された記憶システムをなして居り，上位メモリとしては磁気ディスクを，下位メモリとして多数の磁気テープを備えており，必要に応じて磁気テープから磁気ディスクへと情報の自動読み出し（または逆に磁気テープへの書き込み）を行い，コンピュータ側のアクセス要求に応ずる構造となっている。平均アクセス時間は約15秒，容量は数百ＧＢ（ギガバイト）が達成されている。

オフィス等ではフロッピ・ディスクが安価で操作し易い点で大いに利用されている。これはマ

イコンやオフィスコンで情報の蓄積ファイルとしての利用の他，従来の紙カード，紙テープに代る入力媒体としての役割も果している。両面倍密度のものは１ＭＢ以上の記憶容量がある。この他20㎝（８インチ）固定ディスクが登場しており，オフィス・コンピュータなどのデータベースへの使用がなされている。20㎝ディスクの容量の拡大も進められており，現在数十ＭＢであるが，80年代中ばまでに数百ＭＢに達するとの予想もなされている。

(2) 音声情報の蓄積

一般に音声は，よく知られているように録音テープを用いて蓄積できる。しかしながら，コンピュータ制御のもとで音声情報を扱うためには，ランダム・アクセス性が必要である。すなわち，極力小さい待合せ時間で音声（音楽なども含む）の開始，終了の制御が可能であることが必要である。この他，当然のことながら音声品質，蓄積容量，経済性も要求される。音声情報を蓄積しておき，必要に応じて出力する，いわゆる音声合成には，大別して表２－５の３つの方式がある。

表２－５　音声合成の主な方式

方　　式	蓄積単位	音声品質	情 報 量
(a) 録 音 編 集	単語・文節	○	20 ～ 100kビット/秒
(b) パラメータ編集・分析・合成	単語・文節	○	1 ～ 10kビット/秒
(c) テキスト合成	音節・単音	×	100ビット/秒

a) 録音編集方式

単語あるいは文節を単位として，音声をＰＣＭなどの形で録音記録しておき，必要なときにそれらを選択編集して１つのまとまった文章を作る方式である。国鉄の電話座席予約システムなどで実用化されている。

b) パラメータ編集分析合成方式

単語や短文を音声波形そのものではなく，音声パラメータの形に情報圧縮して蓄積しておき，必要なときにそのパラメータから音声を合成する方式である。音声パラメータの求め方と音声復元の方法とは対をなすものであり，種々の方法が考案されている。

c) テキスト合成方式

音節や単音を単位として記録しておき，必要に応じてこれらを組み合わせて文章を作る方式である。しかし，アクセント，イントネーション，ピッチなど韻律情報を作り出すためには，出力したい文章（＝音節列）に対し，構文解析，意味解析を必要とし，未だ研究段階にある。上記a)やb)で単語や文節単位の音声情報を蓄積しておくためのファイルとしては，システムの規模によっても変るが，半導体メモリや磁気ドラム，磁気ディスクが利用される。

なお画像応答システムなどにおける動画の付帯音声や音楽は，ＶＴＲやフィルムのサウンド・トラックに録音しておき，これを動画とともに端末に出力することにより，その目的が達成

される。

(3) 画像情報の蓄積

画像情報と一言で言われるものに，多様なものを含んでおり，それらは大別して図２－４のように分類される。このうち，最も単純なものは静止画のうち２値画像であるが，それでも一般にはＡ４サイズ１枚で0.７ＭＢ（ 10本／㎜の場合）程度の情報量となる。色彩画であるリモート・センシング画像では，4.5～128ＭＢ／画面程度の情報となる。このように，一般に画像情報は１画面当りの情報量が大きいのが特徴である。したがって，一方において情報圧縮の技術が重要であるが，同時に情報蓄積ファイルとしては大容量のものが必要となる。

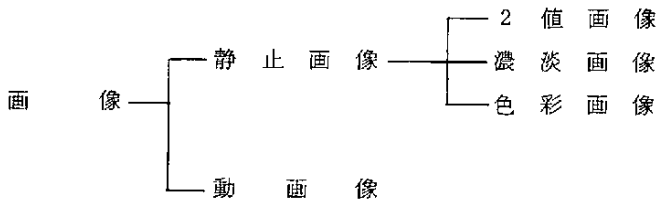


図２－４　画像情報の分類

２値画像のうち，次のような場合には，その蓄積・伝送を行なう上で非常に簡単になる。すなわち，

a） すべてがコード化データとその簡単な配列ルールによって再現できるものの場合

これは，例えば文章が整然と書かれた１ページの情報である。文章中の各文字は，文字コードによって表現され，文字の並び方も１行何文字，何行／ページという情報だけで表わされる。（改行記号や句読点その他の記号もコード化データである。）この場合には，文字発生器（文字コードに対して，その文字の具体的な形の情報を出力する装置）によって簡単に再現できる。

b） 線分の組み合わせで再現できるものの場合

線分は始点の座標と，方向および長さによって表わされる。この表現方法はベクトル表示と呼ばれることがある。この表現法ですべての情報が記述されている画像を特に「図形」と呼び，一般の画像と区別することもある。図形にも非常に複雑な図形があり得るが，実用上は，一般の画像として扱うよりは情報量が少なくて済むことが多い。ＸＹプロッタやグラフィック・ディスプレイなどは，図形情報を扱うに適した仕様となっている。

画像情報は，その用途によって扱いや処理内容が多様であり，当然そのための手段も多様である。以下に主なる画像情報の蓄積手段の概要を述べる。

① マイクロフィッシュ検索装置

文書，図面，写真などの静止画像情報を，マイクロフィッシュに縮小して蓄積でき，かつ簡単な操作で検索できる。図書館や，企業内の図面管理室，資料管理室等で利用されている。検索装置そのものでは，目的画面のフィッシュ上のアドレスを指定されることにより，該当

画面を自動的にスクリーンに投影する機能がある。コンピュータと接続することにより，多角的なキーワード指定による検索など高度な検索システムが実現されている。また，これと画像入力装置とを組み合わせることにより，画像伝送を伴う放送・教育など情報提供システムの画像ファイル機能を実現することができる。表２－６にマイクロフィッシュ検索装置の仕様の１例を示す。

② 光ディスク

光ディスクは，レーザー光を用いて情報を記録・再生するディスク装置で,大容量画像ファイルや大容量コンピュータ・メモリーへの応用が考えられている。記録・再生には幾つかの方式が開発されているが，ほぼ共通の特徴は，

（i）大容量である。

（ii）非接触で記録・再生ができる。

（iii）ゴミやホコリなどの影響に強い。

（iv）記録後すぐに再生ができる。

（v）追加記録ができる。

（vi）高速ランダム・アクセスができる。

（vii）保存寿命が長い。

などである。ただし，現在の所書き換えはできない。情報の容量は直径 30 cmのディスク面で約 10^{10} ビットという大容量である。ドキュメントなど静止画用に利用する場合数万ページが収容できる。また動画用にも利用され，ＳＲＩ社のものの場合直径 30 cmディスクで約 30 分の画像を記録できる。

コンピュータによる制御と組み合わせることにより，画像データの検索の他，画像の切貼り編集による新たな画像の創出や，画像の修正・追加などが可能である。

表２－６　マイクロフィッシュ検索機の仕様例
（日立製作所ＭＰ６００Ａ）[4]

項　　目	仕　様　・　機　能
使用フィルム	COSATI（5×12），COM（7×9），NMA（7×9） （その他フィッシュについてもご相談に応じます。）
フィッシュ・サイズ	105 × 148.75（mm）
収納フィッシュ枚数	600枚（20カセット）
平均アクセス・タイム	5　秒
スクリーン・サイズ	312 × 460（mm）
プリント方式	ドライ・シルバー方式
プリント・サイズ	Ａ３，Ａ４
プリント用紙	ドライ・シルバー用ロール・ペーパー（Ａ３換算３２０枚収容）
プリント・スピード	16秒／枚，　連続約１０秒
投影倍率	COSATI　20倍，　NMA　25倍

③ その他のメモリ

上記の他，システムの目的によって，磁気ディスク，磁気テープも画像情報ファイルとして用いられる。また，ビデオ・テープ（ＶＴＲ）も動画の蓄積ファイルとして利用される。コンピュータ制御により要求に応じマシン・ハンドが駆動され，目的のビデオ・カセットを所定のＶＴＲに装填する装置（ランダム・アクセス動画ファイル装置）も実現されている。

この他，画像ファイル装置から検索された画像情報の一時的記憶には，磁気ドラム，磁気ディスク等の他，コア・メモリ，ＩＣメモリ，磁気バブル，ＣＣＤ素子等も用いられる。画像データは１画面当りの情報量が多いために，一時的な情報だけでコンピュータの主メモリが不足する場合が多く，したがって外部記憶装置が必要となるからである。

2.3.2 情報の伝送

既に述べたように，コード化データ，音声情報，および画像情報とも，マシン系内ではディジタルの電気信号に変換して扱うことができ，これらの情報の伝送も，原理的には従来からある電気通信技術によって実施することが可能である。しかしながら，伝送コスト，スピード等の面からの制約，発展経過の歴史的事情等から，実際には夫々の場合に個別にそれに適した情報圧縮の工夫や，適した伝送方法がとられている。音声情報だけの伝送については電話が古くから用いられている。ファクシミリの場合には高速伝送のための各種の工夫がなされて来ている。

一方，伝送路，伝送網についての伝送コストの低価格化や性能の向上に対しても努力が重ねられ，革新が図られて来ている。

(1) 光伝送技術

ガラスなどの透明な物質で作られたファイバーの中を光信号を通すことにより，遠方に情報を送るのが光伝送である。当然ながら，発光素子と受光素子とにより，電気信号から光信号への変換および，逆変換が必要である。情報の伝送に光を用いることの利点は次のものである。

(i) 従来の導電体ケーブルに比べ，軽量ケーブルで大量の情報が伝送できる。

(ii) 電気的雑音の多い所に配線しても雑音の影響を受けない。

(iii) 伝送損失が少ない。

これにより，光伝送には大きな期待が寄せられており，30～100Ｍビット／秒の光伝送システムはすでに実用レベルである。１Ｇビット／秒程度まで実用化可能と考えられている。

(2) 電話網のディジタル化

現状の交換機は音声のアナログ信号をそのまま交換するものが主体である。一方，アナログ信号をサンプリングした２進符号化パルス列（ＰＣＭ符号）で伝送する。いわゆるＰＣＭは，市外電話回線にはすでに導入されている。したがって，ＰＣＭ情報をそのまま交換する時分割交換機に切りかえることにより，電話網のディジタル化が期待される。

ディジタル化によって，

(i) 網コストの低減

(ii) 伝送品質の向上

(iii) 音声に限らず，コード化データや画像情報の通信も可能となる。

これによって，従来メディアごとに独立であった機器が統合される方向づけができ上ると考えられる。

(3) 構内網

構内網については，広域の通信のような既成網からの制約が少ないので，オフィス内で有用な各種の新電話サービスのほか，オフィス内の各種の情報機器，すなわちＦＡＸ，ビデオ端末，タイプライタ，計算機などの接続とそれらの有効活用が早期に実現されるものと考えられる。

2.4 システムの事例

2.4.1 既存システムの例

システムの目的の中に，情報メディアの統合をも包含して開発された既存の情報処理システムの例として，画像応答システム（ＶＲＳ）について，その概要を以下に示す。

(1) 概　要

近年，工業化社会の進展に伴って膨大な情報が氾濫しており，今後もいっそうこの傾向が強まると予想される中で，情報を必要とするときに，利用者が自由に選択できるような，すなわち受け手主導形で必要な情報を取得できる通信システム実現への期待が高まってきている。

これに応ずる手段としては，コンピュータを利用した様々なシステムが考えられるが，更に，視聴覚機能を併用した画像情報システムが有効であり，そのような形態のシステムとして画像応答システム（ＶＲＳ）がある。ＶＲＳは一般のテレビジョン受像機とプッシュフォンなどを端末とし，これを既設の電話ケーブルに中継器を挿入した伝送路により画像センターと接続し，利用者が簡易な操作で，必要な情報を必要とするときに会話形式で，画像・音声による豊富な形で取得できるシステムである。日本電信電話公社内での利用実験を通じて，このシステムの有用性及び基本的技術を確認するとともに，今後更に継続して技術開発が必要な幾つかの要点が把握された。

(2) ＶＲＳの特徴

画像通信システムは，そのサービス形態から表２－７に示すように分類することができる。ＶＲＳは情報センタから端末に向かって単方向に流れるＣ－Ｅ（Center－to－End）形システムの中で，端末ごとの要求に応じて情報を提示する個別リクエスト形システムである。この個別リクエスト形システムは，同表から明らかなように未開拓な領域であるが，次に述べるような数々の特徴をもっている。

表２－７　画像通信システムの分類

区伝	項目 / 伝達方式				サービス形態	適用例
端末相互間	交換機能なし				専用形	ITV(工業用テレビジョン)，テレビ会議，ファクシミリ
端末相互間	交換機能あり				交換形	テレビ電話，電話ファクシミリ
センター端末間	片方向の伝達	下り方向（センタから端末へ）	同報系	上り回線なし	配達形	テレビジョン放送，テレビジョン再放送
センター端末間	片方向の伝達	下り方向（センタから端末へ）	同報系	上り回線あり	放送応答形	意見調査，通信教育
センター端末間	片方向の伝達	下り方向（センタから端末へ）	個別系		個別リクエスト形	CAI(Computer Assated Instruction)，検索情報検索・案内，ショッピング，予約
センター端末間	片方向の伝達	上り方向（端末からセンタへ）			情報収集形	遠隔監視，テレメータ
センター端末間	双方向の伝達				双方向形	カウンセリング，遠隔(医事)診断

注：アンダラインは既に行なわれているサービスを示す。

① 加入者数と加入者の受ける利便は無関係であり，加入者数の小さい時点でも，その効用は変わらない。

② 個別の要求に個別に応ずるシステムであるため，利用時間や情報の選択に対する自由度が高い。

③ マン・マシン・システムであるため，相手を意識せずに利用できる。

④ 特に視覚情報は，情報をパターン認識的にとらえることができる特質をもっており，C－E形システムとしてなじみやすい。

更に，ＶＲＳ固有の特徴として，

⑤ 静止画情報の伝達は一瞬のうちに行なわれるので，その即答性は会話形サービスに適している。

⑥ 静止画だけでなく，動画や音声，それらを組み合わせたサービス（複合静止画サービス）も提供できる。

などを挙げることができる。

(3) システム構成の概要

ＶＲＳは一般テレビジョン受像機とプッシュフォンなどを端末とし，これとセンタとを既設電話用又は広帯域平衡対ケーブルを用いた伝送路で接続した構成をとっている。実験システムの基本構成は図２－５に示すとおりである。すなわち，画像センタ，交換機，伝送路及び端末の四つのサブシステムから構成される。その動作概要は次に述べる通りである。

① 利用者がプッシュフォン（又は簡易キーボード）からサービス別に定められた番号（"1"は静止画サービス，"２"は同報動画サービス，"３"はリクエスト動画サービス）をダイヤルすることにより，端末がセンタに接続され，それぞれのサービスに応じた要求が受けられる。

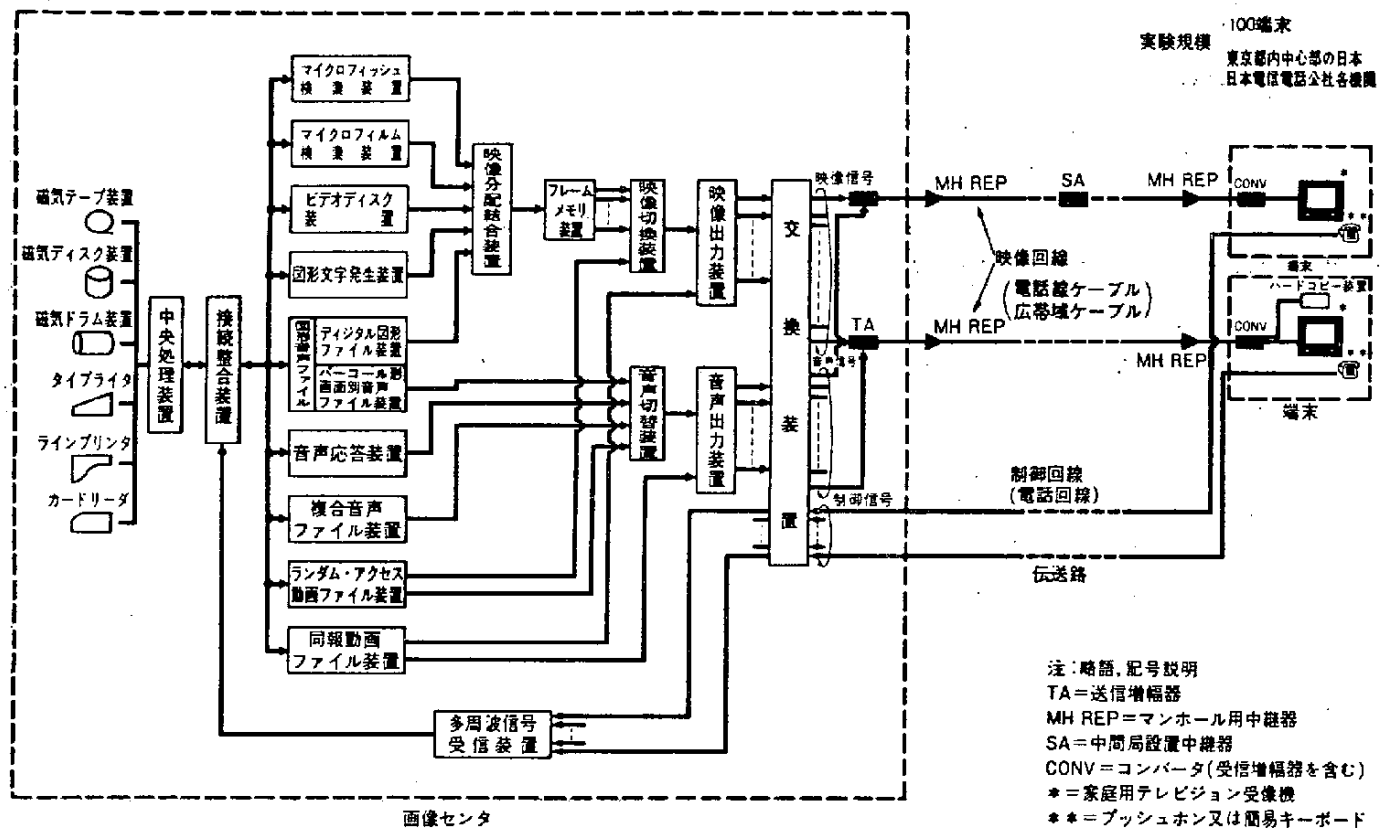


図２－５　実験システムの構成

② 端末からの要求はプッシュフォンのボタンを操作することによって行なわれる。センタではプッシュフォンボタン信号を受信し，コンピュータによって要求内容を解釈し，それに対応した画像情報あるいは音声情報をファイル装置から選択抽出する。

③ センタのファイル装置から選ばれた画像情報と音声情報は，結合されてカラー映像ベースバンド信号として端末に伝送される。

④ 端末では，受信した信号をコンバータによってテレビジョン放送電波と同形式の信号に変換し，一般のテレビジョン受像機の空きチャンネルで受像する。

各種画像・音声ファイル装置の概要を表２－８に示す。

表２－８　各種画像・音声ファイル装置の概要

区分	装置名	ファイル内容	容量など	平均アクセス時間	記事
静止画	マイクロフィッシュ検索装置	静止画	5,940齣/台	3.8秒	――
	マイクロフィルム検索装置	静止画	7,200齣/台	0.45秒	――
	ビデオディスク装置	静止画	900フレーム/台	1.0秒	――
	図形文字発生装置	文字 図形 カラー：７色	約10,000齣 外部記憶装置のデータエリア容量による。	文字：28ms/10文字 図形：10ms/10ドット	文字の大きさ 標準文字：30×30ドット 小形文字：20×20ドット 文字種類：約2,300文字
	ディジタル図形ファイル装置	図形 カラー：有彩色13色，無彩色4色	3,000～4,000画面	0.1秒	――
動画	16mm自動装塡映写装置	動画	最大10巻装塡可能 １巻：最大70分	――	フィルム用 カラーカメラ使用同報動画用
	3/4インチビデオテープレコーダ	動画	１巻：最大60分	――	同報動画用
	ランダムアクセス動画ファイル装置	動画	カセットテープ：120巻 再生用ＶＴＲ：12台	――	複合静止画用 リクエスト動画用
音声	音声応答装置	システムメッセージなど	最大メッセージ数：400	0.5秒	音声単語を文章に編集・出力
	テープレコーダ	同報動画番組案内用音声	最大７分エンドレス	――	BGMなど （同報動画休止中出力）
	アナウンスマシン	障害・サービス休止，試験中などの情報	４トラック/１台 最大14分エンドレス/トラック	――	センタ情報用
	バーコール形画面別音声ファイル装置	画面別音声（人声のみ）	15秒/画面×約5,000画面	0.1秒	静止画の画面別説明用
	複合音声ファイル装置 （ランダムアクセス形カセットテープ再生機）	画面別音声	最大収容メッセージ398種類	18秒	――
	（磁気ディスク再生機）		トラック数50トラック最大11.5秒	0.5秒以下	
	（エンドレステープレコーダ）		４トラック/１台 最大14分エンドレス/トラック	――	

注：BGM＝バックグラウンドミュージック

(4) 利用実験とその結果

個別学習，情報案内，クイズ，ゲームなどのサービス番組を用いて利用実験を行なうとともに，利用者に対するアンケート調査などによりＶＲＳの有用性が確認され，かつトラヒック特性や利用者習性などについてこれまで知られていなかった興味あるデータが得られつつある。利用実験に用いたサービス番組の一例を表２－９に示す。以下にこれまで得られた一部の結果を示すが，更に機能面，利用対象の拡大を図り，より広範な調査を行なうためシステムの機能追加を行ない引き続き利用実験を含めた調査・試験を実施している。

① システム機能に対する評価

このシステムの機能に対する要求度・満足度について既存のテレビジョン放送受信時での要求の度合，及びＶＲＳ使用後の機能満足度に対してアンケート調査を実施した結果を図２－６に示す。以下に述べるように，このシステムの基本的機能が有効であるとの意見が多い。

(a) 利用時刻の任意性：利用者が利用したい時に任意に利用することができる。

(b) 番組選択性：多数の番組から欲しい番組を自由に選択できる。

(c) 番組反復性：同じ番組を繰り返し見ることができる。また番組内で同じ画面を繰り返し見ることができる。

表２－９　実験サービスに用いた番組例

区分		内容
静止画	教養	英語入門
		数学演習
		囲碁入門
		その他一般教養
	各種案内	電話商品案内
		宿泊施設案内
	娯楽	ゲーム
		クイズ
動画		教養，事故知識など

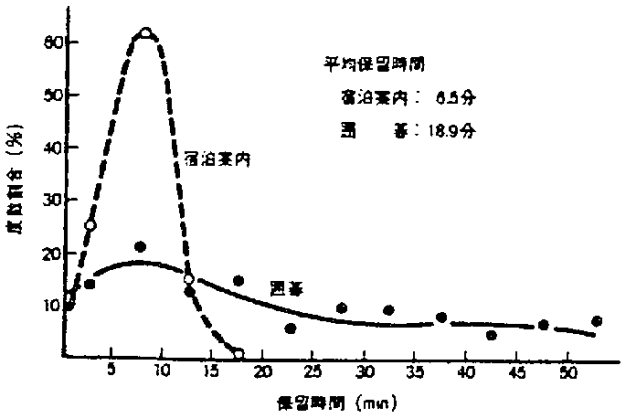


図２－６　保留時間分布

(d) 利用時間の任意性：番組の進行を利用者のペースに合わせて進めることができる。

(e) 一般のテレビジョン受像機を用いているので親近感がある。

(f) 相手が機械であることより安心感がある。

② 利用傾向

(a) 呼はランダムに生起し，その分布は指数形である。

(b) 呼の平均保留時間はサービス内容により大幅に異なる。図２－７にその代表例を示す。

(c) 端末から入力を行なう間隔は平均値で個別学習形のとき 11 秒，情報検索形のとき９秒という値が得られ，両者のサービス形式に一応の差が認められているが，番組の内容あるいは提示画面内容そのものにかなり依存するものと思われる。

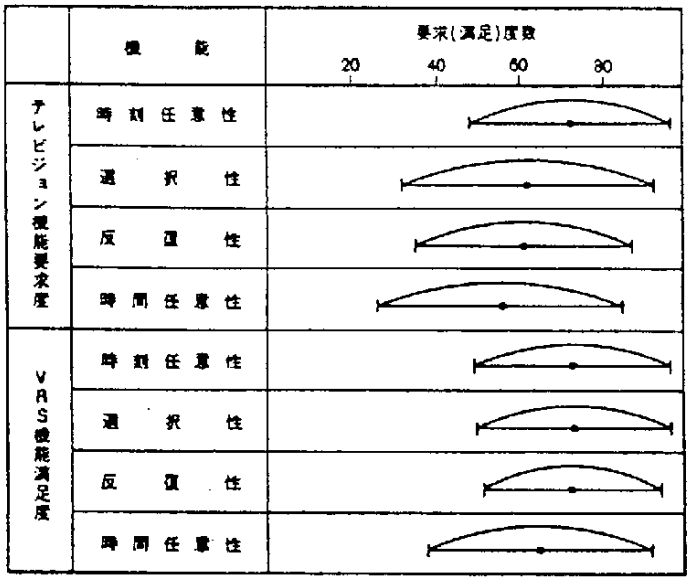


図２－７　機能要求度・満足度

2.4.2　研究開発中のシステムの例

情報メディアを統合し，使い易い情報システムを作成するための研究開発が各種の研究機関や企業で行なわれている。その結果の一端は各種の展示会等を通じて公開されている。それらの中からいくつかの例を以下に示す。

なお，以下に示す(1)～(3)の例は展示会で示された説明資料であり，(4)～(6)は展示会で公開されたシステムについて簡単に説明したものである。

(1) 光ディスクによる文書管理システム

光ディスクによる文書管理システム

Intelligent Document File System with Optical Disk

■背景および概要

「情報処理」はコンピュータの発達により大容量記憶・高速演算・オンライン処理など著しい発展をみせています。しかし，従来のコンピュータ処理の対象は，英数字・カナ文字などのいわゆる「コード化情報」に限られており，手書き文字・印影・図面などのイメージ情報は対象外でした。

今回，日立が開発した光ディスクは，超高密度の記憶容量で，手書き文書，図面など，コード化が困難なイメージ情報のファイルとして最適で，多くの用途が期待されます。

この大容量光ディスクを利用した日立文書管理システムは，次の機能・性能を備えています。

- 対象文書　手書き文書，印影，図面など（白黒2値）
- 記憶容量　A4版20,000ページ／ディスク両面，8ドライブ/システム
- 解像度　8本／mm
- 検索速度　0.3秒
- 端末数　1システム当たり8式（FAX，CRT，半導体レーザプリンタなど）
- イメージ編集機能　切出・合成・転記・作表など。

■特長および原理

記録媒体は，内面に金属薄膜を蒸着した2枚のガラス円板を隙間をあけて重ね合わせた構造で，ディスク両面への記録を可能とするとともに，記録面である金属薄膜を外部から完全に保護しています。

記録は，文書などの画像を電子走査して得たディジタル信号をレーザ光線にかえ，回転するディスクの金属薄膜に照射し，ディジタル状の孔をあけることにより行います。また再生は，記録面に微弱なレーザ光線をあて，孔の有無を反射光で検出しています。

記録再生には，半導体レーザを用いることにより，装置は小形化されています。

登録した多数の文書類から，必要なものを取出す検索機能や，光ディスクのファイル管理機能には，ミニコンピュータを使用しています。

ファイルされている画像は，必要に応じ，「切出」「合成」が行われます。たとえば，各帳票の任意の部分を電子的に切出し，それを合成して一覧表を作ることができます。また，手書き文字などをガイド情報として操作性の良い文書検索が行えます。

端末は，高解像度ディスプレイ・ハードコピー装置・キーボードから構成され，複数の端末を接続することも可能です。

このシステムの特長は次の通りです。

(1)記憶密度が高く磁気ディスクの約60倍です。
(2)高速度トラッキング制御により，追加書込みが自由です。
(3)半永久的な寿命です。
(4)マルチアクセス，リモートアクセスが可能です。
(5)切出，合成などの編集が可能です。

■将来の展望

光ディスクによる文書管理システムは，大容量ファイル（高密度記憶），容易な記録，リモートアクセス可能という特性から，省力・省スペースおよび業務の質の向上の面で高い効果が見込まれます。

また，他のオフィス機器と有機的に結合し，いわゆるオフィスオートメーションの一環として，オフィス業務の効率向上に大いに役立つことが期待されます。

図1．展示システム原理図

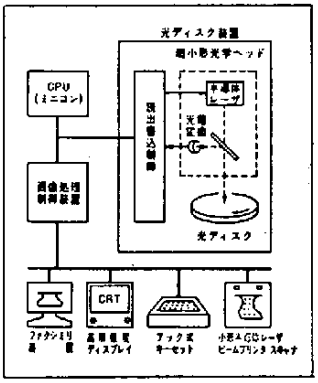
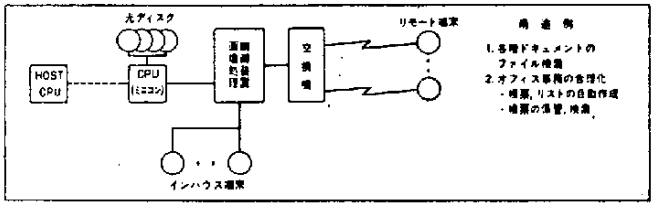
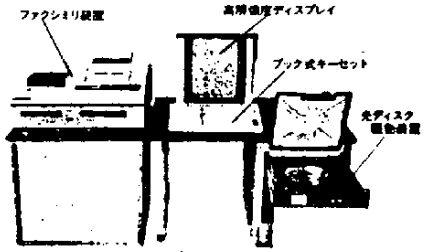


図2．システム構成例



"光ディスク"



HITACHI

タッチパネル付トレンド・グラフィック

Trend Graphic with Touch Panel

■概要

(1)タッチパネル部

タッチパネルは、ブラウン管表示端末の前面に装着した透明パネルであり、オペレータがブラウン管に表示された項目や図形に対応するパネル面を、直接指でタッチすることによって、その位置情報を得る装置です。

この位置情報とブラウン管に表示中の情報から計算機は、オペレータの要求を知り、必要な処理を行ったり、オペレータに必要な情報を表示します。

(2)トレンド・グラフィック部

このトレンド・グラフィックは、キャラクタ・ディスプレイにトレンドグラフ表示機能を付加した端末装置です。

豊富な文字表示形態制御機能による従来以上の見やすい文字表示に加えて、グラフ表示機能による視覚に訴える表示ができるため、情報の入力用としても出力用としても最適な端末として、幅広い業務目的に適用できます。

■特長

高精細の明るいカラーで、豊富なグラフや図形が簡単なコマンドで表示できます。また、ブラウン管の表示も直接指でタッチして入力できるので、誰にでも間違いなく、早く、疲れずに操作ができます。

(1)タッチパネル部

- タッチパネルは、指でタッチしたときにパネルの支持点に生じる力のバランスから位置を求めているので、特殊なパネルを必要とせず、構造が簡単で、適度に軽いタッチで入力できるため使用感にすぐれています。
- マイクロコンピュータを使用して位置計算を行い、外乱の影響を低減して精度の向上を図っています。
- 指による入力が可能であるとともに、入出力部が一体化しているため、使いやすくマンマシン性にすぐれています。
- 位置情報が連続的に得られるため、表示する項目や図形の位置を任意に選ぶことができます。
- ブラウン管上に表示する内容はソフトウェアで自由に選ぶことができ、従って地図などの表示やソフトキーボードとしての使用も可能となります。

(2)トレンド・グラフィック部

- 簡易なコマンドで多彩なグラフ表示が可能です。
- 高精細度カラーブラウン管の採用により2,560文字(80字×32行)の多量な情報表示が可能です。
- カラー7色、拡大文字、太い線、リバース、ブリンクなどの豊富な表示形態制御機能により、見やすく視覚に訴える画面表示ができます。
- データはフィールド単位で扱い、転送フィールド指定などにより、中央システムとの情報の授受を効率よく行うことができます。

図1．タッチパネル付トレンド・グラフィック

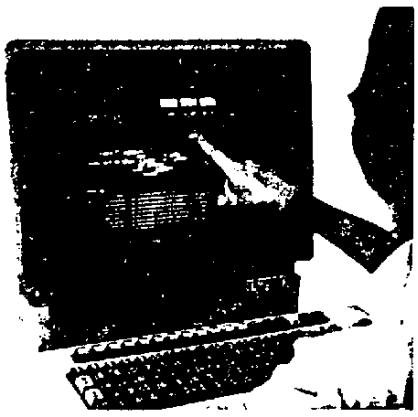
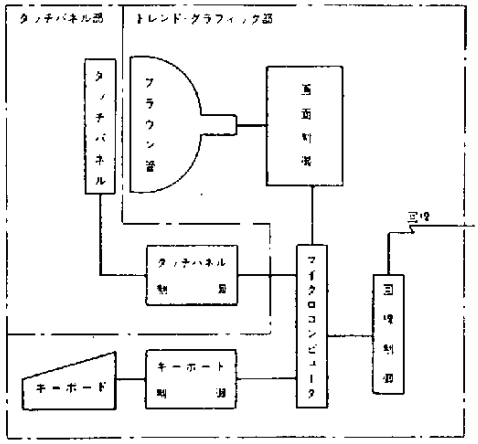


図2．構成図



オフィスワーク用 高機能情報検索システム

Stand-Alone Information Retrieval System for Office Use

■背景および概要

今日，オフィス・オートメーション・システムの重要性が強く叫ばれています。その主旨は，多様な文書情報の作成・管理・検索を，コンピュータの導入により自動化し，事務作業能率の抜本的改良を図る点にあります。事実，一般的な事務所では，「情報検索」つまり「仕事の推進に必要な文書類・図面類をファイルから取出す作業」に費される時間は，全事務作業時間の25%を占めると言われています。

オフィスで使用される情報には，手書き文字，図面など，コード化して計算機に入力することが困難なもの（イメージ情報）が多いことも事実です。

このようなイメージ情報を，マイクロフィッシュ化して，スペースを節約しつつ保管することは，従来から広く行われていました。マイクロフィッシュ化情報を容易に蓄積・検索・保守できるコンピュータ・システムを開発したならば，オフィス・ワークの能率向上に貢献できるのではないでしょうか。

日立では，このような考えのもとに，オフィスでの使用に適した高機能情報検索システムを開発いたしました。

(1)中央の大形コンピュータ・センタを必要としないスタンドアロン形であるため，小さなオフィスに設置できます。

(2)汎用の蓄積・検索・保守プログラムが一式準備されており，導入が容易です。

(3)キャラクタ・ディスプレイとの会話で，どなたでも簡単に即刻，検索が行えます。

(4)数万件におよぶ情報の中から，目的の情報を数秒で検索できる性能を持っています。

■特長

1件の情報は，図面・報票などのマイクロフィッシュ化が可能なイメージ情報と，タイトル（見出し），内容梗概，文書作成年月日，キーワード（索引用語）などのコード化すべきディジタル情報との組合せです。このシステムでは，両者が混合したものを取扱える機能を持っています。イメージ情報は，マイクロフィッシュ検索機画面に，ディジタル情報はキャラクタ・ディスプレイに表示します。

(1)マルチファセット／マルチレベル構造シソーラスを装備できます。

必要な情報を的確に検索するためには，情報の分類ならびに索引付けが正確に行われることが前提となります。このシステムには，蓄積情報および検索要求を，いろいろな複数観点（マルチファセット）から柔軟かつ多次元的に表現できる分類用キーワード体系（シソーラス）を装備できます。さらに，キーワード間の概念的な包含関係を表現できるマルチレベル構造も装備できます。

(2)自動論理検索機能を持っています。

面倒な論理記号式で，検索要求を表現する必要がありません。ブックキーの中から，関心のあるキーワードを拾い，そのボタンを押すだけで十分です。論理式の生成は，システムがシソーラスの構造を考慮しながら自動的に行います。

(3)柔軟なシステム定義機能を持っています。

使用する業務や目的にあわせて，パラメータを指定しつつ，システムの定義を行えます。ブックキー上でのキーワードの配置，情報1件あたりのデータ長，キャラクタ・ディスプレイ上での入出力画面様式などを柔軟に設定できます。この機能は，小さなシステムの中に，コンパクトに大量の情報を蓄積するためにも有効です。

■用途

(1)信頼性情報検索システムです。

製品の信頼性保証のため実施される各種試験の報告書は，きわめて貴重な情報です。このシステムの持つマルチファセット・シソーラスにより，このような情報を多角的に分類・蓄積が可能で，設計事務の能率向上に貢献します。

(2)エンジニアリング情報検索システムです。

各種プラント開発にともなう伝票，見積，技術図面のように，相互に関連した多種類の情報を，柔軟なシステム定義機能を利用しつつ効果的に蓄積できます。技術用語，価格，工期などによる検索も容易です。

(3)その他

図面，特許，新聞記事，雑誌記事，各種文献検索など多方面へ活用可能です。

図1．オフィスワーク用 高機能情報検索システム

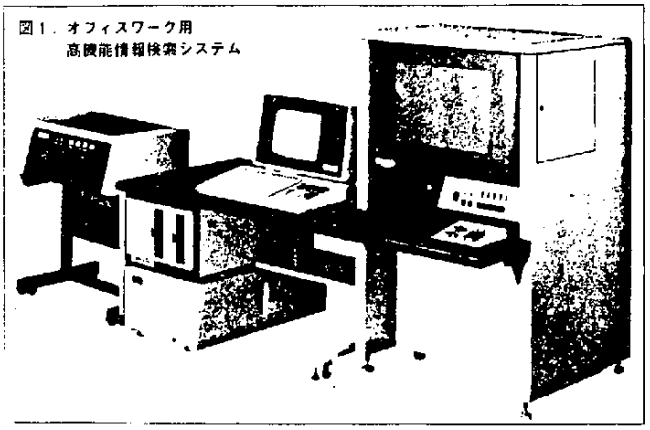
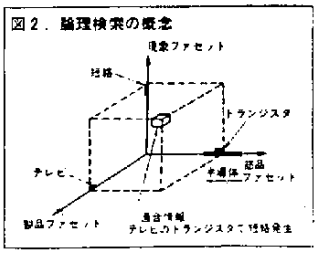


図2．論理検索の概念



HITACHI

(4) イメージエディティングシステム

第55回ビジネスショーに，リコーより参考出品されたシステムである(図2－8)。ワードプロセッサ機能とイメージ処理機能が複合化されている。

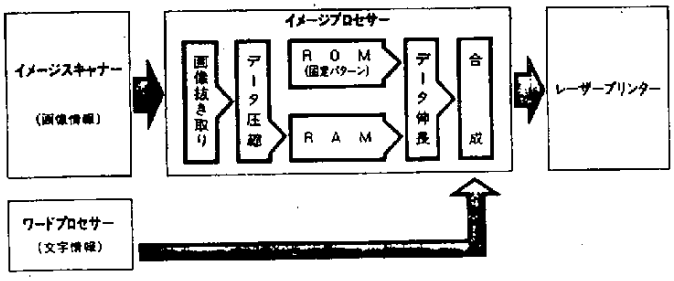


図2－8　ブロックダイアグラム

絵，地図，表混じりの文書作成ができる。さらに，絵などの配置が変えられる。定型パターンを使うことによって文書作成時間を短縮できる(図2－9)。

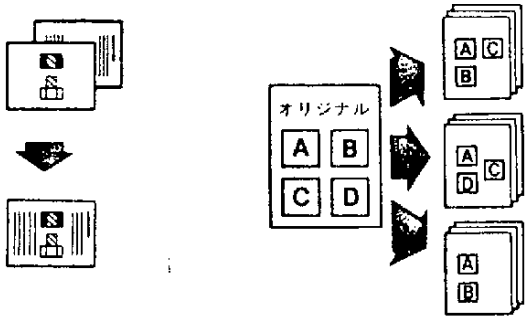


図2－9　機能概略

(5) FAXを使った図形合成システム

'80日立技術展に出品されたシステムである(図2－10)。文字，図形，画像メディアが複合されている。

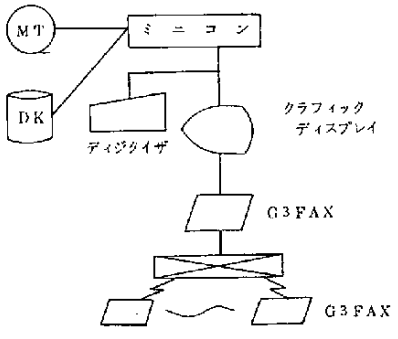


図2－10　ブロックダイアグラム

既にＭＴに入っている図形情報に対し，ディジタイザとグラフィックディスプレイを使って，修正，追記を行ない，画像情報に変換した後，回線経由でＦＡＸに送付する。

編集された図形を，画像装置であるＦＡＸに直接出力できる。

(6) インテリジェントファクシミリ

'79データショーに三菱より出品されたシステムである(図２－１１)。手書き文字，手書き編集マークと画像メディアが複合化されている。

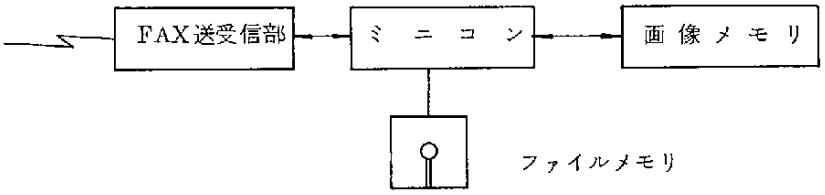


図２－１１　ブロックダイアグラム

ＦＡＸ自身に画像編集合成機能を持っており，入力はすべて，ＦＡＸメカニズムから行なわれるインテリジェントＦＡＸである。

帳票，文書等に手書きで修正を加えて入力すると，送信側のＦＡＸで編集を行い，修正済の帳票，文書が相手側に送出される。

2.5　今後の動向

現在は各メディア(特に音声，図形，画像)を個別に取扱うことができる装置が，出現してきて，それぞれマン・マシン・ユーザ・インタフェイスの向上に役立っており，さらにマルチ・メディア化の試行がなされ始めた段階に到っていると考えられる。

今後は，次の４つの面での展開が考えられる。

① 各々の感覚メディアを取扱う技術

② 装置上でのメディアの組合せの在り方

③ システムとしての在り方

④ 標準化

(1) 各々の感覚メディアを取扱う技術

音声応答においては，応答速度，連続音のなめらかさ等で人間の音声に近づくであろう。音声入力についても，不特定話者，連続音認識へと進み，現状システムにおける苛立ちは解消されていくであろう。

図形については，ベクトル発生パターンも豊富になり，オペレータからの入力は，より簡易になるであろう。

画像については，カラー化，階調の導入が進むであろう。また，特徴抽出，ノイズ除去等の技術も汎用システムに導入されていくであろう。画面サイズも大型まで取扱え，高解像度の画像が取扱えるであろう。

(2) 装置上でのメディアの組合せの在り方

ここ数年，試行錯誤の時代が続き，各種のマルチメディア装置が開発され，淘汰されていき，その中から本当にマン・マシン・ユーザ・インタフェイスを向上させる装置が出てくるであろう。

(3) システムとしての在り方

装置レベルでのマルチ・メディア化も進むが，同時にそれらの装置毎，あるいは，ホストとネットワークへ接続する形態が増加するであろう。ネットワーク自体の伝送速度も高まるが，それにも増して画像データのデータ量は膨大である。

したがって，ファイルは分散して持つ形態が増加してくるであろう。

(4) 標準化

現時点では，各システム個有に開発しているが，データ交換，ネットワーク・プロトコル等の問題から標準化が推進されることになるであろう。

1) 国外の動き

ＣＣＩＴＴ（国際電信電話諮問委員会）において，テレックス，ビデオテックスシステムという概念での標準化が進んでいる。

2) 国内の動き

電電公社ＤＣＮＡの第２版でマルチ・メディアへの対応を考えている。

以上述べた様に，マルチ・メディア化は試行錯誤を繰り返しながら，進展していくであろう。

参　考　文　献

1) 土師克己ほか3名：画像応答システムの音声ファイル装置，日立評論，Vol.60，№11，pp. 49－52（1978）

2) 新田義彦ほか3名：使いやすい会話型のマイクロフィッシュ情報検索システム，日立評論，Vol.61，№2，pp. 57－62（1979）

3) 土師克己ほか2名：画像応答システムの画像ファイル装置，日立評論，Vol. 60，№11，pp. 41－48（1978）

4) 日立製作所：HIPACS　日立マイクロフィッシュ検索システム，製品カタログCD－319R

5) 新井克彦ほか1名：オフィスオートメーション技術(3)－情報の伝送－，電子通信学会誌，Vol.64，№2（1981－2）

6) 電子通信学会：光伝送技術特集，電子通信学会誌，Vol. 63，№11（1980－11）

7) 穂鷹良介：講座データベース〔Ⅲ〕，〔完〕，電子通信学会誌，Vol. 63，№3（1980－3）

8) 篠田英範ほか1名：画像データベース，電子通信学会誌，Vol. 63，№12（1980－12）

9) 坂井利之：情報システムにおける音声の認識と合成，情報処理，Vol. 21，№8（1980-8）

10) 隈元釜夫ほか1名：キャプテンシステム－実験サービスを開始した文字図形情報ネットワークシステム－，情報処理，Vol. 21，№9（1980－9）

11) 日立製作所：1980日立技術展　（1980－11）

12) Computer Report，日本経営科学研究所（Jan. 1981）

13) 木村一嘉：図形情報システム－その設計方法と事例－，マグロウヒル好学社

14) (社)日本電子工業振興協会：フューチャ・オフィス・システムに関する現状と動向，電子工業月報，第22巻，第9号

15) 日本電信電話公社：データ通信網アーキラクチャ（DCNA）について

16) 1980年ビジネスショー・カタログ

17) 1980年データショー・カタログ

18) 日本工業技術センタ：インテリジェントシステム・ファクシミリの機能とアプリケーション，工業技術セミナー，№431

3. 新しい入力装置

3.1 はじめに

本章では，メディアとして音声，図形，画像などを利用する入力装置を中心に取りあげる。図形と画像については，これらをはっきりと区別して定義づけることは難しい。

通常，画像入力装置と呼ばれているのは，写真のような映像をデータ列に変換し，まとまりとしてコンピュータ等に入力するための装置であり，取込まれたデータは２次元以上の行列として表現される。

これに対し図形入力装置と呼ばれているものの定義は必ずしも明確ではない。例えば，手書き文字は典型的な線図形であるが，紙面上に書かれた文字の形状は，映像として「画像入力装置」によってコンピュータに入力される。他方，タブレット上に文字を書いている過程で，座標列を逐次コンピュータに入力する場合には，タブレットを図形入力装置と規定するのは，ほぼ通念になっており，この場合には線図形である文字は「図形入力装置」によってコンピュータに入力される。

しかし，この同じタブレットをペンタッチ式の漢字入力装置として用いる場合には，これを図形入力装置と呼ぶかどうかは疑問である。他の例として，機能的にはタブレットに似ているが，人間が指先で指した位置を検出し，その座標をコンピュータ等に入力するものにタッチセンサがある。

このような関連において考えると，タブレットも基本的には点の座標を入力する座標入力装置であり，コンピュータ内部においてこれらの点座標列を処理して線図形に対応づけた時にタブレットが積極的に図形入力装置として位置づけられると云えよう。

点の位置を入力することに重点を置く座標入力装置は，狭義の図形入力装置には入れ難いが，マン・マシン・コミュニケーションの観点からは多くの重要な利用法があるので本報告では一部の座標入力装置も含めて考察の対象とする。

3.2 入力装置の記述

3.2.1 形式的記述

入出力装置には色々な目的・形式のものがあり，機能，性能，形状，物理的特性などの特徴も様々に異なっている。合理的なマン・マシン・システムを設計するためには，直感のみに頼らず，トップ・ダウン的に，まず，機能面に着目して目的とするシステムに必要な入出力装置が備えるべき性質を規定し，それから様々な条件を加味して設計を段階的に進めて行くようなアプローチが必要となる。

このためには実在のハードウェアを抽象化し，その本質的な特徴に基づいて記述を作成しておき，これらの記述を手掛りにして，必要な機能を満たす装置の内容を段階的に確定して行くようなアプローチが必要である。このような考え方に基づいて，特に機能面から入出力装置を記述しようとする方法に「ロジカルＩ／Ｏ」という考え方がある。

ロジカルＩ／Ｏのような記述に基づいて，設計途上で必要な機能をトップ・ダウンで段々と詳細に記述し詰めて行くと，ある場合には現在の処，未開発な入出力装置に到達することもあろう。あるいは，現在１つの装置の機能を部分的に組み合わせることによって，複合された機能を有する新しい入出力装置として規定できる場合もあるかも知れない。このような考え方を更に押し進めると，好ましい標準的機能を持つ仮想入出力装置（バーチャルＩ／Ｏデバイス）を規定してくることも可能であろう。

このような試みは，まだ十分に固まってはいないが，ここでは１つの試みとして，入出力装置中の，特に画像入出力装置を中心にして形式的記述を行った。その一例を表３－１に示す。表中，<>で囲んだ属性は右辺に現われる場合には変数として更に細かなレベルの記述を持ち，右辺で<>付でないものは最終的な属性になる。｜は「または」を表わす。

表３－１　入出力装置の形式的記述の一例　（つづく）

入出力装置 :: ＝<情報形式の変換><ハードウェア記述>
情報形式の変換 :: ＝<入力・変換方式><内部表現><入力機能性能>
入力・変換方式 :: ＝音声<音声認識方式>｜画像<画像入力方式>｜
図形<図形入力方式>｜位置<座標入力方式>｜
記号<記号入力方式>｜……
内部表現 :: ＝変数｜<行列>
行　　列 :: ＝１次元行列｜２次元行列｜３次元行列｜４次元行列
入力機能性能 :: ＝<入力機能><入力性能>
ハードウェア記述 :: ＝<物理的諸元><経済性><使用性><サービス>
物理的諸元 :: ＝形状寸法・重量・所要電源・設置環境条件
経　済　性 :: ＝設置コスト・運用コスト・消耗品
使　用　性 :: ＝操作法・取扱い訓練の必要性・操作性
サ　ー　ビ　ス :: ＝設置時のサポート・メンテナンス
入力機能 :: ＝<媒体><観測位置設定方式><光源><色><走査形式>
媒　　体 :: ＝自然界｜フィルム｜紙｜布｜……
観測位置設定方式 :: ＝入力装置移動方式｜<媒体装着方式>
媒体装着方式 :: ＝平面型｜円筒型｜……
光　　源 :: ＝受光方式｜反射方式｜透過方式
色　　:: ＝色彩画像｜濃淡画像
走査形式 :: ＝ラスタスキャン｜ランダムスキャン｜ヘリカルスキャン｜……

表３－１　入出力装置の形式的記述の一例　（つづき）

入力性能 :: =<分解能>・入力位置精度・入力速度
分 解 能 :: =空間分解能・濃度分解能

3.2.2　分類的表現

音声，画像，図形などのコンピュータ内部での利用の仕方と情報の表現については，特にマン・マシン・コミュニケーションとの関連から最近色々と考察されている。

ここでは１つの試みとして，人間が受けとるメディアとしての表現とコンピュータに入力された時の最初の段階での情報の表現構造の２つを座標軸として，各種の情報形態と入力装置の位置づけを試みる。図３－１は，これらの位置づけを表わしたものである。

横軸のメディアは，音，映像，文字，位置に大括りされ，縦軸の入力直後の表現構造はスカラと行列（１～４次元）に分れている。この平面上に名称で表わしたものは通常用いられている情報あるいは処理技術の種類であり，長方形で囲んだものは入力装置名である。

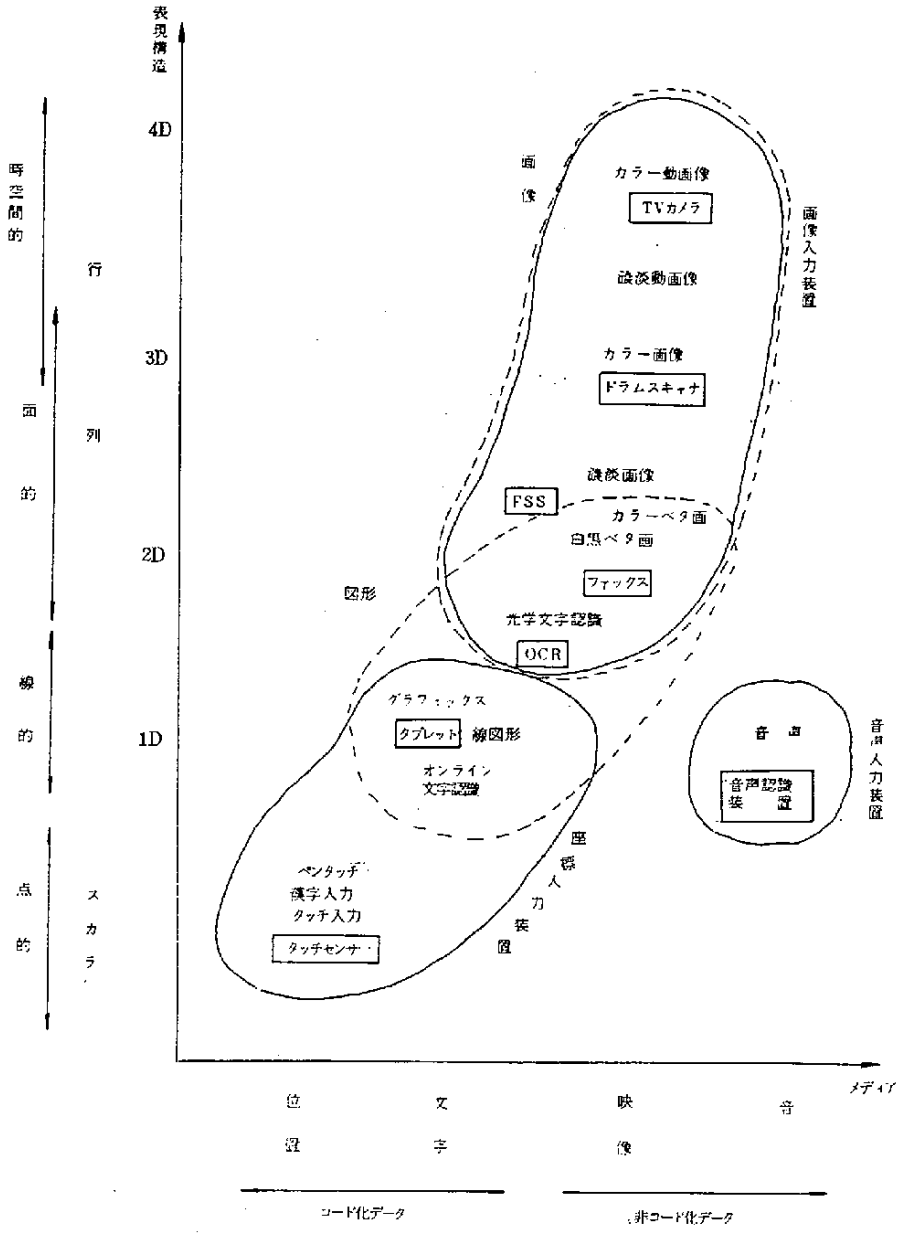


図３－１　入力装置の位置づけ

タブレットやタッチセンサ等は方式的に共通なので，ここでは座標入力装置として一括りにしてある。図３－１に見られるように，画像の入力は画像入力装置とよく対応している。音声入力装置は独立したものとして分類される。

図形と呼ばれるカテゴリの情報としては，線図形の他，ベタ画を入れることが多い。ベタ画は閉領域の集合で，領域の区別のために特定の閉領域だけが黒く塗られたり，あるいは特定の色で塗られたりする。ベタ画のような図形を映像として把えた場合には，3.1節で述べたように画像入力装置によってコンピュータに入力するが，領域の輪郭をタブレットのペンでなぞって入力し，この輪郭情報をコンピュータ内部で処理して図形情報とする事も可能である。

このように，図形として認められる情報に対する入力装置は必ずしも１つの形式ではない。このため「図形入力装置」という呼称は，タブレットやライトペンの如き座標入力装置で入力された座標情報をコンピュータ内部で処理して形状に対応づける場合にのみある程度通念となっているが，それ以外に関しては議論が曖昧になるのはこの辺に原因があると考えられる。

他の分類法としてコード化，非コード化の分類がある。これは，当初，コード化データである文字や数値のみを処理対象としていたコンピュータの用途を拡大して，画像データとか音声データを，そのままの形で蓄積したり処理したりする概念を導入するために，ノンコーデッド・インホメーション（ここでは非コード化と呼ぶ）という用語で説明されたものであるが，他方では「コンピュータで扱う情報は全てコード化されたものでなければならない筈だ」という議論もある。*

コード化，非コード化についてのこれらの関係は，図3－1では，横軸のメディアについては映像と音を非コード化データとし，それ以外をコード化データとする区別を行ない，縦軸のコンピュータ内部での表現構造については，全てコード化データとして取扱うことで説明した。

図３－１に示した各種の入力装置は，メディアを変換してコンピュータに取込む際にどのような表現構造として取扱うかを基準に分類したものであった。しかし，マン・マシン・システムにおける入力装置からの情報は，単にコンピュータに取込まれるだけでなく，更に各種の処理解析が施されて，メディアが持っている意味を抽出した上で，より高度な入力情報として利用される場合もある。このような観点から考えると，入力情報に対し，図３－２に示す様な処理の深さの階層を考えることができる。このような階層を考えることによって，各種の入力装置の位置づけと役割をより明確に把握することができる。

画像について云えば，蓄積，加工，認識，理解のような段階を経るに従って，入力情報は，より記述的になり，言語での表現に近づき高度化されてくる。これと平行して，音声に対しても，通信，処理，蓄積，認識，カナ漢字変換，理解といった段階を経るに従って，入力情報は段々と高

---

*たとえば，白黒の写真を十分に細かくディジタイズして２次元行列データとしてコンピュータに入力する例を考えると，この行列全体が表わしている画像を考える場合には非コードという考え方は妥当であるように思えるが，個々の画素に着目し，ライン間相関を利用して画像データの圧縮処理をするとすれば，個々の画素データはコード化されていると見なす方が妥当であろう。コード，非コードの区別は必ずしも明確ではない。

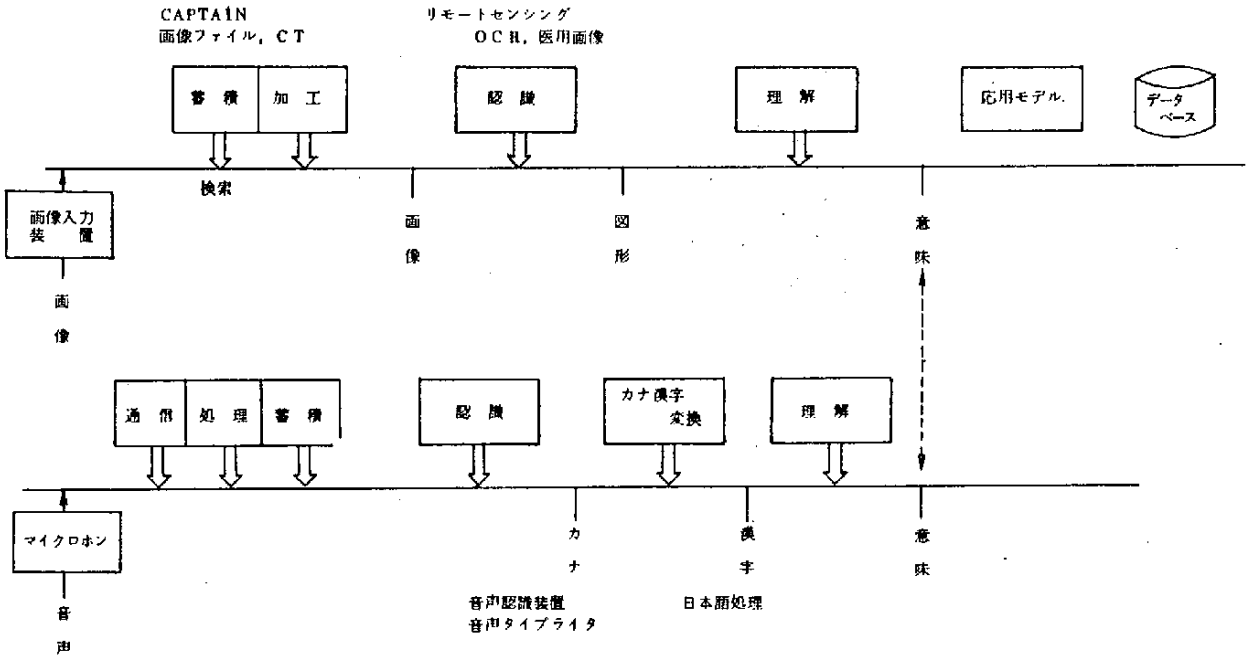


図３－２　入力情報に対する処理階層と装置

度化され記述的になる。画像系における理解と音声系における理解とは対応し，いずれの場合にも意味のレベルの情報抽出が期待される。

これらの高度な処理解析機能は，入力装置とは独立にコンピュータ内部で行なわれるシステムも多いが，ＯＣＲや音声認識装置などの場合には，認識処理までも含んで１つのまとまった装置になっており，高度なインテリジェンスを持った入力装置として位置づけることができる。

また，入力装置のあるものは，それ単位で用いられるが，特にマン・マシン・システムでは，出力装置に得られるシステムからの指示や応答を手掛りにして，人間が入力装置から情報を入力して行くようなインタラクティブな利用法も少なくなく，出力装置との関連において入力装置を考察する必要がある場合も少なくない。

3.3　入力装置の現状

3.3.1　音声入力装置

音声は人間にとって最も自然な入力手段であり，使用が容易である。既存入力手段に比べ両手が解放される。電話等既存の通信手段から利用できるなどの特長を有する。

音声認識方式は表３－２に示すように分類できる。*No.*1 タイプは技術的に最も容易で既に数年前に米国で開発され製品化されており，第１世代とも云えよう。用途は限定される。*No.*2 タイプは３年程前に日本で製品化された。これは第２世代と云うべきもので，入力速度が向上するため，特定話者向認識装置の応用分野が拡大するものと期待される。続いて第３世代として実用化されたのは*No.*3 タイプであり，その後*No.*7 タイプが第４世代として実用化される可能性が高い。それ以外の*No.*5 タイプは大学などである程度研究は行なわれているが，今のところ認識性能に問題があり，*No.*1 タイプに対抗して実用化するのはあまり容易ではない。*No.*4 タイプは技術的に難しく開発の見通しが立っていない。*No.*6 及び*No.*8 タイプは音声理解システムの一部として研究が進められており，特に*No.*8 タイプは，電総研において長期的な研究プロジェクトが行なわれている。

音声認識方式は，認識語数の限定，単語セットの限定，発話方法の限定，話者の限定などの制約をうまく利用することによって，最小の処理で最大の認識性能を実現する努力が続けられている。各種の認識方式とその関連は図３－３に示すようになっている。

単語（単音節）単位の認識方式では，認識対象の数を制限することにより，パターン・マッチング方式や識別関数方式のような比較的単純で認識性能の高い方式を利用できる。

パターン・マッチング方式で特定話者を対象とする場合は，本人の単語音声パターンを登録しておき，それと入力音声とのマッチングをとる方式により高精度の認識が行なわれる。この場合，音声の個人差の影響は無視できるが，発声速度による時間軸上の変動に関しては何らかの時間正規化が必要である。単純パターン・マッチング方式では，単語全体を一様に伸縮させて一定値に

表３－２　単語音声認識方式の分類[1)]

No.	タイプ (A) 1: 離散発声 2: 連続発声	タイプ (B) 1: 特定話者 2: 不特定話者	タイプ (C) 1: 単語単位 2: 音声単位	方式の特徴	適用分野	実用化状況	代表例	世代*1
1	1	1	1	技術的に最も容易	仕分装置制御，検査データ入力など	製品化	Threshold Scope	第1
2	2	1	1	高速入力が可能	同上	同上	日電	第2
3	1	2	1	電話入力に適す	予約，問合せなど	研究開発中	日電(大プロ)	第3
4	2	2	1	(技術的に困難)	同上	——	——	
5	1	1	2	メモリ，処理量が小	No. 1 に同じ	研究開発中	東大 など	
6	2	1	2	(SUS*2 に包含される)		——		
7	1	2	2	語彙の変更，拡大が容易	No. 3 に同じ	研究開発中	東北大	第4
8	2	2	2	(No. 6 に同じ)		——		

*1 単に実用化の順序（予想も含む）を示す．
*2 Speech Understanding System

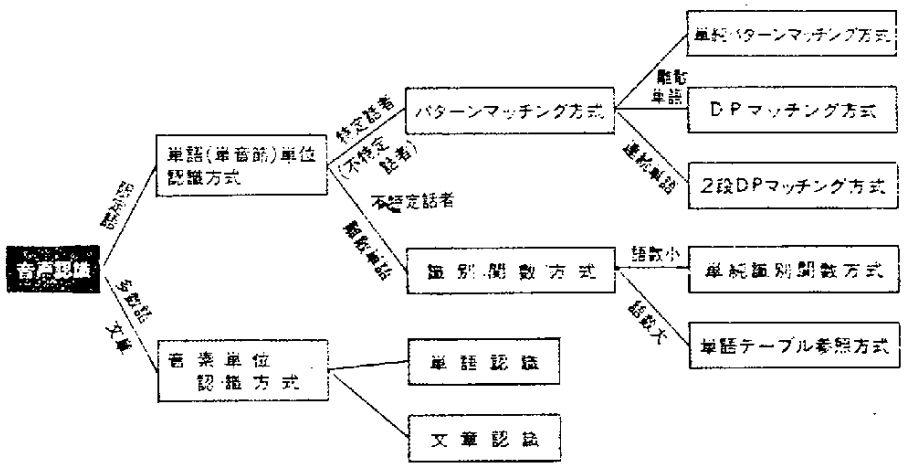


図３－３　音声認識方式の分類と関連図[2)]

揃える線形時間正規化や，音声パターンの変化の多い部分を相対的に伸ばすような非線形時間正規化等が行なわれる。これらはいずれも近似的な正規化法であるが，ＤＰマッチング方式では，動的計画法の手法を利用して，最適な時間正規化がマッチングと同時に行なわれるため高い認識性能が実現されている。２段ＤＰマッチング方式では，単語レベルでのマッチングと単語系列と

してのマッチングの2段階にDPを用いており，これによって高精度の連続単語認識が実現された。パターン・マッチング方式は，標準パターンを複数個用いるなどの拡張により不特定話者用にもある程度用いることができる。

不特定話者の音声パターンには個人差による大きい変動があり，単語音声を特徴ベクトル空間上の点として表わすとカテゴリ毎にある分布をなして拡がっている。識別関数方式は，このようなカテゴリの分布間の境界を識別関数として求め，これに基づいて認識を行なう方式である。これには単純識別関数方式とこれをベースにして，より大語彙の場合にも高い認識率が得られるように改良・強化された単語テーブル参照方式がある。

認識語数が数百以上になると，単語単位の認識方式では処理量の負担からほぼ限界となる。この場合，単語を構成する個々の音素を分離して認識し，その組み合わせで単語認識を行なう音素単位の認識方式をとれば，音素の数はたかだか数十種類であるから単語数の増大には容易に対処できる。この方法は，さらに文音声の認識にも拡張することができる。しかし，現状では基本となる音素の分離と認識の技術が困難であり，認識性能を向上させるには高度な技術を要する。

3.3.2 画像入力装置

(1) フライング・スポット・スキャナ

光源としてCRT*の輝点をレンズ系を通して集光して対象物に当て，反射光または透過光を光電変換する。CRT管面上の位置はCRTに加える電気信号によって制御することが可能であり，テレビのようなラスタ走査またはコンピュータなどで位置指定することによってランダム走査をすることができる。走査を暗室内で行なう必要があり，装置が多少大がかりになる欠点がある。35～70㎜のフィルム画像など比較的小さな媒体での画像入力に用いられる。

(2) TVカメラ画像

TV**カメラから得られるビデオ信号をサンプリングし，AD***変換してコンピュータへの画像入力に用いるものである。ビジコン管を用いた工業用TVカメラ（ITV）が安価で手軽であり，広く利用されている。TVカメラは，光学系を適当に選ぶことによって風景とか物体などの3次元対象，紙などの媒体上の画像，背後から一様な光源で照射したフィルムなど多様な画像を対象として取扱える利点がある。

標準TV方式のカメラを用いる場合には，1画面分の画像バッファ・メモリを介してフレーム単位で画像をコンピュータに入力する方法が便利である。

カラーTVカメラを用いたカラー画像入力の他，静止対象物に対しては簡便法として，モノクロのTVカメラの前に，赤，緑，青の3原色分解用のフィルタを設け，これを順次切りかえてカラー画像入力を行なうことができる。フィルタ切替方式は，入力に若干時間がかかる欠点がある。

---

* CRT…… Cathod Ray Tube

** TV …… Television

*** AD …… Analog to Digital

(3) ドラム・スキャナ

原理は図３－４に示す通りで，回転ドラム上に画像を記録した媒体を巻きつけ，ドラムの回転によって主走査を行なうと共に，これに同期して光学ヘッドを機械的に送ることによって副走査を行なう。フィルムを読取る透過型と紙面から読取る反射型とがある。透過型では，回転ドラムの回転軸を片持ちにすると共に，回転ドラムの一部をフィルム寸法に見合った適当な大きさの窓型に切り抜き，ドラムの内外に光源と光電変換器を配置する。濃淡画像用の両方の機器が開発され市販されている。

ドラム・スキャナは，機械的に順次走査をするため，比較的低速ではあるが位置精度が優れており，かつ，大型図面の読取りにも適している。

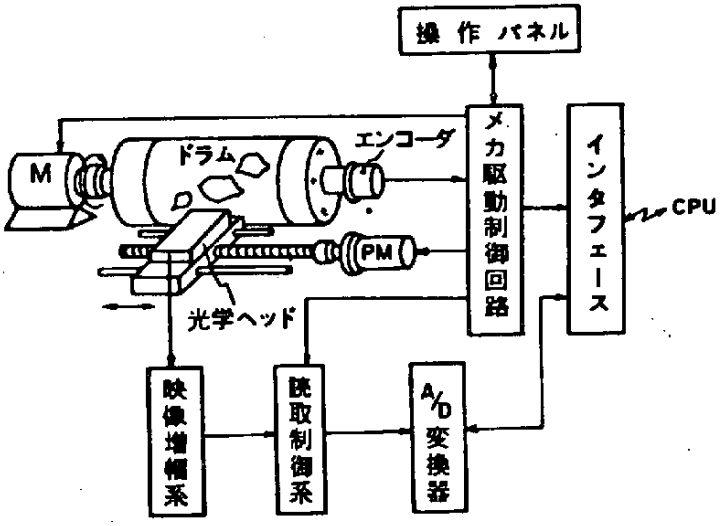


図３－４　ドラム・スキャナの原理図

(4) ファクシミリ送信機

ドラム・スキャナと全く同じ原理で画像を電気信号に変換するもので，最近では大量に製造されているため機器の価格も比較的安くかつ手軽に利用できる。

最近では，ファクシミリ用のコンピュータ・インタフェイスを開発し，ディジタル画像信号に変換してコンピュータへの画像入力装置として利用することが行なわれている。本来の画像通信機能に加えて，コンピュータに接続され画像入力機器としての利用の道が開かれたため，画像通信と画像処理とを組み合わせた新しいディジタル・システムとして，将来のオフィス・オートメーション用にも重要な機器になるものと期待されている。

現在のところ，ファクシミリの標準機器を利用しているため，取扱い対象は白黒画像で画面寸法も比較的小さく反射型のみである。原理的には，新聞ファクシミリなど透過型のものをコンピュータへの画像入力用に利用することも可能である。

(5) ＣＣＤスキャナ

ＣＣＤ*を撮像デバイスとして用いる固体画像入力装置あるいは素子そのものを指す。光学像をＣＣＤデバイス上に結像させ，電子的に走査して電気信号を取り出す。ＣＣＤには１次元アレーと２次元アレーとがある。１次元アレーの場合には，アレーの長手方向の走査はＣＣＤ内部で行ない，これと直角方向の走査はＣＣＤデバイスと対象画面とを相対的に移動させることによって実現する。通常は用紙などの媒体をＣＣＤスキャナと直角に走らせる。１次元ＣＣＤアレーは，ファクシミリやＯＣＲ**のスキャナとしてよく用いられている。２次元ＣＣＤスキャナは，主としてＴＶカメラ用に利用されている。

(6) レーザ・スキャナ

レーザ・スキャナは，レーザ光を偏向させて対象物体に当てて走査し，その反射光を光電変換するスキャナである。レーザ光を偏向させる走査方式としては，回転多面鏡と走査レンズを組み合わせた方式と，ホログラムを回転させる方式とが代表的である。後者は，特にホログラム・スキャナと呼ばれることもあり，ホログラムがレンズとスキャナとの機能を兼ねてレーザ光を偏向させる。簡単な光学系で複雑な走査パターンを実現できることと，読み取り深度が深く取れることが特徴であり，現在，主としてバーコードを用いるＰＯＳ***スキャナとして実用化されている。

(7) 画像入力装置の方式比較

(1)から(6)に述べた各種の画像入力装置について，走査方式と発光方式による分類を表３－３に示し，分解能，走査方式，その他の特徴をまとめて比較したものを表３－４に示す。

3.3.3 座標入力装置

座標入力装置の代表的なものとして，タブレット，ディジタイザ，ライトペン，タッチセンサなどがある。

これらの装置の機能は，基本的には，座標入力面上をペン状のスタイラスやカーソルや指先などの点指示手段で指示すると，その点の座標が検出されて電気信号として出力されるものである。座標検出のサンプリング時間間隔が短いタブレットのような入力装置ではスタイラスで線図形や文字を描くと，それらの軌跡を線図形としてコンピュータに入力し処理することが出来るので，「図形入力装置」と呼ぶことも一般に行なわれている。また，ライトペンのように，その構成上表示装置であるディスプレイと不可分なものでは，出力される映像と組み合わせて特色のある入力機能を果させる使い方もある。

座標入力装置は表３－５に示すように，ディジタル方式においては，入力面の量子化方法，点指示手段と入力面との間の位置信号の結合方法によって，また，アナログ方式においては，位置－アナログ量変換方法などによって，いくつかの方法に分類することができる。

＊ ＣＣＤ…… Charge Coupled Device

＊＊ ＯＣＲ…… Optical Character Reader

＊＊＊ＰＯＳ…… Point of Sales

座標入力装置の利用法には，１点ずつ指定してその座標値を入力する位置指定の用法と線図形や文字を手書きしてサンプリングされた座標列をコンピュータに入力する描画的用法とがある。個々の装置ごとに，どちらの用法に，より適しているかは差があり，各装置の特徴の１つをなしている。

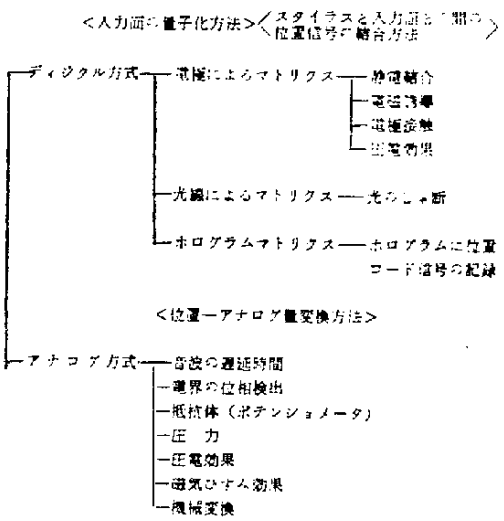
表３－３　走査と発光方式による画像入力装置の分類

発光 / 走査	発光方式	非発光方式
機械的走査	ドラム・スキャナ，ファクシミリ，レーザ・スキャナ	
電子・機械組み合わせ		ＣＣＤ（１次元）
電子的走査	フライング・スポット・スキャナ	ＣＣＤ（２次元） ＴＶカメラ（ビジコン管）

表３－４　画像入力装置の比較

入力方式	分解能	走査方式	その他
フライング スポットスキャナ	1000 × 1000 ～ 4000 × 4000 絵素	順次走査又は ランダム走査 電子走査	暗室内で走査 比較的小型のフィルム画像等が主。
ＴＶカメラ式	480 × 630 絵素	順次走査 電子走査	簡易形スキャナとしてよく使われている。 残像が多い。
ドラムスキャナ	反射型： 10～20絵素／㎜ 透過型： 40～80絵素／㎜	順次走査 主・副走査とも機械走査	低速であるが大型図面の入力が可。
ＣＣＤスキャナ	一次元： 2048絵素／ライン 二次元： 通常ＴＶ画像程度	順次走査 電子走査 但し，一次元の場合副走査は機械走査	半導体スキャナ
レーザスキャナ	2000～4000 絵素／ライン	順次走査 回転ミラーによる走査	走査装置と撮像対象との距離変動に対し分解能低下少ない。

表３－５　座標入力装置の方式分類



(1)　タブレット

手書き図形を入力する装置として最も一般的な装置と云えよう。25 × 25㎠程度の入力面を持ち，1㎜当り３～10本程度の分解能を持つものが多い。入力した図形のモニタリングのため，ディスプレイと組み合わせて使用することが多い。スタイラスと呼ぶペンの先端を座標入力面に接触／接近させると，そのｘ，ｙ座標が検出されて座標データが出力される。

座標データの他に，スタイラスの先端を画面に押しつけているか離しているかを検出する機能を備えているのが普通である。これは点列のうち図形を構成する線の部分とそれ以外の部分の区別や，特定の入力指示をコンピュータに伝える目的で利用される。装置によっては，この機能を更に拡張し，スタイラスの先が画面に接近して一定の距離（約1㎝）より近づくと検出信号が出るような近接検知機構を備えたものもある。

タブレットでは，画面上に紙を置いて，その上からボールペンを組み込んだスタイラスで図形や文字を描くことにより，人間には普通の紙にボールペンで書くのと同じ動作をさせながら，座標を電気的に検出してコンピュータに入力できるハードコピー付きのものも多い。タブレットは位置指定の用法もできるが，描画的用法にも適している点が特長である。

タブレットの位置指定の用法を積極的に利用した入力装置として，ペンタッチ式漢字入力装置がある。基本的には，タブレット画面上に漢字一覧表をのせて固定しておき，入力したい漢字を人間が選び出してスタイラスで指示すると，スタイラスの座標が検出されてこの座標値からどの漢字を入力したのかが判別されるものである。必要な程度に座標の分解能をおとし，漢字コードに変換して出力できる機能を備えた専用の漢字入力装置が色々と試作され市販されている。

(2)　ディジタイザ

ディジタイザは，紙などに描かれた図面から，要所々々の点の座標をディジタイズしてコンピュータに入力するための装置であり，タブレットとよく似ている。タブレットと比較すると，一般

に大型図面を高精度に入力するのに適している。点指示のためには，カーソルが用いられ，メカニカル機構を用いたアーム型と，機械的制約なしにカーソルを自由に移動できるフリーカーソル型とがある。

タブレットとディジタイザとを厳密に区別することは難しい。両者の性格を組み合せたタブレット・ディジタイザなどの市販品もある。

(3) ライトペン

ライトペンは，ディスプレイ装置と一体化した座標入力装置であり，基本的には，ディスプレイの輝点をスタイラスに組み込んだ受光素子によって検知したタイミングを利用している。このタイミングによって，ディスプレイ表示の位置制御機構では輝点の位置を知ることができるが，この位置座標がライトペンの指している座標情報になる。位置制御機構はハードウェアとしてディスプレイ内部に組み込まれていることもあり，あるいはコンピュータのソフトウェアが直接制御している場合もあるが，いずれの場合にも，座標入力装置として利用することが可能である。

ライトペンは色々な特徴を持った使い方ができるが，ディスプレイ管面とライトペンとの間に紙などの不透明なものを置いてハードコピーを取ったりすることができないのは不便なことが多い。

(4) タッチセンサ

タッチ・センシティブ・ディジタイザとも呼ばれる。タブレットのスタイラスとして器具を用いず，人間が指先で直接指示すると，その座標値が検出できるようにしたものである。方式的には，タブレットなどのうちスタイラスを受動的な物体にした方式がそのまま利用できる。

実際の例としては，座標入力面としてガラス板を用い，ガラスの縦横2辺にセラミック変換器からパルスを加えてガラス表面に表面弾性波を送出する。ガラス表面上を押している指先などがあると，表面弾性波は反射されて変換器で受信され，反射パルスの時間が測定されて座標が計算される。変換器は送・受信が時分割で用いられる。

他の方式としては，発光ダイオードと受光トランジスタを組み合わせて座標入力面の1辺から対向辺に向って入力面に沿った平行光線ビームを走らせておき，このビームが指先などで遮られたのを利用して指でタッチした場所の座標を検出するものがある。

タッチセンサは，指先を用いるため位置の指定精度は高くはないが，マン・マシン・コミュニケーションの入力装置としては，今後興味深い利用法が色々と出てくるものと予想される。

3.3.4 応用システム

音声入力装置を中心とした応用としては，物流関係の配送センタでの荷物の仕分け，小包などの仕分けがある。電電公社が開発中の「ボイスQ－Aシステム」は，電話を通じての音声入力－音声応答による座席予約システムである。また，西独では音声で地名を入力し，これによって検索した地図をディスプレイに表示する地理情報システムが開発されている。

画像処理関係では，医用画像，リモートセンシングにおける衛星画像，航空写真，気象衛星画

像，産業応用画像などを対象として多くの画像入力，画像処理が行なわれている。

画像処理技術を応用して，主としてオフィスにおける文書処理を支援するためのシステム開発の試みも盛んになってきており，文章情報と図面情報を別々の映像としてコンピュータのファイルに蓄積しておき，必要に応じて適当な情報を選択してきて，それらを都合よく配置し，合成してハードコピーを作り出す図形処理はいくつか具体例が出てきている。

3.4 現状の評価と将来への展望

3.4.1 音声入力

前に図２－２に示した処理の深さの階層で考えると，音声入力装置は認識レベルのものが実用に供されている。現在，製品の主流になっている単語単位認識方式では，同音異義の処理を加えれば漢字での表現は問題はないが，単語数の制約が問題である。これに対し音素単位認識方式は実用化研究が進められており，いずれ本格的な実用化が行なわれ，カナ漢字変換と組み合わせて日本語文章入力の強力な手段になろう。更に進んだ段階では，当然，文章理解に向って研究が進められることになろう。

使い易さの面からは，連続発声・不特定話者・音素単位認識が望まれる。他方，製品化されてきた音声入力装置に対し新らたな応用が今後色々と試みられるであろうが，マン・マシン・システムに大量に導入されるためには手軽さと経済性の面での一層の進歩が必要であろう。手軽さの面では，小型化，軽量化，装置またはコンポネントとしての取扱い易さにおける改良が必要である。価格面でも，現状から１桁以上の大幅な価格低下が必要である。これらの問題解決のためには，音声合成装置がＬＳＩチップ化されたように，音声認識装置についてもＬＳＩ化することが重要であろう。

マン・マシン・システムの入力手段として音声のみで完結しようとすると無理があり，画像，図形など他のメディアと組み合わせた総合的な入力手段の開発へ向うのが１つの有効な方向であろう。

3.4.2 画像入力

画像入力装置はいくつか実用化されているが，現状ではコンピュータでの画像の利用は限られた分野や研究用でしか行なわれていない。この原因としては，画像のデータ量と処理量が膨大なため，これ迄のところコンピュータの能力が不足で一般的な処理対象になり得なかった事があげられる。しかし，コンピュータの記憶容量の増大と処理速度の向上に伴ない，あるいは専用画像処理装置の開発などに伴なって，画像が急速にディジタル情報処理の対象としてクローズアップされてこよう。

このような情勢変化に伴ない，従来，特殊な分野でしか利用されなかった画像入力装置の面でも，急速な進歩発展が要求されることになろう。ここでも，小型化，軽量化，取扱いの容易さ，

低価格化は重要な改善項目である。

特にオフィス用に，現在のコピー機器程度の手軽さで，文書をディジタイズしてコンピュータのファイルに入力できるスキャナの開発が望まれる。この様なスキャナでは，ファイルの無駄な増大を防止するために，紙面の一部分のみを選択的に入力する機能が不可欠である。

入力の際の分解能は，目的に応じて選択できることが望ましいが，処理や再生を考慮すると分解能を始め各種の入力規格をきめておくことが望ましく，規格化の検討は早期に取組むべき１つの課題であろう。

画像データの蓄積・加工は，最近いくつかの実用例が出てきたが，インテリジェント・コピアでの動き等にも見られるように，今後のオフィス・オートメーションにおいて重要な役割りを果すものと考えられる。

ファイルの性能は今後どんどん向上して行くと考えられるが，これとは別に，入力された画像データ量を有効に圧縮することは常に考慮すべきことであり，従来のデータ圧縮技術と共に特に認識処理技術が重要である。

この面では，認識対象は限られるが，既に文字認識の分野で，手書き英数字・カナＯＣＲが実用化されて大きな成果をあげている。今後，漢字ＯＣＲの実用化に向い，日本語入力が本格的に行なわれるようになると，次のステップの日本語理解への取り組みも一層活発になるであろう。

文字認識以外の一般の画像の認識は，当面ロボットや医用画像などのかなり限定された分野で急速に発展する可能性があるが，何らかの程度に画像理解と結んだ形で高度化しながら進められるものと思われる。

3.4.3　図形入力

図形の入力は，3.3.2で見たように画像入力装置と座標入力装置によって行なわれている。また，図３－２にも示されている通り，画像を処理して意味の抽出に向う途上に図形が関係することがある。これらの事は，「図形」という用語が意味している実体について，更に深く考察してみる必要性を示唆しているように思える。

図形入力の問題には，特にコンピュータ内部での表現とか処理の問題が絡み合っているので，画像入力と同じような意味合いで図形入力を論じるのが適当かどうかについては一考を要する。

図形の取扱いに必要な情報を入力する座標入力装置については，3.3.3で述べたように，多くの装置が実用化されている。座標データは低次な情報であるため，これを制御のための位置情報として用いたり，線図形を構成する座標列として用いたりする意味づけは，コンピュータ内部での処理による。

処理を伴なう図形入力の例として，これまで研究段階に留ったオンライン実時間文字認識を発展させ実用化することは，今後，日本語入力との関連で重要になろう。これを更に発展させたオンライン図形認識は，特に，ＣＡＤ＊の入力機能を向上させる面から注目すべき技術である。

＊　ＣＡＤ……Computer　Aided Design

他方，映像としての図形処理では，各種の図面認識，図面理解が重要な問題となろう。特に，日常用いられている程度の図面を処理対象とする技術の開発は極めて重要である。

3.5 ま と め

音声，画像，図形などのメディアに対し，枠組としての記述を試み，現状，将来への展望の面から考察した。検討は，まだ取りかかりの域を脱せず，今後残された課題は少なくない。

形式的記述方法については1つの試みを行なったが，音声入力，座標入力装置等についての記述を追加して更に詳細にすること，このような記述を利用することによって，入出力装置の機能性能の整理を行ない，統一的な見方を確立して行くこと，今後どのような装置の研究開発に向うべきかの指針を得ることに迄結びつけて行くことが重要である。

現状調査では，応用技術の調査を組織的に進める迄に至らなかった。今後この調査を進めること，特に，この方面では，研究室レベルでの先進的な試みについて，我国と共に米国，欧州諸国の先端的研究を調べておく必要がある。

現在見えてきている技術が何時頃実用化されるかの予測については，例えばデルファイ法などの実施が有効であろう。これらの調査の場合，技術的可能性と共に，コスト面に重点を置いた予測が重要と思われる。認識・理解・モデル化などの情報の抽象化・高度化のための研究・技術開発は重要な課題であるが，これと共に，実際的な利用面ではもっと低レベルの処理のみによっても進歩発展する分野は少なくない。このような観点からは，利用分野の大きさの予測も加味した研究課題の明確化も重要であろう。

新しいメディアの利用分野では，技術の開発・発展が急テンポであり，多大の資源が投入されているが，この分野の効率的な発展のためには，様々な努力の有機的な協力関係の確立，その為の段階的な規格化の推進などは重要かつ緊急であり，これらへの取り組みも含めて検討しておく必要性があろう。

参　考　文　献

1) 千　葉：単語音声の認識，情報処理，Vol. 19, №7, pp. 667（1978）

2) 千　葉：音声入力の動向について，事務と経営，pp. 46（1979—11）

3) 坂　井：情報システムにおける音声の認識と合成，情報処理，Vol. 21, №8, pp. 820（1980）

4) 阪　口：図形入力装置，信学会誌，Vol. 60, №6, pp. 645（1977）

5) 首　藤：図形処理システムにおける図形，画像の入力方法，技研情報センターセミナー資料（1980）

6) 河　田：日本語のワード・プロセッシング，情報処理，Vol. 21, №8, pp. 894（1980）

7) （財）日本情報処理開発協会：海外における周辺装置の動向調査報告書（1977）

8) 小　林：ファクシミリ通信の動向，信学誌，Vol. 61, №12, pp. 1347（1978）

9) 山　崎：ファクシミリ国際標準化の動向，テレビジョン学会誌，Vol. 33, №9, pp. 695（1979）

10) 中・高速の標準機が登場し白熱化するファクシミリ販売競争，日経エレクトロニクス，1979年2月5日号，pp. 162

11) 南　部：躍進するファクシミリ通信の展望，データ通信，1980年3月号，pp. 31

12) 第3世代のファクシミリ，データ通信，1980年3月号，pp. 35

13) 千　葉：音声認識の応用，信学会関西支部専門講習会講演論文集，pp. 80（1980）

14) 地名言うだけで画面に地図，日経産業新聞，1978年6月13日号

15) 電話で座席予約，合成音声で即答・声の内容を電算処理・電々公社，日経新聞，1979年12月31日号

16) 音声認識装置導入急ピッチ，日本工業新聞，1980年1月17日号

17) 音声入力装置・富士通が本格受注へ，産業新聞，1980年4月15日号

18) 日電が本格実用化・不特定話者用の音声認識装置，日本工業新聞，1980年5月30日号

19) 音声でデータ入力，産業新聞，1980年6月6日号

20) 音声入力で立体式，日刊工業新聞，1980年8月13日号

4. マン・マシン・ユーザ・インタフェイスの人間工学的考察

4.1 はじめに

当然のこととして，マン・マシン・ユーザ・インタフェイスは，人間の処理への介入を必要とする処理系において必要とされる機能であり，それがどのような機能を備えておくべきかは，処理系の中での人間の役割，および，人間の処理への介入の仕方によって決められるべきものである。

ただ，一般的に人間が処理系（機械）に介入する場合の人間と機械との接点は，人間から機械へ，逆に，機械から人間へ，情報を伝達するための入出力機能の部分であり，これを人間側からみると，入力機能は，視覚・聴覚・触覚といった感覚機能であり，機械への出力機能は，手足の運動機能とか，音声である。これに対する機械側の入出力機能は，当然，これらの人間側の諸機能とうまくインタフェイスの取れたものでなければならず，即ち人間側からみて，機械は，

- 使い易く
- 分り易く
- 抵抗感の無い

ものでなければならない。

ここで，いかに人間と機械とのインタフェイスを良くするか，換言すれば，親和性を良くするかということが問題になり，人間工学面からの探究が，必要となるわけである。 すでに実用になっている機械には，大なり小なり，この人間工学の知識が設計に，盛り込まれているが，まだまだ，入出力インタフェイスの人間工学の面的とも言える，感覚機能，運動機能との対応が主で，全人向の平均値への対応であると言える。しかし，機械側の機能が向上し，インテリジェンスが，増せば増すほど，マン・マシン・ユーザ・インタフェイスにも，より高度なものが要求され，単に，面的な入出力機能のみでなく，人間の持つ心理的，精神的な奥深いものと，より密に，親和する様な機能が，要求される。さらに極言すれば，人間のより知的なもの（頭脳，知識ベース）とリンクできるような，マン・マシン・ユーザ・インタフェイスが，要求されているのではないだろうか。最近，会話型処理，あるいは，会話型…という様なシステムが出廻っているが，それのマン・マシン間のコミュニケーションの中身は，人間一人間のコミュニケーションからみると，極めてたどたどしく不充分なものである。

一体それは何故であろうか？使用する言葉（言語）の問題であろうか？人間工学的な配慮で，何かが欠けているからであろうか？本章では，情報処理システムのインテリジェンスを向上させ，人間との親和性を，より密にする要素として，人間工学面からみた技術とは，いかなるものかについて検討する。そのために，まず人間工学とは何かについて，簡単に触れ，次にそれが，現在の機械には，どう生かされ，またどの様な問題点を持っているかを，アンケート結果から考察し，最後に，将来のマン・マシン・ユーザ・インタフェイスのあり方について考察する。

4.2　マン・マシン・ユーザ・インタフェイスの現状と人間工学的考察

4.2.1　人間工学とは

人間工学という言葉は，「人間工学概論」[1]によれば，HUMAN　ENGINEERING（アメリカで使われている）の訳である。人間工学は，元来，産業心理学の一分野であって，作業環境のうち，物的環境の方面を研究するものである。ヨーロッパでは，ERGONOMICSという言葉を用いるが，これは，ギリシャ語の，ERGON（仕事）＋NOMOS（管理又は法）＋ICS（学を意味する接尾語）からなるから，人間の仕事を，適正に管理する学問の意味である。換言すれば，仕事をするときに，エネルギーを合理的に配分して，人間として，正しい（無理のない，自然な）作業をする為には，どうしたらよいかを，研究する科学である。

いままでは，機械を，設計する場合には，とかく，機械本位に考え，人間の使い易さや，疲労などの人間の側を，二次的に，考えていた。そのために機械に適合する人間が，適性により，選択されて，操作に従事し，またそのために，訓練された。人間の性能を無視して，設計されている機械を，操作する場合には，機械に追いまわされて，疲労したり，トラブルを起こしたりした。いわば，機械に追われていたのである。これとは反対に，人間が，主となり，機械や器具を人間本位に，人間の使い易い様に作ろうというのが，人間工学であり，その特色は，総合科学であり，心理学と医学が中心となって，工学，その他の諸科学の協力により，研究が行なわれるものである。

ただし，人間工学は，あくまで，全人向の平均値への対応が主で，人間それぞれの持っている多様性を，カバーするものではない。したがって，マン・マシン・ユーザ・インタフェイスを考えていく上でも，どこまでの領域を，設定するかで，観点を変える必要も，でてくるであろう。

4.2.2　マン・マシン・ユーザ・インタフェイスの現状

我々が，機器のカタログを手にするとき，必ず謳われているのが，「この……には，人間工学的配慮がなされています」の一行であるが，その実態は，いかほどであろうか。詳しくは，ユーザの立場から述べる次節を参照していただくとして，ここでは，大きく把えることとする。

ハードウェア面から述べると，外部仕様に対する操作性の点では，各メーカでそれぞれ基準をつくっている。例を図4－1に表わす。その内容は，人間の作業域・基本姿勢・視認度・視空間等が，考えられているが，いずれも汎用化指向のためか，ユーザ各人が，ある程度，フレキシブルに，自分に合う様に変更できる様には，考えられていない。また，環境に対する考え方も受動的で，マシンも含めて，環境にもアドバイスしてより良い作業環境を具現させようという配慮はない。もっともこれについては，ユーザ側にも責任はあると考えられる。次に，内部仕様に対する知覚，聴覚の面であるが，視覚の面では，CRTなどの文字のちらつきがない様に，基本的配慮はなされているが，色調・輝度等については，選択できるという範囲だけに，とどめ，もう一つユーザ側に立っては，考えられていない。また，聴覚の面では，各ターミナルの出す音には，全

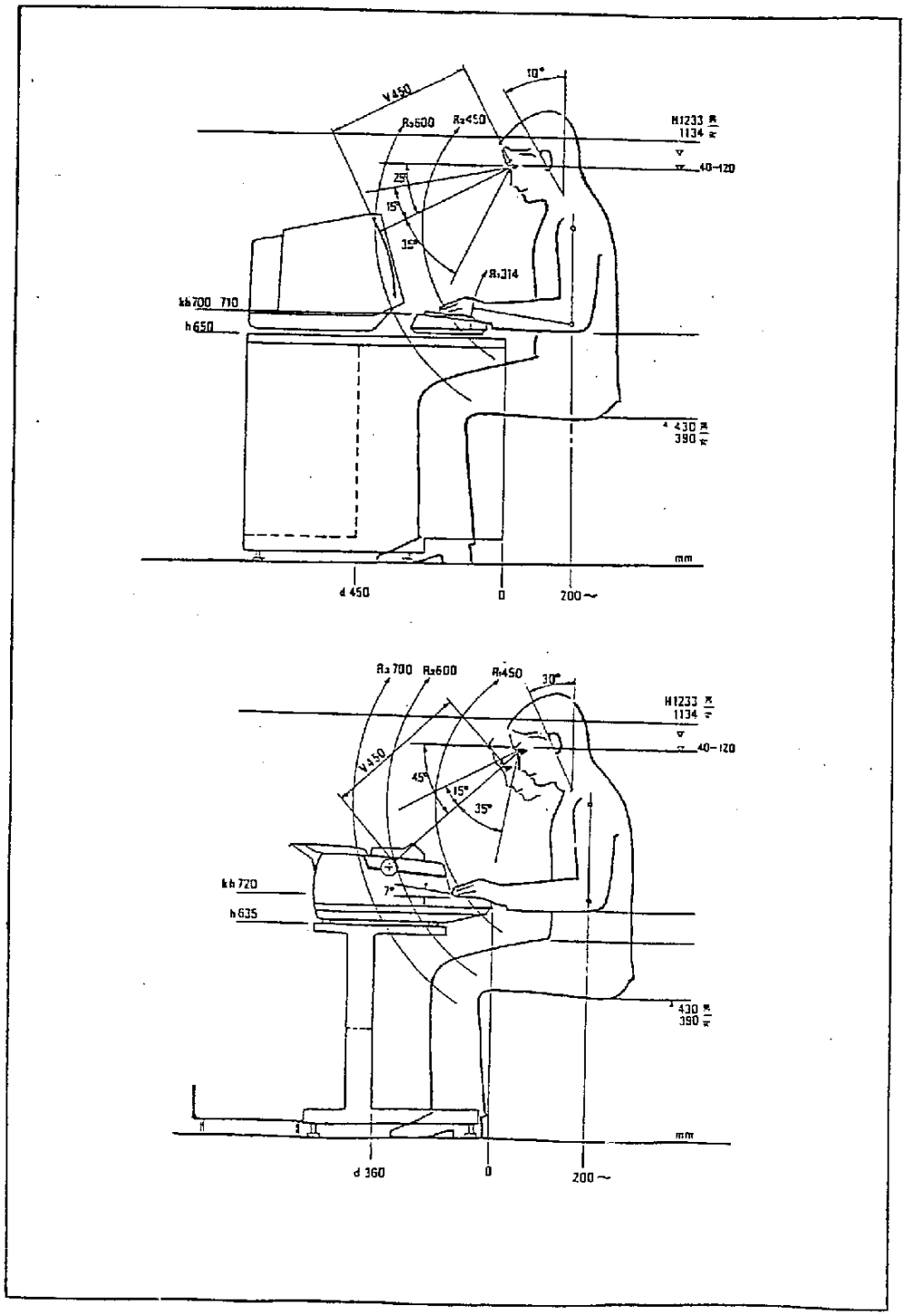


図４－１　外部仕様の基準の例

く無関心で，もっぱら，音声応答のことのみに，関心がいっている様である。以上は，大まかな，ハードウェア面であるが，ユーザにとって頭のいたい，メーカ別に，オペレーションが異なる点については，無関心のままである。

つぎに，ソフトウェア面であるが，総じていえることは，ユーザの立場では作られていなく，インタラクション面で，ユーザが，とまどうことが多い様である。特に，ユーザのアクセスに対する機械側の応答が不充分で，処理に対する不安感を生ぜしめ，イライラの原因となっていることが多い。この様な問題の解決は，ハードウェア面より，むしろソフトウェア面での工夫，即ち，マン・マシン・コミュニケーションの内容および方法に対する追求により，成されるべきであるが，まだまだ配慮が充分とはいえない。

4.3 ユーザの立場より見た問題点ならびに要求機能

これについては，個人的見解で，述べていくと，「木を見て，林を見ず」に，なりかねないので，約100人のユーザに，図4－2の様な，アンケート用紙を配り，その結果をもとに，述べていくこととしたい。ただし，ユーザの使用しているターミナルとしては，主として，ＣＲＴが，多くなっていることを，あらかじめおことわりしておかねばならない。また，アンケートに応じてくれた人達の経験については，図4－3に表わす様な，分布となっている。以下に，アンケートの項目に添って述べていくこととする。

マン・マシン・ユーザ・インタフェイスに関するアンケート

1. あなたのシステムに関する，ご経験をお書き下さい。
   あなたは下のどれにあたりますか。○印と経験年数をお書き下さい。

   1　マネージャー　　（　　）
   2　システムエンジニア　　（　　）
   3　プログラマ　　（　　）
   4　エンドユーザ　　（　　）
   5　その他〔　　〕（　　）

2. あなたが，端末を操作しているときに，心理的にイヤになったり，イライラすることがありますか。あるとすれば，それは，どんな場合ですか。

端末名	場合もしくは原因
〔　　〕	〔　　〕
〔　　〕	〔　　〕
〔　　〕	〔　　〕
〔　　〕	〔　　〕
〔　　〕	〔　　〕

3. あなたが端末を操作中，もしくは，操作後，疲労を感じますか。もしあるとすればそれはどんな場合ですか。

端末名	使用状況	疲労箇所
〔　　〕	〔　　〕	〔　　〕
〔　　〕	〔　　〕	〔　　〕
〔　　〕	〔　　〕	〔　　〕
〔　　〕	〔　　〕	〔　　〕
〔　　〕	〔　　〕	〔　　〕

4. あなたが端末を操作中，どんな点が気になりますか。該当するものに○印をつけて下さい。又，具体的指適は（ ）の中にお書き下さい。

   1）画面の文字の大きさ（小さい　大きい　その他（　　））
   2）　〃　　の輝度，色（　　）
   3）キーボードのキー配列　（　　）
   4）　〃　　のファクションキー（　　）
   5）画面全体の大きさ　（　　）
   6）画面の向き　（もっと斜めにした方がよい　その他（　　））

図4－2　アンケート用紙（つづく）

7) エラーメッセージの内容（　　）
8) モード変換の方法　（　　）
9) 端末の出す音　（　　）
10) ライトペンの操作　（　　）
11) 操作中の作業スペース（少ない　全然ない　その他（　　）
実情はどうしていますか：）

12) その他お気付の点：

5. 端末の置かれている環境で何か気になることがありますか。該当するものに○印をつけて下さい。
1) 照　明（暗すぎる　明るすぎる　まぶしい　その他（　　））
2) 空　調（暖冷房がききすぎる　きかない　その他（　　））
3) 騒　音（うるさい　耳ざわりである　大きい　その他（　　））
4−1) 作業台（高すぎる　低すぎる　せまい　その他（　　））
4−2) 椅　子（高い　低い　かたい　やわらかい　その他（　　））
5) その他お気付の点：

6. もし夢の端末が，あなたのお役に立つなら，それには，どんな機能が欲しいですか。

図4−2　アンケート用紙（つづき）

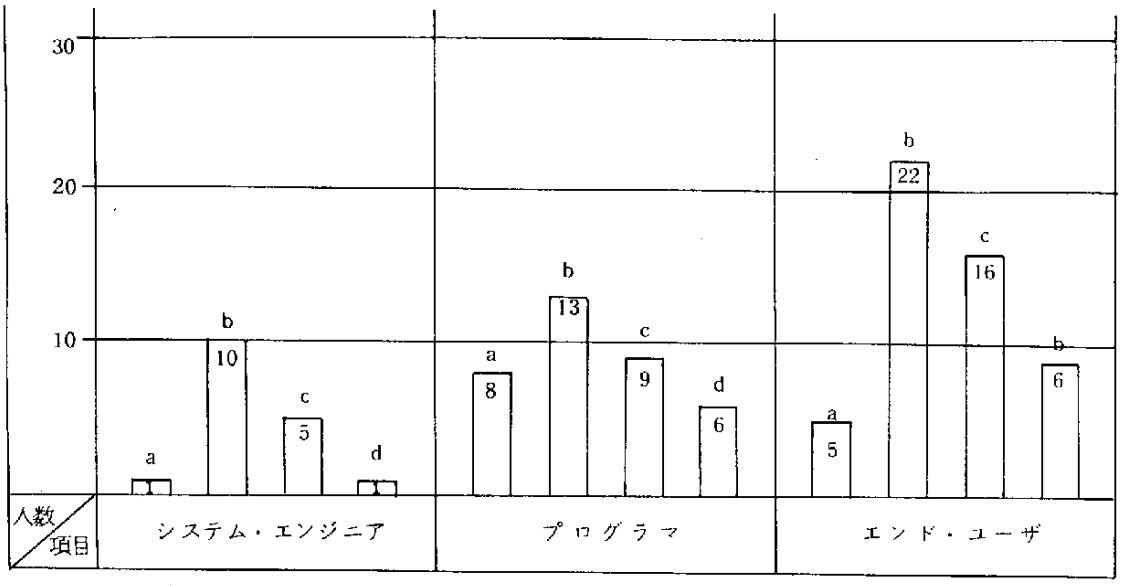


図4－3　回答者の分類

経験年数　a:2年以下　c:5～10年
b:2～5年　d:10年以上

4.3.1　心理的側面

ターミナルに向って，作業中に，心理的にイヤになったり，イライラしてくる原因を，探ってみると，表4－1に，表わすような結果となり，主として，レスポンス・タイムの改善，即ち，スピード・アップが，その原因を解消してくれそうであるが，これには，ハードウェア・ソフトウェアの両面の問題がありそうである。人間が，基本的に，どれ位なら待てるのかについては，個人差，慣れ（あるいは諦め）等があり，一概には言えないが，5秒ぐらいが，限界で，それを越す場合には，何等かの，インフォメーションがなければ，イライラしてくるものと，考えられる。これには，ハードウェア面の処理のスピード・アップ，アーキテクチャの改善と，ソフトウェア面のアルゴリズムの改善，さらには，人間とのインタラクションを改良するための，メッセージのやりとりなどが，問題として考えられる。

ついで，システム・ダウンのことが，あげられているが，これは，運営上の問題が，主として考えられるが，その他に，使う側の頻度，ボリューム，回線の混みぐあい等の，複雑な問題があるが，人間工学的側面としては，除外して考えたい。

表４－１　心理的イライラの原因　（百分率：数量／回答者数）

No.	項　　目	コ　メ　ン　ト	数　量	百分率
1	応答が遅い	10秒以上，ALLOC／LINKの際原因不明のとき	55	69.6
		LOGONよりの速い開始ができない	5	6.3
		ハード・コピーのスピードが遅い	4	5.1
		待ち時間が長い	4	5.1
		処理中の表示がない	4	5.1
		1－小計	72	91.1
2	システム・ダウン／マシン・ダウン		14	17.7
3	音が耳につく	ダイプ・ボール，ファン，ハード・コピー	5	6.3
4	モード変換の煩雑さ		3	3.8
5	画面への光の反射		3	3.8
6	エラーの際の再入力		1	1.2
7	カタカナのエディットの遅さ		1	1.2
8	ライト・ペンのピックの不正確さ		1	1.2
	特になし		2	2.5
	無回答		5	6.3

4.3.2 生理的側面

人間が，何等かの思考をする際の，手助けとして，ターミナルを使用する場合に，どこから疲労してくるかという点については，表4－2に表わす様に，圧倒的に，眼が疲れるようであるが，これは，単にハードウェア面のみにとどまらず，後の節で述べる環境面とも，大いに関係するものである。眼の疲れを，訴える人にも，個人差があり，10～15分の連続使用で，疲れのでる人から，1時間以上で，始めて疲れる人まで，様々であるが，画面の文字の読み辛さによるという点についても，文字自身の問題と，設置環境での照明の問題，その日の体調等，複合した原因に依るものと考えられる。人間の視認力についても，個人差があり，総てに応じることはできないと思うが，表示する文字，線等の輝度，サイズ等を，ユーザ側でより広く選択できるようにすることで，使用時間の経過に伴う，視認力の減退に応じられる様なことも，考えていく必要があろう。

ついで，肩が疲労するということであるが，この原因としては，不慣れ，緊張感，キーボードの問題，作業スペース等，いろいろ考えられるが，後述する操作性の分析から考えてみると，作業スペースの狭少なことからくる無理な姿勢と，キーボードのハンドリングの問題が，大きく関与しているものと考えられる。以下，順に，神経，首等の疲労であるが，全体を通じていえることは，生理的疲労は，操作性並びに環境の問題を，総合的に考えて，原因をさぐり，解決しなければならないということである。

表4－2　生理的疲労の原因　（百分率：数量／回答者数）

No.	個所	主な理由	数量	百分率
1	眼	長時間使用，照明の反射，不鮮明	52	65.8
2	肩		8	10.1
3	神経		5	6.3
4	首		4	5.1
5	腕	作業位置が高い	2	2.5
6	指		2	2.5
7	背中		1	1.3
8	腰		1	1.3
9	全身		2	2.5
	特になし		5	6.3
	無回答		14	17.7

4.3.3 操作性に関する側面

ここでは，人間がターミナルに接する際の操作性の問題を，ハードウェア・ソフトウェア・その他の面で述べることとする。これらは，表4－3に表わす様に，多岐に渡っており，相互に結びついているものと考えられるが，ユーザが，問題点として挙げたものの中で集中したのは，操作中の作業スペースが，狭少もしくは，無に等しいということを，考えてみると，これには，作る側の論理と使う側の論理のギャップが現われたものであるが，非常にプリミティブな項目であるが，案外盲点になっているということであろう。現在のターミナルは，汎用化指向となっているため，使う側の使い方，作業分野，内容等に対しても，平均的なところを目指している点が，使う側にとっては，ぴったりフィットして，使い易いということに成り難いという面などに現われているのである。

ついで，ハードウェア面での要求が多かったのは，画面の表示量の拡大である。これは，通常使っている書類の大きさに合わせたいという人間心理の慣性に，起因するものと考えられるが，主としてエンジニアとかプランナの意見であることから考えて，一つの画面で，同時に考えられる範囲を増やしたいということかもしれない。この点は，観点を変えれば，用途別に，必要な画面の大きさというのは，異なり，ユーザの業種別のワーク・ステーション等の登場を期待しているのではないかとも読みとれるのである。続いて，意見が集中したのは，キーボードに関することであるが，日本人がやはりタイプライタ人種ではないと，言ってしまえば，それまでであるが，キーボードの配列も，従来の慣習に，こだわることなく，使い易さの追求をすべきであり，また，機種間に，統一性が無い点は，ユーザにとってみれば，非常に，迷惑なことであるので，統一の方向に向って，努力すべきであろう。その他に，画面の向き，文字，ライトペン操作，音等の問題もあるが，これは，表4－3のコメントの欄を参照されたい。

また，ソフトウェア面で，要求が集中したのは，エラー・メッセージ，モード変換であるが，両方ともに，言えることは，マン・マシン・ユーザ・インタラクションの面で考えるなら，最も基本的な点で，人間が，マシン側に期待するのは，入力した内容に対する，期待する答が迅速に出力されることが第一ではあるが，途中の過程でのやりとり，換言すれば，会話の交換のスムーズ性を，期待しているということであろう。即ち，人間の思考を支援するツールとして，インタラクションのスムーズ性が，要求されているのである。とくに，エラー・メッセージに関しては，人間のアクセスの間違いによる（？）のであるが，その内容は，ソフトウェア側から見て，人間のプロセスの誤りに対する再アクセスの要求が多いが，単に，メッセージの内容を改善して，ユーザに，わかり易くして欲しいという言葉の裏には，ソフトウェアのアルゴリズムの改善により，人間が，頻繁に犯す，アクセスの誤りを，フォローする様にして欲しいという要求と，とれないこともないであろう。

表４－３　操作性の問題点　（百分率：数量／回答者数）

分　類	項　目	コメント	数　量	百分率
1.ハード ウェア面	画面の文字の大きさ	小さい，読みにくい	13	16.5
	画面の文字の輝度・色	入力文字（暗），出力文字（明）照明が反射してみにくい，色分けしろ	11	13.9
	画面の大きさ	表示量を大きく，60行×132桁	24	30.4
	画面の向き	多少上からながめる程度，45°位	13	16.5
	画面のちらつき	リフレッシュの際，表示量の多いとき	8	10.1
	キーボードの配列操作性	機種ごとに違うのでやりにくい，使用頻度を考えて配列せよ，ストロークを短くしろ	28	35.4
	端末発生音	ファン，L／Pがうるさい，調整できるように	19	24.1
	ライト・ペン 操作	感度が悪い，ワイヤレスにしろ，置く場所に一工夫，コードがじゃまになる	20	25.3
		小　計	136	172.2
2.ソフト ウェア面	エラーメッセージ	日本語で表示せよ，わかりやすくユーザのとるべき処置の明示	20	25.3
	モード変換方法	もっと簡単にならないか	5	6.3
	そ　の　他	インプット中の桁数を数え易く，カーソルの横移動5ないし10飛としろ，データ修正はフルスクリーンで待ち時間終了時の合図を	6	7.6
		小　計	31	39.2
3.環　境	操作中の作業スペース	少いもしくは狭いデータを広げられない	45	57.0

4.3.4.　作業環境に関する側面

作業環境に関する問題点，もしくは要求については，表４－４に表わすとおりであるが，内容としては，照明・空調・騒音・作業スペースに関する項目が入っている。ここでも，操作性のところで述べた，作業スペースに，集中しているが，これは，アンケートの項目のミスで，オーバラップしている為である。ここの内容では，作業台と椅子に分けているために，さらに細かくみると，スペースの広さの次に，椅子の硬度が，問題となっているが，これは，作業時間とその内容に，起因しているものと考えられる。

ついで，照明・空調・騒音が，ほぼ同数の問題点であるが，騒音の場合は，白色雑音・暗騒音等の周囲からでてくるものと，ハードウェアそのものがだすものとが，混在化しているものと考

えられるが，これについても，人間とマシンとのインタラクションが，煩雑・難解になったときに，より強く感じられるものであろう。

表4－4　作業環境の問題点　（百分率：数量／回答者数）

No.	項目		コメント	数量	百分率
1.	照明		暗い，暗すぎる	14	17.7
			明るい，まぶしい	5	6.3
			画面に反射する	5	6.3
			小計	24	30.4
2.	空調		ききすぎる，特に冷房	14	17.7
			きかない	10	12.7
			小計	24	30.4
3.	騒音		うるさい，耳ざわりである	22	27.8
4.	作業関連	作業台	せまい	38	48.1
			高さが合わない	12	15.2
			専用台が欲しい	2	2.5
			小計	52	65.8
		椅子	かたい	18	22.8
			高さが合わない	11	13.9
			小計	29	36.7
			中計	81	102.5
5.	その他		別室で落ちついてやりたい	6	7.6
			専用のカバーが欲しい	2	2.5

また，空調の場合は，アーキテクチャ上，ＣＰＵの近くに置く必要のあるターミナルの場合に，特に多く，これは，ＣＰＵ自身が，うまく機能するための環境が，人間の快適環境を越えるためと考えられる。これに対しても，人間が頻繁に，接触するターミナルは，リモートで操作できる様な処置を施すことで解決できよう。さらに，照明は，基本的に，何ら考慮されておらず，一般のビル内の照明と同じで，作業の精密度，思考性等に対する配慮はなく，天井フラット型の照明により，平面上でしか，基本的照度をとってないために，おこることで，改良するには，各ターミナルごとに，照明を考えていく必要があろう。

4.3.5　将来に要求される機能

本件に関する要求のバラツキは，表4－5に示す様に，現時点における改善のレベルから，空想のレベルまであり，一定していないが，いくつかの傾向が見られる様に思われるのである。1つは，パーソナル指向であり，これは，個人の秘書的に処理してくれる，専用の小型ポータブル・タイプのもので，要求内容は，個人のスケジュール管理，データ管理から始まって，作業を省力化する為の，口述筆記・翻訳・音声による入出力・日本語によるアクセス等である。

次には，オートメーション指向である。エンジニアリング・ベースまたはプランニング・ベー

表4－5　将来要求される機能　（百分率：数量／回答者数）

No.	項　　目	コ　メ　ン　ト	数　量	百分率
1	音声による入出力	女性の声で応答，操作ミスの音声による訂正，キーボードをなくす	21	26.6
2	小型化（卓上化）	ポータブル化，表示部の平面化，薄型	8	10.1
3	フレキシブルな作表・作画機能	平面図から立体図 仕様書から図面に	6	7.6
4	手書き文字・図形の認識	入力媒体を自由に	5	6.3
5	日本語でのアクセス	漢字，手書漢字ＯＣＲ	4	5.1
6	カラー・ハード・コピー	多色彩	4	5.1
7	処理中，応答中に別のことが多重でできないか		4	5.1
8	個人用の専用端末	パーソナル化，自宅でもできる，口述筆記，スケジュール管理，情報管理	3	3.8
9	画面の使い分け	キャラクタ部とグラフィック部	3	3.8
10	翻　訳　機　能		2	2.5
11	脳波からの直接入力		2	2.5
12	使用しても疲れない		2	2.5
13	データ編集の簡素化		1	1.3
	無　回　答		28	35.4

スの思考プロセスで，省力化を目指す手書きの文字・図形等の認識機能，入力データからの，フレキシブルな作表・作画機能，データ編集の簡易化等が主であるが，これなどは，いわゆるユーザの創造的活動の，支援ツールとしての願望であろう。その他に，変ったところでは，脳波の変動を，うまく認識・理解させたいという要求であろうが，現状の入出力の際の手続・アクセスの煩雑さに対する痛烈な反論とともに，将来の夢として，人間とマシンのテレパスによるコミュニケーションという様な，ＳＦ的なものとが，輻輳しているが，非常に，示唆的な意見である。また，１つの画面の使い分け，即ち，キャラクタ部とグラフィック部を，混在できないかという要求もあったが，これは，将来のワーク・ステーションとして，全体を考える際の方向とも考えられる。また，一般的要求としては，ハード・コピーの多色彩化，キーボードの改良，レスポンス・タイムの改良などがあげられている。

4.4　人間工学的側面からみた今後のマン・マシン・ユーザ・インタフェイスのあり方

価値の体系が破壊され多様化し，産業構造の変革を余儀なくされている現在，その内容は次第に知識集約型へ移行してくるであろう。そうすればますます，創造的開発型業務が多くなり，これを推進するには，人間の持つ創造力，直観力等の能力と，コンピュータの特性である数値解析，データの大量蓄積等の能力との，調和のある協調が，必要となってくると思われる。この人間の創造的作業を支援するツールとしてのシステムにおいては，マン・マシンの問題はますます重要になってくると思われる。人間の思考プロセスを考えてみると，その内容は，過去のデータを検索し，分析し，さらには方向をきめ，思案をめぐらすという様な形となろう。ここで，人間が最も苦手なのは，データ検索とか，数値解析とかいう緻密で，やっかいな，単純的繰返し作業である。これは，コンピュータの最も得意とするところであるから，相互に補いながら，仕事を進めることは，当然考えられるが，一歩進んで，人間の創造力，直観力をたくましくする情報で，システムと会話できれば，さらに完全なものとなろう。それは何であろうか？非常に難しい問題であるが，それは図とか絵という人間が一目みて，理解できるものか，あるいは日常的会話状態で会話を持つことか，いろいろ考えられるが，人間が，コンピュータに要求する機能として考えられるのは，連想機能，学習機能，図形等の非定型データの認識・理解機能，処理の高速性，自然言語に近い形での会話などであろう。この様な機能を有するものがあれば，多品種少量生産時代を迎えるにあたって，生産上での高技能を要求する人間の知的活動が支援できるであろう。ここで再度マン・マシン・ユーザ・インタフェイスの方向として考えられることは，マンとマシンの接点の改良技術にとどまることなく，人間の知的活動を支援するツールとして何が必要かというコンセンプトを作り，すでに述べたような機能を開発していくことであろう。その機能のバックグラウンドとしては，知識ベースとデータベースとの調和あるリンクを達成する必要があるであろう。

参　考　文　献

1）人間工学概論，　朝倉書店，　真辺春蔵，　長町三生編集

2）人間工学，　日刊工業新聞社，　坪内和夫

3）造形心理学入門，　美術出版社，　本明　寛

4）もう一つの技術，　学陽書房，　総合研究開発機構編集

5）心理生理，　中山書房，　新美良純

6）心理学評論，　「心理学評論」

7）動作時間研究，　日刊工業新聞社，　大坪　壇訳

8）人間工学ハンドブック，　金原出版

9）産業心理Ⅱ（照明），　河出書房，　真辺春蔵

10）心理学的測定法，　東大出版，　田中良久

11）精神測定法，　培風館，　秋重義治訳

12）工業における官能検査ハンドブック，　日科技連，　増山元三郎，　三浦　新

第Ⅱ部　情報システムにおける分散型データベース技術の調査研究

1. 序　論

近年のVLSI技術の目ざましい進歩は，情報処理全体に大きなインパクトを与えてきている。現在のメインフレームの中～大型機並みの処理能力とメモリを備えたパーソナルコンピュータの実現はここ数年以内に可能となり，今まで情報化し得なかった種々の分野へこうした新しいシステムが普及していくと思われる。情報化の1つの大きな目標は工業生産性と比較した時の低生産性が現在問題となっているオフィス業務の自動化である。小型のパーソナルコンピュータを各従業員に対して設置し，これらとセンタの大型システムとを通信ネットワークによって有機的に結合した情報システムによって，オフィスの情報処理システム（オフィス情報処理（OIS））は実現される。この時の主要な問題は，情報システムにおいて最重要な共有リソースとしてのデータベースが新たに情報システムのなかで，どの様に位置づけられるかである。将来，データベースシステムは，現在の文字数値情報に加えてプログラム，図形，画像といった新たな質のデータを扱かっていかなければならない。オフィスを例にとっても多く（ほとんど）の情報が現在のデータベース技術では，データベース化し得ていない。この理由として我々は次の2点があると考えている。

(1) 統合概念に基づいた現在のデータベースシステムの限界

(2) 各サイトごとに生成されたデータを，サイト間で有機的に統合利用する技術即ち，分散型データベースシステム（DDBS）技術の未熟さ

の無矛盾性を一切認めない現在のデータベースシステムから，データの論理的冗長性をかなり認めた新たなデータベースシステムが必要になってくるように思える。この問題について、我々は第3章において，新たなデータベース概念としてのフォームフローシステムを提案している。我々の基本的考え方は，今後情報システムはデータベース中心に構築されていかなければならず，データベースは従来のデータに加えて，プログラム．そして整理されていないデータを積極的に組みこんでいくものでなければならないという点である。

今述べた様に，将来の情報システムは現在の大型機並み又はそれ以上の処理能力と格納容量を備えた小型情報システムと，特徴を備えた超大型情報システムとが通信ネットワークによって有機的に結合されたものであろう。この時情報システムの核としてのデータベースリソースは最も主要な役割を演じる。我々はここ5年間では，ローカルネットワークによってある部局単位（データ利用の単位）に上述のシステムが統合されていくものと考えている。各情報システムのデータリソースは目的ごとに統合され，抽象化され，ユーザはシステムの形態を意識することなく，必要な情報を得れる。この新たな情報システムを支える技術は分散型データベースシステム（DDBS）である。本報告書では当協会が昭和52年度－54年度にかけて研究開発した分散型データベースシステムJDDBS－I〔JDDBS80〕を基礎にして，オフィス情報システム等の新たな情報システムの基本技術となりうる第II期分散型データベースシステム（JDDBS－II）のモデル化とその一部のインプリメンテーションについて主に述べてある（第5章～7章）。

具体的には第2章では，既存データベースモデルの再整理を行ない，特にCODASYLモデルの形式

化を試みた。第3章では，新たなアプリケーションとして重要なオフィス情報処理とデータベースとデータとの関連について検討し，このモデルとしてのフォームフローモデルを提案した。第4章から第7章にかけては，第Ⅱ期分散型データベースシステムJDDBS－Ⅱのモデル化を試みている。第4章では，JDDBS－Ⅱの全体アーキテクチャを論じ，第5章では問合せの処理概要を述べている。第6章では，ローカル共有媒体ネットワークを用いた時の通信処理モデルと処理アルゴリズムの提案と評価を行なっている。第7章では，CODASYL DBTG データベースシステムのリレーショナルインタフェースについて論じ，そのインプリメンテーションと評価とを行なっている。このシステムは，DDBSにおいて異種DBSの共通インタフェースになるとともに，各DBSのエンドユーザインタフェースともなる。

2. データベースシステムモデル

2.1 データベースシステムアーキテクチャ

ここでは，2.1.1でデータベースが何を表現するのか，2.1.2でデータベースの特徴と3層スキーマ構造について述べることにする。

2.1.1 データベースとは ？

データベースとは実世界（ reality ）をモデル化したものである。しかし実世界は，無限の情報をもつために実際には，この実世界の一部（ a slice of reality ）をモデル化したものである。

実世界の一部のなかでも，時間的変化に対して不変な部分と可変な部分とが存在する。不変な部分をモデル化したものが，intension（内包）と呼ばれるものであり，可変部分をモデル化したものが，extension（外延）と呼ばれるものである。

実世界の一部をモデル化するための記述系がデータモデルと呼ばれるものである。現在，データモデルとしては，リレーショナル，階層型，ネットワーク型がある。また，モデルをデータモデルを用いて記述したものをスキーマという。

2.1.2 データベースの特徴と3層スキーマ

従来のファイルシステムからデータベースシステムを特徴づける概念として統合（ integration ）概念がある。統合とは，関連する幾つかのアプリケーションによって用いられるデータを1つのユニークなデータの集合に統一することである。データは個々のアプリケーション固有なものとしてではなく，関連するアプリケーションの集合としての共同体によって共有されるものとしてとらえることである。この論理的な集中化によってシステム全体におけるデータの冗長性を減らし，データの無矛盾性（ consistency ）を高めることができる。

また，データを論理的に集中させるだけでなく，データを物理的側面と論理的側面に分離して，物理的な側面の進化（ evolution ）から，データの記述とアクセスを守るデータ独立性〔CODDE70〕の概念が導入された。データ独立性の概念を発展させた概念として，ANSI／X3／SPARC〔ANSIX75〕が，三層スキーマ構造の概念を導入した。三層構造は論理的な側面を更に，ユーザのアプリケーションに依存する部分と，論理的に不変な部分とに分離するものである。

三層構造では，データベースシステム（以降DBSと略す）に対して次の様な3つのスキーマ階層を設けている。

① 外部スキーマ層

データベースシステムのアプリケーションに依存する部分

② 内部スキーマ層

物理的アクセスと格納の有効性に関した部分

③ 概念スキーマ層

アプリケーションの多様性と，物理デバイスの進化に適応するために，外部スキーマ層と内部スキーマ層とのマッピングの核となる部分。データベース全体の意味を表わす層（ semantic integration ）の概念スキーマはＤＢＳ内で中心的な役割を持っている。

各スキーマ層は，あるデータモデルと言語とによって特徴づけられる。

データベースの三層スキーマ構造を示したのが図2.1である〔DATEC77〕。

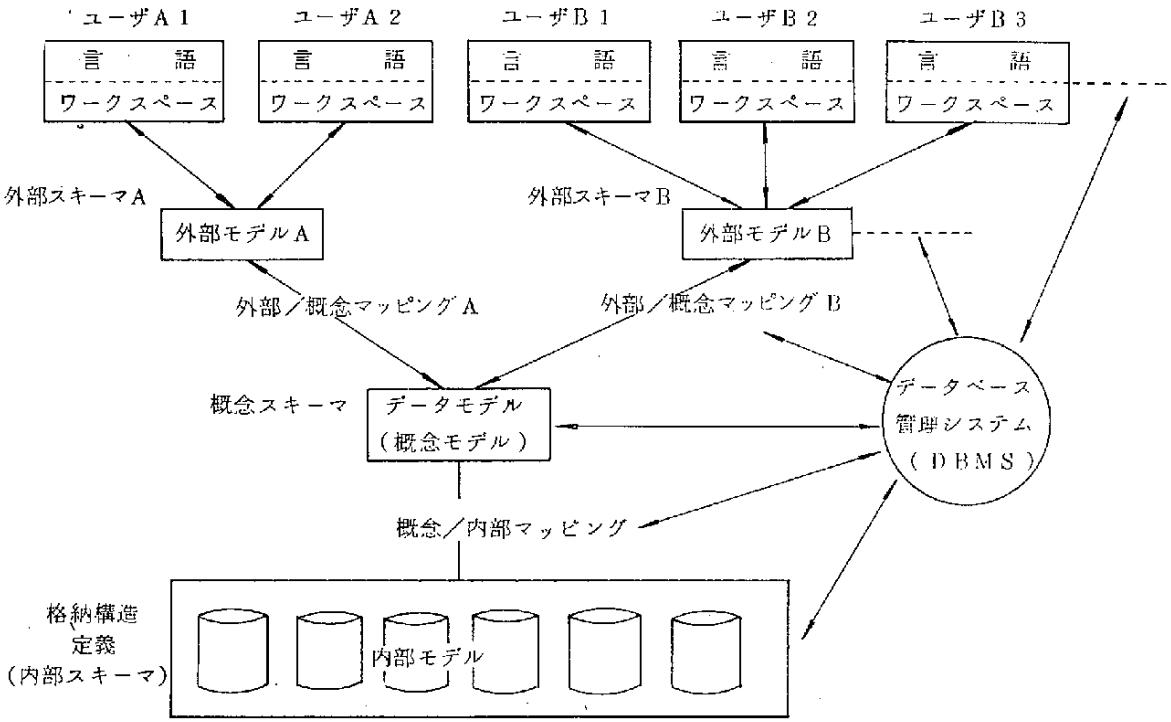


図2.1　データベースシステムのアーキテクチャ

2.2　リレーショナルモデル

ここでは，リレーショナルモデルに関して，2.2.2でextension（外延）の定義を，2.2.3でintension（内包）の定義を，2.2.4で従属関係を，2.2.5で正規形を，2.2.6でデータサブ言語を，2.2.7 viewについて述べることにする。

2.2.1 はじめに

データベースは実世界の情報を記述するものであるが，実世界の情報としては実世界を表わす事象（entity）とその事象間の関係性（relationship）が存在する。

リレーショナルモデルでは，この事象と事象間の関係を同一のものとして扱っている（他のデータモデル，例えばCODASYL DBTG モデルでは，事象と事象間の関係性は完全に区別されている）。リレーショナルモデルでは，事象も事象間の関係性もリレーションとして扱い，テーブル（値の集合）という単一形式で表わしている。このことによりデータを扱う基本操作（検索，変更，削除，挿入）が同種のオペレーションで実行できる。

また，リレーションは数学的に定義されているので，形式的に理解されやすく，設計もしやすい（e.g. D－tree 〔TANAY 79 a〕）一面，データの意味表現力の弱さをもっている。

リレーショナルモデルの利点をあげると，

(1) 簡潔性

データとして単一形式の表（リレーション）を扱う言語が非手続き型言語である。

(2) Closure

I／Oオペレーションの結果をリレーションとして同一オペレーションでアクセスできる。他のモデルはできない。

(3) データ独立性（data independency）

データ独立性として，物理的独立性と論理的独立性がある。

○物理的独立性

Storage構造の変化に対するプログラムの変更を防ぐ。これは，データモデルとして，リレーショナルモデルは物理的構造と明確に分離されていることによる。

○論理的独立性

データモデルの定義の成長に対するプログラムの変更を防ぐ。リレーショナルモデルの場合，ANSI／X3／SPARCの3層スキーマ構造に対応して，論理的側面を外部スキーマ（view）と概念スキーマに分類することが出来ることによる。

これらの利点のために，リレーショナルモデルは非定型業務に適していると考えられる。

2.2.2 リレーショナルモデルのextensional 定義

リレーショナルモデルのextension(外延)は，データベースの中味を表わすもので，リレーションで定義され，これは時間的に変化する。

いま，Diを値の集合とする（Di内の各値は，互いにユニークである）。しかし，各Diは互いに，distinctである必要はない（即ち，Di = Dj（i≒j）も可である）。

D_1，D_2，……，D_nの直積（cartesian product）をDとすると

$$D = D_1 \times D_2 \times \cdots\cdots \times D_n$$

$$= \{ (d_1, d_2, \cdots, d_n) \mid d_1 \in D_1 \wedge d_2 \in D_2 \wedge \cdots\cdots \wedge d_n \in D_n \}$$

D_1, ……, D_n の直積空間の中で，ある関係を満足する組（d_1, d_2, ……, d_n）の集合をリレーションRと呼ぶ。

$$R \in D_1 \times D_2 \times \cdots\cdots \times D_n$$
$$= \{ (v_{i1}, v_{i2}, \cdots\cdots, v_{in}) \mid i, i \in \{ 1, 2 \cdots\cdots C \} \wedge (v_{i1}, \cdots\cdots, v_{in}) \in D_1 \times \cdots\cdots \times D_n \wedge (v_{i1}, \cdots\cdots, v_{in}) \neq (v'_{i1}, \cdots\cdots v_{i'n}) \wedge i \neq i' \}$$

この時，nをRの次数（degree），CをRのカーディナリティ（cardinality），（v_{i1} ……, v_{in}）をRの組（tuple）と呼ぶ。

各D_iの位置に対応させて，ユニークな名前を割り当て，この名前を属性（attribute）という。属性A_iは，リレーションRからD_iへの写像である。

$$A_i : R \rightarrow D_i$$

A_iをRのi番目の属性，D_iをA_iの定義域（Domain）という。

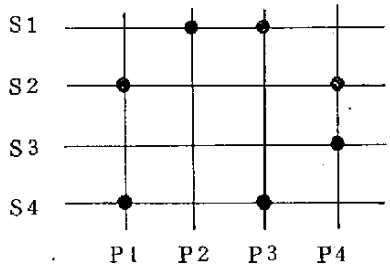
リレーションの例として，サプライヤー部品データベースについて考えてみる。

$D_1 = \{ P_1, P_2, P_3, P_4 \}$　　A_1 ＝ 部品番号（P＃）

$D_2 = \{ S_1, S_2, S_3, S_4 \}$　　A_2 ＝ サプライヤ番号（S＃）

$D_3 = \{ 100, 200, 300, 400 \}$　A_3 ＝ 数量

とした時に，サプライヤが供給する部品の関係をみた場合，図2.2の様になっていたとする。これをテーブル形式で書き直すと表2.1の様になる。



・印が条件を満足する組である。

図2.2　サプライヤが供給する部品の関係

S＃（サプライヤ番号）	S＃（部品番号）
S1	P2
S1	P3
S2	P1
S2	P4
S3	P4
S4	P1
S4	P3

表2.1　サプライヤが供給する部品の関係

表2.1の場合，次数は2，カーディナリィティは7である。

2.2.3 リレーショナルモデルの intentional 定義

リレーショナルモデルの intension（内包）とは，データベースを記述するもので，リレーショナルスキーマで定義される。

リレーションの時間不変部分の記述をリレーショナルスキームと呼ぶ。またリレーショナルスキームの集合をリレーショナルスキーマと呼ぶ。

リレーショナルスキームは，リレーション名，属性名，インテグリィティ条件（従属関係）によって定義される。

リレーショナルスキーム

：＝＜リレーション名＞＜Ω，Γ＞

但し，Ωは属性の集合

Γは従属関係の集合

従属関係については，次章で述べる。

2.2.4 従 属 関 係

A 関数従属性（ functional dependency ）

実世界の意味構造に関する概念がスキーマには必要である。関数従属性もその概念の1つである。

あるリレーションR（X，Y，Z）で，X，Y，Zを属性の集合とし，X∩Y＝∅，Y∩Z＝∅，Z∩X＝∅，すなわちX，Y，Zが互いに disjoint であるとき，R内の属性集合Xの各々の値が属性集合Yの唯一つの値とのみ関連していることをYがXに関数従属しているという。このことを，X→Yを表わすことにする。例えば，サプライヤー部品データベースで，各サプライヤーが一つの都市にしか存在しない場合，都市名はサプライヤ番号に関数従属していることになる。

関数従属関係（以後ＦＤと略する）に関して次のことが成り立つ。

（ＦＤ０） 自明な関数従属性

$X \supseteq Y \Rightarrow X \rightarrow Y$

（ＦＤ１） reflexivity （反射律）

$Y \subseteq X \Rightarrow X \rightarrow Y$

（⇒は，左辺が成り立てば右辺が成り立つことを意味する。）

（ＦＤ２） augmentation

$Z \subseteq W \wedge X \rightarrow Y \Rightarrow XW \rightarrow YZ$

ＦＤ３のようにＹを通して，Ｘに関数従属するものを，この場合，ＺがＸに推移従属しているという。

この他に上の3つのFD1～FD3を使って，次のことを証明することができる。

（FD4） pseudo－transitivity

$X \rightarrow Y \wedge YW \rightarrow Z \Rightarrow XW \rightarrow Z$

（証明）

$X \rightarrow Y \overset{FD2}{\Rightarrow} XW \rightarrow YW$

$XW \rightarrow YW \wedge YW \rightarrow Z \overset{FD3}{\Rightarrow} XW \rightarrow Z$

（FD5） UNION

$X \rightarrow Y \wedge X \rightarrow Z \Rightarrow X \rightarrow YZ$

（証明）

$X \rightarrow Y \overset{FD2}{\Rightarrow} X \rightarrow XY$

$X \rightarrow Z \overset{FD2}{\Rightarrow} XY \rightarrow YZ$

$X \rightarrow XY \wedge XY \rightarrow YZ \overset{FD3}{\Rightarrow} X \rightarrow YZ$

（FD6） decomposition

$X \rightarrow YZ \Rightarrow X \rightarrow Y \cap X \rightarrow Z$

（証明）

$YZ \rightarrow Y$ （FD1より）

$YZ \rightarrow Z$ （ 〃 ）

$X \rightarrow YZ \wedge YZ \rightarrow Y \overset{FD3}{\Rightarrow} X \rightarrow Y$

$X \rightarrow YZ \wedge YZ \rightarrow Z \overset{FD3}{\Rightarrow} X \rightarrow Z$

B 多値従属関係（Multi－valued dependency）

実世界の意味構造は関数従属だけでは表わされず，多値従属関係の概念が導入された〔FA－GIR77〕。

あるリレーションR（X，Y，Z）で，X，Y，Zは属性の集合で，$X \cap Y = \emptyset$，$Y \cap Z = \emptyset$，$Z \cap X = \emptyset$であるとする。この時YがXに多値従属することを$X \rightarrow\!\!\rightarrow Y$と表わすことにする。

$X \rightarrow\!\!\rightarrow Y$がリレーションR（X，Y，Z）で成り立つための必要十分条件は，

(1) R内のXの各々の値は，Yの値の集合と関連している。

(2) このとき，(1)の関連性が，Zの値に依らない。

ことである。

この例としては，親と子供の関係が多値従属関係である。

リレーションR（A）で，

$X \subseteq A$，$Y \subseteq A$，$Z = A - (X \cup Y)$が成り立ち，Y_{XZ}を次の様に定義する。

$$Y_{XZ} \triangleq \{ y \mid <x, y, z> \in R \wedge \exists z \in Z \}$$

X→→YがR内で常に成り立つための必要十分条件は，

全てのx，z，z′に対して，

1) $Y_{xz} \neq \emptyset$，$Y_{xz'} \neq \emptyset$

2) $Y_{xz} = Y_{xz'}$

が成り立つことである。

これはY_{xz}が，xにのみ依存していることを表わしている。

例えば，表2.2の様なリレーションを考えると，X＝COURSE，Y＝TEACHER，Z＝TEXTとすると，

$$TEACHER_{Physics,\ Basic\ Mechanics} = \{ Prof.\ Gree, Prof.\ Brown \}$$

$$TEACHER_{Physics,\ Principle\ of\ Optics} = \{ Prof.Gree, Prof.\ Brown \}$$

また，

$$TEXT_{Physics,\ Prof\ Green} = \{ Basic\ Mechanics, Principle\ of\ Optics \}$$

$$TEXT_{Physics,\ Prof\ Brown} = \{ Basic\ Mechanics, Principle\ of\ Optics \}$$

CTX

COURSE	TEACHER	TEXT
Physics	Prof. Green	Basic Mechanics
Physics	Prof. Green	Principles of Optics
Physics	Prof. Brown	Basic Mechanics
Physics	Prof. Brown	Principles of Optics

表2.2

が成り立つ。よってTEACHER，およびTEXTは各々COURSEに多値従属であることが言える。

これはTEACHERとテキストがCOURSEに対して相互独立であることに原因がある。

多値従属関係（以後，MVDと略する。）もFDと同様に次のことが成り立つ。

（MVD0） complementation

$X \cup Y \cup Z = A \ \wedge \ Y \cap Z \subseteq X$が成り立つと仮定すると，

$X \rightarrow\rightarrow Y \Leftrightarrow X \rightarrow\rightarrow Z$

（MVD1） reflexivity（反射律）

$Y \subseteq X \Rightarrow X \rightarrow\rightarrow Y$

（MVD2） augmentation

$Z \subseteq W \ \wedge X \rightarrow\rightarrow Y \Rightarrow XW \rightarrow\rightarrow YZ$

（MVD3） transitivity（推移律）

$X \rightarrow\!\!\rightarrow Y \wedge Y \rightarrow\!\!\rightarrow Z \Rightarrow X \rightarrow\!\!\rightarrow Z - Y$

また上記のMVDO～MOD3を使って次の規則を導くことが出来る。

（MVD4） pseudo − transitivity

$X \rightarrow\!\!\rightarrow Y \wedge YW \rightarrow\!\!\rightarrow Z \Rightarrow XW \rightarrow\!\!\rightarrow Z - YW$

（証明）

$X \rightarrow\!\!\rightarrow Y \overset{MVD2}{\Rightarrow} XW \rightarrow\!\!\rightarrow YW$

$XW \rightarrow\!\!\rightarrow YW \wedge YW \rightarrow\!\!\rightarrow Z \overset{MVD3}{\Rightarrow} XW \rightarrow\!\!\rightarrow Z - YW$

（MVD5） union

$X \rightarrow\!\!\rightarrow Y \wedge X \rightarrow\!\!\rightarrow Z \Rightarrow X \rightarrow\!\!\rightarrow YZ$

（証明）

$X \rightarrow\!\!\rightarrow Y \overset{MVD2}{\Rightarrow} X \rightarrow\!\!\rightarrow XY$

$X \rightarrow\!\!\rightarrow Z \overset{MVD2}{\Rightarrow} XY \rightarrow\!\!\rightarrow ZY$

$X \rightarrow\!\!\rightarrow XY \wedge XY \rightarrow\!\!\rightarrow YZ \overset{MVD3}{\Rightarrow} X \rightarrow\!\!\rightarrow YZ - XY$

（MVD6） decomposition

$X \rightarrow\!\!\rightarrow Y \wedge Y \rightarrow\!\!\rightarrow Z \Rightarrow X \rightarrow\!\!\rightarrow Y \cap Z$ (1)

$X \rightarrow\!\!\rightarrow Y - Z$ (2)

$X \rightarrow\!\!\rightarrow Z - Y$ (3)

（証明）

属性集合Yに対し，A − YをY^cで表わすことにする。(2)，(3)は対照的なので，(1)，(2)を証明する。

$X \rightarrow\!\!\rightarrow Y,\ X \rightarrow\!\!\rightarrow Z \overset{MVD0}{\Rightarrow} X \rightarrow\!\!\rightarrow Y^c,\ X \rightarrow\!\!\rightarrow Z^c$

$X \rightarrow\!\!\rightarrow Y^c,\ X \rightarrow\!\!\rightarrow Z^c, \overset{MVD5}{\Rightarrow} X \rightarrow\!\!\rightarrow Y^c Z^c$

(1) $X \rightarrow\!\!\rightarrow Y^c Z^c \overset{MVD0}{\Rightarrow} X \rightarrow\!\!\rightarrow Y \cap Z$

(2) $X \rightarrow\!\!\rightarrow Y^c,\ X \rightarrow\!\!\rightarrow Z^c \Rightarrow X \rightarrow\!\!\rightarrow Y \cap Z^c \overset{MVD6(1)}{\Rightarrow} X \rightarrow\!\!\rightarrow Y - Z$

2.2.5 正規形（Normal form）

A．序

リレーショナルモデルにおいては，リレーションを正規形にする正規化（normalization）は大きな問題の１つである。リレーションを正規形にする目的には次の２つの点にあると言える。

(1) データの更新異常を避けること。

(2) データの冗長性を減少すること。

である。

それに加えて，正規化することはリレーションが表わす実世界の意味や制限を正確にリレーションへ反映させることでもある。

正規形には，第1正規形，第2正規形，第3正規形，Boyce－Codd 正規形，Bernstein 正規形及び第4正規形がある〔TANAY　79C〕。各々の正規形の関係は図2.3の様になっている。矢印の下の正規形は上の正規形を包含していることを表わしている。

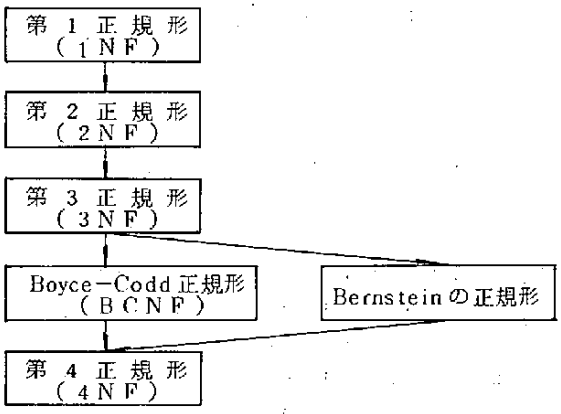


図2.3　正規形の関係

B．リレーションにおけるキーの概念

リレーションの中の組（ tuple ）を一意に識別できる属性又は属性の集合を候補キー（ candidate key ）という。また，この候補キーの中で最小の属性集合をもつものを主キー（primary key ）という。候補キーのうち，複数の属性からキーが成り立っているものを複合キー（Supper key又はComposite key ）という。

その他に，あるリレーションR1では主キーではないが，別のリレーションR2で主キーである属性があるとすると，その属性は外部キー（ foreign key ）であるという。

上記のいずれのキーでもない属性または属性の集合を非キー（ non key ）という。

C．第1正規形（1NF）

リレーションRが第1正規形であるというのは，すべての定義域が原子値だけを含んでいることを言う。

第1正規形ではあるが，第2，第3，第4正規形でないリレーションはいくつかの理由により望ましくない構造を持っている。

この点を明らかにするために，次のようなリレーションを考えてみる。FIRST（S#，STATUS，CITY，P#，QTY ）というリレーションについて考えてみる。従属関係は，QTYが

（S#，P#）の組に，STATUSとCITYが各々S#に，そしてSTATUSがCITYに関係従属であると仮定する。従属関係を図示すると図2.4の様になる。またFIRSTサンプルテーブルを表2.3に考えてみる。

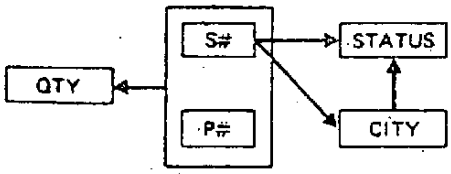


図2.4　FIRSTの関数従属図

FIRST

S#	STATUS	CITY	P#	QTY
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	P2	200
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	400

表2.3　FIRSTのサンプルテーブル

更新時の問題点として次のことが起こる。

◦挿入

ある都市にあるサプライヤに対する情報は少なくとも1つの部品を供給しないかぎり挿入できない。

◦削除

あるサプライヤに対するFIRSTの組だけ削除したくても，そのサプライヤと関連するshipmentだけでなく，サプライヤのある都市の情報も消えてしまう。

◦変更

与えられたサプライヤに対するCITYの値はFIRSTにいくつもあらわれているので，この冗長性により，変更時に不一性を生ずる問題にぶつかる。

D．第2正規形（２ＮＦ）

リレーションRが第２正規形であるというのは，それが第１正規形であり，非キー属性が主キーに自明でない関数従属であることをいう。

C．のリレーションFIRST内には自明な関数従属関係S#→STATUS，S#→CITY が成り立っている。よってこのリレーションを適当なProjection（射影）*によって第２正規にすることができる。

従属関係は図2.5のようになり，リレーションSECONDとSPのサンプルデータは表2.4のようになる。

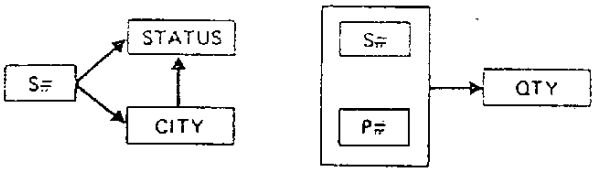


図2.5　SECONDとSPの関数従属図

SECOND

S#	STATUS	CITY
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London
S5	30	Athens

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

表2.4　SECONDとSPのサンプルテーブル

＊射影とは，テーブル形式のたて，すなわち指定する属性を取り出して新しいリレーションをつくるoperationである。

更新時の問題点

(1) 挿　入

C．で仮定したSTATUSがCITYに関数従属であるという事実が，あるサプライヤがその都市に出来るまでリレーション内に表わすことが出来ない。

(2) 削　除

もしある都市に対するSECOND組だけを削除したい場合でも，サプライヤに関する情報だけでなく，その都市がそのSTATUSを持つという情報が消えてしまう。

(3) 更　新

与えられた都市に対してSTATUSがいくつも表われているので，更新の時に不一致性が生ずる問題にぶつかる。

E．第3正規形（3NF）

リレーションRが第3正規形であるというのは，そのリレーションが第2正規形であり，すべての非キー属性が主キーに対して推移従属でないことをいう。

リレーションSECONDを第3正規形にするには，リレーションSC（S#，CITY）とCS（CITY，STATUS）に射影することである。この時の関数従属関係は図2.6の様になり，リレーションSCとCSのサンプルテーブルは表2.5の様になる。

ある与えられた時の与えられたリレーションのテーブル（snapshot）を見ることで，そのリレーションが第3正規性であるかどうか言うことはできない。そのリレーションが第3正規形であるかどうかは，データの意味を知ることが必要である。

図2.6　SCとCSとの関数従属図

SC

S#	CITY
S1	London
S2	Paris
S3	Paris
S4	London
S5	Athens

CS

CITY	STATUS
Athens	30
London	20
Paris	10

表2.5　SCとCSのサンプルデータ

F. Boyce－Codd の正規形（BCNF）

第2，第3正規形においては，主キーに対する条件しか付けられていなかったが，1つ以上の候補キーを持つ場合に対して第3正規形を拡張したものがBoyce－Coddの正規形（以降BCNFと略する）である。BCNFを定義する前，determinant（決定子）を定義しておく。determinantとは，他の属性によって関数従属されている属性または属性の集合のことを言う。

（BCNFの定義）

正規化されたリレーションRがBCNFであるというのは，determinantがあれば，常にそれはcandidate key（候補キー）であることをいう。

いま，オーバラップ（即ち，同じ属性を含む）している候補キーを持つリレーションの例について考えてみる。

リレーションSSP（S#，SNAME，P#，QTY）において，S#とSNAMEが相互に関数従属であり，SNAMEがユニークであるとする。この時，このリレーションの候補キーとしては，（S#，P#）と（SNAME，P#）である。しかし，determinantであるS#とSNAMEはリレーションSSPに対する候補キーではないのでBCNFにはならない。

ところがSSPは3NFである。この様にオーバラップする候補キーを持つリレーションに対して3NFは充分な条件を与えていないことになる。

G. Bernsteinの正規形

前述した正規形の他に，第3正規形の拡張ともいえるBernsteinの正規形と呼ばれるものがあるのでここに紹介しておく。

（Bernsteinの正規形の定義）

リレーションRがBernsteinの正規形であるというのは，任意の非キー属性Aに対して，候補キーに関数従属されておらず，また非キー属性を含まない属性Xが存在し，X→Aが成り立つことがないことをいう。

H. 第4正規形

第3正規形も望ましくない性質を持っている場合がある。それは第3正規形が多値従属関係を含んでいる場合である。

例えば，リレーションCTX（COURSE，TEACHER，TEXT）を考えてみる。仮定として，与えられたCOURSEに対して，関連する教師やテキストの数はいくつもある。また教師とテキストは互いに独立であるとする。リレーションCTXのサンプルデータを表2.6の様に表わす。

CTX

COURSE	TEACHER	TEXT
Physics	Prof. Green	Basic Mechanics
Physics	Prof. Green	Principles of Optics
Physics	Prof. Brown	Basic Mechanics
Physics	Prof. Brown	Principles of Optics

表2.6　CTXのサンプルテーブル

CTXは更新時において問題となる冗長を含んでいる。しかし，CTXは3NFである。これはCTXに関する場合，教師とテキストの間の相互独立に原因がある。教師はコースに多値従属である。即ち与えられた（コース値，テキストの値）の組にマッチする教師の値の集合は，個々のコースの値だけに従属している。

CTXをCT（COURSE，TEACHER）とCX（COURSE，TEXT）に分割すれば問題は改善される。

（第4正規形の定義）

リレーションRにnon－trivialな多値従属関係性X→→Yがある時，X，YはRの属性の部分集合であり，Rのすべての属性に対して関数従属性X→Aが成り立つことをいう。

2.2.6 言　語

A．概　要

データベースでデータを操作する言語をデータサブ言語（Data Sub－language，DSL）と呼んでいる。データサブ言語のうち，スキーマを定義する言語を，データ定義言語（Data Definition Language，DDL）と呼び，データを格納したり，検索したりする言語をデータ操作言語（Data Manipulation Language，DML）と呼ぶ。データサブ言語には，ホスト言語に埋め込む型式のものと独立型式のものと二種類ある。リレーショナルモデルにおけるデータサブ言語の特徴は，高度に非手続き型の言語であり，関係形式的に閉じた系（中間の出力結果を再び入力として使用することができる。）であることである。

次にリレーショナルデータサブ言語の例としてDSL ALPHA（リレーショナル計算言語）とリレーショナル代数（関係代数）についてとり上げてみることにする。

B．関係代数

関係代数は2つのoperationに分けることができる。集合演算（和，積，差，直積演算）を行なうset operationと関係演算（選択，射影，ジョイン，商演算）とに分けることができる。

1）集合演算

集合演算のうち和（union），積（intersection），差（difference）演算を行なうためのリレーションに対するための必要条件としてunion－compatibleという概念がある。まず最初にunion－compatibleについて定義する。

R，Sを2つのリレーションとする。R（r_1，……，rn）S（s_1，……，sm）。RとSとがunion－compatibleであるというのは，Rの内の各々の属性がS内に1対1に対応する属性をもつとともに，これらの対応する定義域が等しいことである。

すなわち，

R（r_1，……，rn），S（s_1，……sm）において，

n＝mで

すなわち

R (r_1, ……, rn), S (s_1, ……, sm) において,

n = m で

$(\forall r \in \{ r_1, \cdots\cdots, rn \})(\exists S \in \{ s_1, \cdots\cdots, sm \})$ (dom*(r)=dom(s))∧

$(\forall s \in \{ s_1, \cdots\cdots, sm \})(\exists r \in \{ r_1, \cdots\cdots, rn \})$ (dom(s) = dom(r))

いま，和演算，積演算と差演算の例として次のようなリレーションAとBを使うことにする。

A

a_1	a_2	a_3
1	A	1
2	B	2
3	C	3

B

b_1	b_2	b_3
1	A	1
2	B	3
3	C	3

① 和演算 (Union)

リレーションAとBがお互いに union − compatible であるとき，和演算を次の様に表わす。

A ∪ B

これは，Aもしくは B 又は両方に属するタプルを全て含んだ集合をつくる。

A∪Bの結果は，次の様になる。

A ∪ B

a_1	a_2	a_3
1	A	1
2	B	2
2	B	3
3	C	3

② 積演算

リレーションAとBがお互いに union − compatible であるとき，積演算を次の様に表わす。

A ∩ B

これは，AとBの両方に属するタプルを全て含んだ集合をつくる。

A∩Bの結果は次の様になる。

A ∩ B

a_1	a_2	a_3
1	A	1
3	C	3

＊ dom (r) は，r の定義域を表わす。

③ 差演算（difference）

リレーションAとBがお互いにunion － compatibleであるとき，差演算を次の様に表わす。

A － B

これは，Aには属し，Bには属さないタプルを全て含んだ集合をつくる。

A－Bの結果は次の様になる。

A － B

a_1	a_2	a_3
2	B	2

④ 直積演算（Extended Cartesian product）

リレーションAとBの直積演算は次の様に書くことができる。

A × B

これは，Aに属するタプルaとBに属するタプルbの連鎖であるタプルtを全て含む集合になる。

a＝（a_1，……，am）とb＝（bm＋1，……，bm＋n）の連鎖とは，

t＝（a_1，……，am，bm＋1，……，bm＋n）

をつくることである。

例えば，A（a_1，a_2，a_3）とB（b_1，b_2，b_3）を下のテーブルの様に仮定すると，A×Bの結果は次のようになる。

A

a_1	a_2	a_3
1	A	1
2	B	2
3	C	3

B

b_1	b_2
4	D
5	E

A×B

a_1	a_2	a_3	b_1	b_2
1	A	1	4	D
1	A	1	5	E
2	B	2	4	D
2	B	2	5	E
3	C	3	4	D
3	C	3	5	E

2）関係演算

① 制 限

テーブル形式のリレーション内に制限を満足するタプルを抽出して部分集合をつくる。

制限は，次の様に表わせる。

リレーション名〔属性名1 θ 属性名2 | 属性名 θ 値〕

但し，θは比較演算子$\{<, \leq, =, \geq, >, \neq\}$

例えば，リレーションS（S#，SNAME，STATUS，CITY）が下の様なテーブルであると仮定する。次の様な制限を出した結果は，以下のようになる。

$$R\,[a\,\theta\,v] = \{\, r \mid r \in R \wedge r\,[a]\,\theta\,v \,\}$$

S〔CITY ＝ 'LONDON'〕

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

→ 結果S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S4	Clark	20	London

② 射影（Projection）

リレーションから指定された属性または属性の集合を取り出し，冗長なタプルを取り除くオペレーションである。

これは，次の様に表わせる。

リレーション名〔属性名1，属性名2，……〕

例えば，リレーションP（P#，PNAME，COLOR，WEIGHT，CITY）が下のようなテーブルであると仮定する。

次のような射影を行なうと以下のようになる。

P〔P#〕

P

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Cam	Rod	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

結果

P#
P1
P2
P3
P4
P5
P6

③ 結　合（Join）

結合は2つのリレーション間の共通定義域における値の比較演算子

$$\theta \in \{ <,\ \leq,\ =,\ >,\ \geq,\ \neq \}$$

に基づいている。

リレーションRの属性B_1とリレーションSの属性B_2のジョインは次の様に表わせる。

$$R\,[B_1\ \theta\ B_2]\ S$$
$$=\{\,rs \mid r\in R\wedge s\in S\wedge r\,[B_1]\ \theta\ s\,[B_2]\,\}$$

但し，rs はrとsの連鎖である。

θが"＝"である場合を等価ジョインという。また，等価ジョインの場合，ジョインをとった属性の片方だけを結果に残すnaturalジョインと呼ばれるものがある。

例えば，集合演算の例であるリレーションAとBについて次のようなジョインを考えてみる。

$A\,[a_1=b_1]\,B$

a_1	a_2	a_3	b_1	b_2	b_3
1	A	1	1	A	1
2	B	2	2	B	3
3	C	3	3	C	3

naturalジョイン

a_1	a_2	a_3	b_2	b_3
1	A	1	A	1
2	B	2	B	3
3	C	3	C	3

$A\,[a_1>b_1]\,B$

a_1	a_2	a_3	b_1	b_2	b_3
2	B	2	1	A	1
3	C	3	1	A	1
3	C	3	2	B	3

④ 半結合（semi － join）〔BERNP79〕

リレーションR1の属性B_1のR2の属性B_2によるθ-semi-joinは次の様に表わされる。

$$R_1<B_1\,\theta\,B_2\,]\ R_2 = (\,R_1\,[B_1\,\theta\,B_2]\ R_2\,)\ [A]$$
$$=\{\,r_1 \mid r_1\in R_1\wedge r_2\in R_2\wedge r_1\,[B_1]\ \theta\ r_2\,[B_2]\,\}$$

半結合の性質としては，次のものがある。

i） $R<B_1\,\theta\,B_2]\ S\subseteq R$

ii） $R\,[B_1\,\theta\,B_2]\ S=(\,R<B_1\,\theta\,B_2]\ S\,)\,[B_1\,\theta\,B_2]\ S$

iii） $R\,[B_1\,\theta\,B_2]\ S=R\,[B_1\,\theta\,B_2](S<B_1\,\theta\,B_2]\ R\,)$

i)は半結合がRのサイズを減少することを示し，ii)とiii)は半結合のJoinと同じことを示している。このことは，DDBSの結合に際して半結合を用いると転送コストを減少できることを示している。

⑤ 商（Division）

Divisionは，二項関係（被除数）と単項関係（除数）の間のオペレーションで，結果として単項関係をつくる。例えば，リレーションR（A，B_1）とS（B_2）でB_1とB_2は同じ定義域であるとすると，Divisionは次の様に表わされる。

$$R\,[B_1 \div B_2]\,S = \{\, r\,[A] \mid s \in S\,(S\,[B_2] = r\,[B_1]) \,\}$$

結果は，Aと同じ定義域で定義され，R〔A〕とR〔B_2〕の直積がRに含まれるようなR〔A〕になる。例えば，次の様な例になる。

R

A	B_1
p	1
p	2
q	1
r	1
r	2

S

B_2
1
2

R〔$B_1 \div B_2$〕S　結果

A
p
r

Divisionは全称作用素（universal quanitifier ∀）に対応している。

⑥ Renaming operator

リレーショナル計算言語の目標リストの機能（結果属性に対して名前を付ける機能）に対応するものが必要である。

3) 検索の例

関係代数を使用した検索の例として，部品－サプライヤデータベースについて考えて見る。

① 「部品P_2を供給するサプライヤに対するサプライヤ番号を求めよ。」という問合せは次のように表わす。

$$SP\,[P\# = '\,P_2\,'\,]\,[P\#]$$

関係代数のもとでリレーションは閉じた系であり，関係代数の結果がそれ自身リレーションである。よって再び結果を入力することが出来る（clousure）。関係代数は，各オペレーションの入力と出力との関係で結ばれた高水準な手続きを示している。この概念はデータフロー言語との関連を示している。

② 「赤い色の部品を少なくとも1つ供給できるサプライヤ番号を求めよ」という問合せは次の様に表わす。

$$(P\ [COLOR = 'RED']\ [P\# = P\#]\ SP)\ [S\#]$$

この意味は，最初にリレーションPの制限を行なう。次にその結果とリレーションSPとをP#でジョインをとった結果をS#について射影をとることを表わしている。

4) 格納演算〔TANAY80〕

リレーションへの組の挿入，変更の演算については，次のようにして行なうことが出来る。

① 挿　入

和演算を使用する。例えば，リレーションPに新しい組('P7', 'GREEN', 12, 'ATHENS')を挿入するには，次の様に表わす。

$$P \cup \{('P7', 'WASHER', 'GREEN', 12, 'ATHENS')\}$$

② 削　除

削除は差演算を使用する。例えば，リレーションSから組('S1', 'SMITH', 20, 'NEWYORY')を削除するには，次の様に表わす。

$$S - \{('S1', 'SMITH', 20, 'NEWYORK')\}$$

また，S#として'S1'をもつ組をすべて削除するには，次の様に表わす。

$$S - \{('S1', '?', '?', '?')\}$$

③ 変　更

組の変更は，差演算と和演算を連続して実行することによって行なう。

C. リレーショナル計算言語（Relational Calculas Language）

リレーショナル計算言語は第1階述語論理に基づいた言語である。これは，結合，制限演算を述語（predicate）として把え，述語からなる論理式によって必要な条件を表わし，射影演算を目標リスト（論理式の自由変数に相当し，更に属性のnamingの役目を持つ。）によって表わそうとするものである。リレーショナル計算言語は，述語論理に基づいているために非手続き的である。この例としてはALPHA〔CODDE70〕がある。

リレーショナル計算言語の詳細に入る前に第1階述語論理について触れておこう。

1) 第1階述語論理（first－order logic）

第1階述語論理は，数学の大部分と，日常英語の多くの叙述を表現できる一つの論理システムである。このシステムには，与えられた叙述の集合から，新しい叙述を正当，かつ論理的に演繹できる推論規則がある。

第1階述語論理で用いられる言語は，シンタクス（syntax）によって定義される。シンタクスを明確にするには，言語で用いられるアルファベット記号，および正しく言語を構成するように，これらの記号の配列の仕方を明確にしなければならない。述語論理の表現で，ある重要なク

ラスは完全論理式（well－formed formulus－wffs）と呼ばれるものである。

述語論理のシンタクス

i) Domain（定義域）

値の集合として定義される。

ii) Symbol（記号）：：＝

定数｜変数｜関数記号*｜述語記号**

論理記号（∧，∨，～，⇒，⇐）｜作用素（∃，∀）

iii) Term（項）：：＝

変数｜定数｜関数（f（t_1，……，tn）i.e. f；$D^n \rightarrow D$

但し，fはn－place関数　t_1は項（i＝1，……，n）

iv) Atom（原子論理式）：：＝P（t_1，……，tn）

$P : D^n \rightarrow \{T, F\}$

但し，Pはn－place述語，　t_1は項（i＝1，……，n）

もし，n＝0ならPは命題となる。

v) Literal（リテラル）：：＝P｜～P（Pの否定）

Pは原子論理式である。

vi) Clause（節）：：＝$L_1 \vee L_2 \vee \cdots\cdots \vee Ln$

Liはリテラルである。

vii) Well－formed formula（完全論理式）：：＝

原子論理式｜～A，A∨B，A∧B，A⇒B，A⇔B，if A，B＝wff｜

（Qx）A　where　Q＝作用素

A＝論理式

x＝Aの自由変数

変数xが自由変数であるというのは，論理式の中で，少なくとも1つのxのオカランスが自由であることを言う。

一階述語論理式として次のような例を考えてみる。定義域を整数の集合とし，関数f(x)とg(x)を次のように定義することにする。

f(x)＝　xの直前の整数

g(x)＝　xの直後の整数

＊　関数記号とは，例えば「x＋y」を表わすPLUS（x，y）とか「xの親を表わすFather(x)などの記号である。一般にn個のargumentを持つことができ，n－place関数記号と呼ばれる。

＊＊　述語記号とは，例えば「xはyより大きい」ということを表わすGREATER（x，y）のように定義される。つまり述語は1つの関係を表わしている。一般にn個のargumentをもつことができ，n－place述語記号と呼ばれる。

いま，変数x，yに原子論理式E（x，y）を次の様に定義する。

$$E(x, y) = \text{“}x = y\text{”}$$

「全ての整数は，ただ1つの直前の整数をもつ」というのを表わす一階述語論理式は次のように表わされる。

$$(\forall x)(\exists y)(E(y, f(x)) \wedge (\forall z(E(z, f(x)) \Rightarrow E(y, z))$$

2) 一階述語論理とリレーショナル計算言語の対応

表2.7　一階述語論理とリレーショナル計算言語の対応表

一階述語論理	リレーショナル計算言語
定義域	組の集合
記　号	定数 変数（組変数,） 関数 述語 論理結合記号（∧，∨，～，⇒，⇔） 作用素（∃，∀）
変数　束縛変数 自由変数	変数 変数（目標関数の内）
関　数	算術関数　e.g.　sin(x)，log(x) aggregate関数　e.g.　COUNT(x)
述　語	単項　range 述語 二項比較述語（結合，制限）

① 定義域＝組の集合

② 変数は，組と表わす変数である。ここでリレーションRの属性aがあるとする。xをRの組変数とすると，x.aは組xのaの値（i．e．x〔a〕)を表わすものである。

③ 述語として次の2種類がある。

㋑ 2項比較述語

㋺ range 述語

④ 2項比較述語としては次の2種がある。

㋑ 制限述語（restriction predicate）

㋺ 結合述語（join predicate）

一制限述語は次の形式である。

＜restriction－predicate＞::＝ x. a θ x. b | x. a θ v

ここで　$\theta \in \{<, \leq, =, \geq, >, \neq\}$

v は定数

a と b は x の range リレーション属性

制限述語の意味は次の様である。

$$x.a\,\theta\,x.b = \begin{cases} \text{true} & \underline{\text{if}}\ x[a]\,\theta\,x[b] \\ \text{false} & \underline{\text{otherwise}} \end{cases}$$

$$x.a\,\theta\,v = \begin{cases} \text{true} & \underline{\text{if}}\ x[a]\,\theta\,v \\ \text{false} & \underline{\text{otherwise}} \end{cases}$$

－結合述語は次の形式である。

＜join－predicate＞::＝ x. a θ y.b

a と b とは各々 x と y の属性

結合述語の意味は次の様である。

$$x.a\,\theta\,y.b = \begin{cases} \text{true} & \underline{\text{if}}\ x[a]\,\theta\,y[b] \\ \text{false} & \underline{\text{otherwise}} \end{cases}$$

⑤ proper range 述語は次の形式をしている。

＜range predicate＞::＝ R(x)

ここで　R はリレーション名

x は組変数

これは QUEL（〔HELDG 75〕）の range 文に対応している。

range 述語の意味は次の様である。

$$R(x) = \begin{cases} \text{true} & \underline{\text{if}}\ x \in R \\ \text{false} & \underline{\text{otherwise}} \end{cases}$$

⑥ 作用素

range coupled 作用素としては，次のものがある。

＜range－coupled－quantifier＞::＝

$(\forall\, x \in R)\, Q(x)$ |

$(\exists\, x \in R)\, Q(x)$

ここでは　R はリレーション

xは組変数

Qは論理式

$(\forall x \quad R)\,Q(x)$は$(\forall x)(R(x) \Rightarrow Q(x))$

$(\exists x \quad R)\,Q(x)$は$(\exists x)(R(x) \wedge Q(x))$

を意味している。

作用素は

$R \triangleq \{t_1, t_2, \cdots\cdots, t_n\}$とすると

$(\forall x \in R)\,Q(x) = Q(t_1) \wedge Q(t_2) \wedge \cdots\cdots \wedge Q(t_n)$

$(\exists x \in R)\,Q(x) = Q(t_1) \vee Q(t_2) \vee \cdots\cdots \vee Q(t_n)$

を意味している。

⑦ query は次の形式である。

$W = (t_1, t_2, \cdots\cdots, t_n) : Q$

ここでWは結果リレーション名

$(t_1, t_2, \cdots\cdots, t_n)$は目標リスト

但し、$t_i \neq t_j$ (for $i \neq j$) ($i, j = 1, 2, \cdots\cdots, k$)

t_iは組変数 (x)又は索引組変数 (x. a)

Qは qualification in a range separable wff

Qは次の様な形式である。

$Q = U_1(x_1) \wedge U_2(x_2) \wedge \cdots\cdots \wedge Un(x_n) \wedge V$

ここで Uiは proper range 述語

x_iはVの自由変数

Vは1) V内の全ての自由変数が $(x_1, \cdots\cdots, x_n)$、2) range述語を持たない、3) 全ての作用素はrange coupled されている論理式であることを満足する完全論理式である。

この問合せの結果リレーションは

$W = \{(t_1, t_2, \cdots\cdots, t_n) \mid Q = \text{true}\}$となる。

3) ALPHA 〔CODDE 70〕

ALPHA言語は第1階述語論理に基づいている。ALPHAはユーザとデータモデル間のコミュニケーションエリアとしてワークスペースを使用する。ALPHA言語にはデータモデルからデータを検索しワークスペース(W)に読み込むGET statemcnt とワークスペースの内容でデータモデルを更新するHOLD, UPDATE, DELETE, PUT statement がある。ワークスペースの内容は親言語によって操作される。

以下で，部品－サプライヤデータベースを使い，ALPHA言語の使い方を見ることにする。

① 検　索

検索には，データモデルからワークスペースにデータを読み込むGET statement がある。例によって使い方を見てみよう。

㋑ 「供給される部品のすべての部品番号を求めよ」という問合せは次のように表わされる。

GET W(S P. P#)

リレーション名　　属性名

これは，関係代数の射影と同じ演算になる。結果は，テーブルとしてワークスペースに与えられる。

㋺ 「ステータスが20以上でパリにあるサプライヤのサプライヤ番号を求めよ」という問合せは次のように表わす。

GET W(S. S#) : S. CITY = 'PARIS' ∧ S. STATU > 20

qualificationは，比較演算子 { <, ≤, =, ≥, >, ≠ } とBoolean 演算子 { ∧, ∨, ¬ } の任意の複合表現で表わされる。

㋩ 「部品P_2を供給するサプライヤのサプライヤ名を求めよ」という問合せは次のように表わせる。

RANGE SP X
GET W(S. SNAME) : ∃ X (X. S# = S. S# ∧ X. P# = 'P2')

"∃"は存在作用素と呼ばれ，記号の直後の変数が存在することを表わす。

㋥ 「部品P1を供給しないサプライヤのサプライヤ名を求めよ」という問合せは次のように表わせる。

RANGE SP SPX
GET W(S. SNAME) : ∀ SPX (SPX. S# ≠ S. S# ∨ SPX. P# = 'P1')

"∀"は全称作用素と呼ばれ，すべてに対して条件を満足されることを表わす。

㋭ 「すべての部品を供給するサプライヤのサプライヤ名を求めよ」という問合せは次のように表わされる。

RANGE P PX
RANGE SP SPX
GET W(S. SNAME):∀PX ∃SPX(SPX. S# = S. S# ∧ SPX. P# = PX P#)

この場合のように，存在作用素と全称作用素を一緒に使用する場合，順番に重要な意味があるので注意する必要がある。

② 格納演算

格納演算は，ワークスペースの内容をデータモデルに格納したり，変更したり，削除したりする。同時には1つのリレーションに対してのみ更新することができる。

例えば，「P2部品の色をYELLOWに変更しろ」という問合せは次のように表わされる。

```
HOLD W ( P. P#, P. COLOR ) : P. P# = 'P2'
  W. COLOR = 'YELLOW' （親言語）
UPDATE  W
```

上記の様に，検索のGETと同じ操作をするHOLDステートメントで，DBMSにデータを変更する意志を伝える。実際の変更は親言語によって実行される。

削除も削除するタプルをHOLDステートメントでワークスペースに読み込みDELETEステートメントで削除する。

挿入はワークスペースに親言語で作ったタプルをPUTステートメントでデータモデルに格納する。

2.2.7 view

viewとは，データベースに対するユーザの見方を定義する外部スキーマである。viewは，データベースのsnapshotを定義しており，データベースに対する窓をユーザに与えるものである。

viewとして定義されるリレーションはベースリレーション（概念レベルのリレーション）より導出されるリレーションである。

viewに対する検索は，問合せ変形（quey modification）手法によって，ベースリレーションに対する検索に変換できる（7章を参照）。しかし，viewに対する更新を，自動的にベースリレーションに変換することは，ごく限定された場合においてのみ可能である。

この更新問題は，次の2点に依っているものと思われる。

(1) viewリレーションをベースリレーションと同じレベルのリレーションと考えていて，概念レベルのリレーションに対するオペレーションを何等の制限をすることなく，そのまま使わせている。

(2) リレーショナルモデルは，実世界のデータの意味とは独立である。このため，更新が

i) 実世界のオブジェクト実体の生成，消滅か

ii) オブジェクトの特徴の変化なのか

を明確に区別できない。

この問題解決のために次の2点が必要となろう。

1) viewをユーザが定義する時，ユーザは既にこのviewの使い方を想定している。従って，

viewリレーションに対して，ベースリレーションに対するものと同じオペレーションの無制限な適用は許されないはずである。

このことにより，プログラミング言語におけるデータ抽象化技法を用いることが必要である。

即ち，

a) viewリレーションをベースリレーションより定義する。

b) viewリレーションに対して許されるオペレーションを定義する。

c) viewオペレーションを，ベースリレーションに対するオペレーションによって記述する。

2) リレーショナルモデルに，オブジェクトの概念を導入する。これによって，

i) オブジェクト自体の生成，消滅と

ii) オブジェクトの特徴の変化

との明確な分離を行なう。

上記の試みに付記1のオブジェクトモデルアプローチがある。

2.3 CODASYL DBTG モデル

CODASYL DBTGモデル（以降，単に「DBTGモデル」と呼ぶ）は，事象集合を表わすレコード型と事象集合間の関係性を表わすセット型とリンクレコード型が存在するネットワーク型モデルである。本章では，DBTGモデルについて論じる。

2.3.1で，DBTGモデル概念としてレコード型とセット型に対してフォーマルな定義を試み，2.3.2では内部スキーマレベルについて，2.3.3ではDML言語について，2.3.4で外部スキーマについて述べることにする。

2.3.1 DBTG モデル概念

本節では，DBTGモデルのフォーマルな定義を試みる。

A．事象集合（レコード型）

DBTGモデルにおいて，事象集合はレコード型によって表わされる。

ここでViを値の集合とすると，レコード型Rは次の様に定義できる。

$$R \subseteq V_1 \times V_2 \times \cdots\cdots \times V_n \qquad \cdots\cdots(1)$$

Rの要素 $r \in R$ は，レコードオカランス（record occurrence）と呼ばれる。rは次の様に表わされる。

$$r = (v_1, v_2, \cdots\cdots, v_n) \qquad \cdots\cdots(2)$$

ここで $v_i \in V_i$ for $i = 1, \cdots\cdots, n$

項目（data item）t_iは，レコード型Rから値集合V_iへの写像であり，次の様に表わされる。

$$t_i : R \rightarrow V_i \qquad \cdots\cdots(3)$$

ここで，$r(\in R) = (v_1, \cdots\cdots, v_i, \cdots\cdots, v_n)$に対して，$r[t_i] = v_i$とする。Repeating groupについては，1つの項目として考え，その処理は親言語によって扱うものとする。

DBTGモデルの特徴は，レコード型内の各レコードオカランスが，オカランスの持つ値によっては識別されない（self－identifiedでないことである）。これはリレーショナルモデルとの主要な相違である。各オカランスは自分の位置によって識別される。具体的にはDB－key（データベースキー）と呼ばれる値によって暗黙のうちに識別される。

従って，r_1，$r_2 \in R$が$r_1 = r_2$となることがあり得る。即ち，1つのレコード型内に，同一の値のオカランスが複数存在し得てしまう。

DB－keyの他にレコード型内のオカランスを一意に識別出来るデータ項目としては，次の2つがある。

1) DNA（Duplicates Are Not Allowed）指定の付いているCALC項目（又は項目の集合）。

2) 単一セット型（Singular set－type）によって，DNA指定されたデータ項目（又は項目の集合）。

この2つをリレーショナルモデルのキー概念に対応させることができる。

レコード型のスキームは，次の様に書ける。

$$\underline{R}(t_1, t_2, \cdots\cdots, t_m) \qquad \cdots\cdots(4)$$

ここでRはレコード型の名前，t_1，t_2，……，t_mはそれを構成するデータ項目名である。

B・関係性集合（セット型）

DBTGモデルにおいて関係性を表わすモデル要素としては，次の2つのものがある。

a) セット型

b) リンクレコード型

セット型は，2つのレコード型間の1対nの関係性を表わし，リンクレコード型は，一般に複数レコード型間のn対mの関係性を表わしている。

1) 1対nの関係性

2つのレコード型R_1とR_2との間の1対nの関係性はセット型Sによって表わされる。Sは，R_1とR_2との直積の部分集合である。

$$S \subseteq R_1 \times R_2 \qquad \cdots\cdots(5)$$

$$s(\in S) = (r_1,\ r_2) \qquad \cdots\cdots(6)$$

ここで　$r_1 \in R_1$,　$r_2 \in R_2$

従って，セット型は$\underline{S}$（R_1,　R_2）と表わせる。これを，DBTGデータ構造図（又はBachman図式）で表わすと図2.7の様になる。

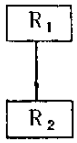


図2.7 セット型

R_1は，セット型Sの親レコード型（owner record type），R_2は，子レコード型（member record type）と呼ばれる。

セット型は，親レコード型の1つのレコードオカランスと，子セット型のn（≧0）個のレコードオカランスとが関係していることを示している。即ち，セット型は，親と子レコード型間の1対nの関係性を表わしている。逆に，セット型は，子レコード型の1つのレコードオカランスが多くとも1つの親レコードオカランスを持っていることを表わしている。

セット型Sのあるセットオカランス SOは，次の様に表わされる。

$$SO \triangleq \{\, r_1 \{\, r_{2i} \mid (r_1, r_{2i}) \in S \,\} \,\} \qquad \cdots(7)$$

ここで　$r_1 \in R_1$,　$r_{2i} \in R_2$　（$i = 1, \cdots\cdots, n$）

特に，次の様なセット型を単一セット（singular set）と呼ぶ。

$$S \subseteq R_1 \times R_2 \qquad \cdots\cdots(8)$$

ここで　$R_1 = \{ r_1 \}$　i. e.　$|R_1| = 1$　（$|R_1|$は，R_1のカーディナリティ）

R_1はシステムレコード型と呼ばれる。

即ち，セット型Sは1つのセットオカランスSOをもっている。

$$SO = \{\, r_1,\ \{\, r_2 \mid \forall r_2 \in R_2 \,\} \,\}$$

セットオカランスSOは，順序集合である。

$$\left.\begin{array}{l} SO = \{\, r_1,\ r_2,\ \cdots\cdots, r_{2n_2} \,\} \\ \text{ここで}\quad r_{21} \prec r_{22} \prec \cdots\cdots \prec r_{2n_2} \\ \text{（昇順）}\quad r_{2i} \prec r_{2j} \quad \text{if} \quad i \le j \wedge r_{2i}[t_k] \le r_{2j}[t_k] \\ \text{（降順）}\quad r_{2i} \prec r_{2j} \quad \text{if} \quad i \le j \wedge r_{2i}[t_k] \ge r_{2j}[t_k] \end{array}\right\} \qquad \cdots\cdots(9)$$

R_2の項目 tkの値によって降順又は昇順に順序づけがなされている。

2）制　約

セット型S（R_1,　R_2）について考える。ここで次の関数 fsを導入する。

$$fs : R_2 \to R_1$$

$$\forall r_2 \in R_2 \quad fs(r_2) \begin{cases} = r_1 \ \underline{\text{if}}\ (r_1,\ r_2) \in S \\ = \emptyset \ \ \underline{\text{otherwise}} \end{cases} \qquad \cdots\cdots(10)$$

同様に fs の逆関数 $fs^{-1}(r_1)$ $\begin{cases} = \{ r_2 \mid (r_1, r_2) \in S \} \\ = \emptyset \quad \underline{if}\ (r_1, r_2) \notin S \end{cases}$

この関数は〔ABRIJ74〕の記法を用いて，図3.2の様に書く。

図2.8

即ち

$$\forall r_2 \in R_2 \quad 0 \leq | fs(r_2) | \leq 1$$
$$\forall r_1 \in R_1 \quad 0 \leq | fs^{-1}(r_1) | \leq n \quad (n \geq 0) \qquad \cdots\cdots(11)$$

DBTGモデルでは，セット型のモデル制約として，次のものがある。

i) $\forall r_2 \in R_2 \quad | fs(r_2) | = 1$ 即ち fs は全関数である。

ii) $\forall r_2 \in R_2 \quad 0 \leq | fs(r_2) | \leq 1$ 即ち fs は半関数である。

i) は全ての子レコードオカランスは必ず親レコードオカランスを持たなければならないことを示している。一方，ii) は必ずしも子レコードオカランスが親レコードオカランスをもたなくてもよいことを示している。i) のタイプのセット型をTタイプ，ii) のタイプのセット型をPタイプと呼ぶことにする。

この制約は，DBTGモデルのメンバシップクラスとして表わされ，更新に対するインテグリティ保持機構となっている。Tタイプの $\underline{S}(R_1, R_2)$ では，次の更新規則が成り立っている。

$$- \underline{delete}\ \ r_1 \in R_1 \ \rightarrow\ \underline{delete}\ ^{\forall} r_2 \in R_2 \quad s.t. \quad (r_1, r_2) \in S$$

これは親レコードオカランスを削除する場合，その親レコードオカランスの全ての子レコードオカランスも削除することを示している（MANDATORY と FIXED.）。

$$- \exists r_1 \in R_1 \ \ s.t.(r_1, r_2) \in S \rightarrow \underline{insert}\ r_2 \in R_2$$

これは子レコードオカランスを挿入する場合に，その親レコードオカランスが存在していなければならないことを示す（AUTOMATIC）。

$$- \underline{delete}\ s \in S \rightarrow \underline{delete}\ \ r_2 \in R_2 \quad s.t.(r_1, r_2) \in S$$

セットオカランスを削除する場合，そのセットオカランスの全ての子レコードオカランスも削除される（MANDATORY と FIXED）。

Pタイプに対しては，任意のオペレーションを，R_1 と R_2 との関連なしに適用できる（MANUAL と OPTIONAL）。

DBTGモデルのメンバシップの概念との対応について考えてみる。メンバシップ には，次の2つのクラスがある。

メンバシップクラス ｛ 消去クラス（ removal　class ）
　　格納クラス（ storage　class ）

消去クラスは，消去演算に対するインテグリティ制約を与えており，次の2つのタイプがある。*

｛ MANDATORY　$fs(r_2) = r_1$という関係の除去と，$r_2 \in R$の除去を分けて行なえない。
　 OPTIONAL　分けて行なえる。

格納クラスは，オカランス又は関係性の新たな追加に関する制約を与えており，次の2つのタイプがある。

｛ AUTOMATIC　新たな$r_2 \in R_2$の生成と，$(r_1, r_2) \in S$の生成とを分けて行なえない。
　 MANUAL　分けて行なえる。

これらのメンバーシップの組み合せと全関数(T)と半関数(P)との関係は表2.8の様になる。

表2.8

	消去クラス	格納クラス
MANDATORY／AUTOMATIC	全関数	全関数
MANDATORY／MANUAL	全関数	半関数
OPTIONAL／AUTOMATIC	半関数	全関数
OPTIONAL／MANUAL	半関数	半関数

3）n対mの関係性

レコード型間のn対mの関係性を表わすために，〔 CODAS73 〕ではリンクレコード型 を用いる。リンクレコード型Lは，次の様に定義出来る。

$$L \subseteq R_1 \times R_2 \times \cdots\cdots \times Rn \quad \cdots\cdots (2)$$

ここで　Rj 含 レコード型（ j＝1，……，n ）

＊　DDL78では，さらにFIXEDタイプが加えられている。
削除演算に対してはMANDATORYと同じ制約をうけるが，セットオカランスの変更が禁止されている。

リンクレコード型のオカランス $l \in L$ は，次の様に定義される。

$$l \in \{ r_1, r_2, \cdots\cdots, r_n \}$$

ここで　$r_j \in Rj$　($j = 1, \cdots\cdots, n$)

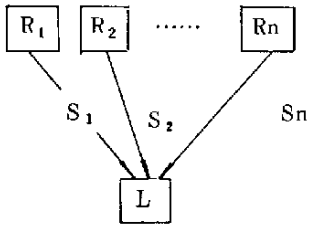


図2.9　リンクレコード型L

DBTGデータ構造図によってリンクレコード型を表わすと，図2.3.3の様になる。Lはリンクレコード型である。R_1，R_2，……，R_n は，Lを子レコード型とするセット型 S_1，S_2，……，S_n の各々の親レコード型となる。Lと各レコード型Riとは，1対nのセット型Siによってリンクされ次の様なセット型Siのスキームになる。

$$\underline{Si}\ (R_j,\ L) \qquad \cdots\cdots(13)$$

この時，S_i はTタイプでなければならない。即ち，リンクレコード型Lの全てのオカランス l は必ず関連するセットの親オカランスをもたなければならない。

Lは，レコード型であるので，項目を持つことができる。

即ち，

$$L \subseteq V_1 \times V_2 \times \cdots\cdots \times Vn \qquad \cdots\cdots(14)$$

項目 t_i は次の様に定義される。

$$t_i : L \rightarrow V_i$$

よってLのスキームは次の様に表わされる。

$$\underline{L}\ (R_1, R_2, \cdots\cdots, R_n,\ t_1, t_2, \cdots\cdots t_n) \qquad \cdots\cdots(15)$$

関連性として，同一レコード型内のオカランス間に存在するものがある。

即ち，

$$L \subseteq R \times R \qquad \cdots\cdots(16)$$

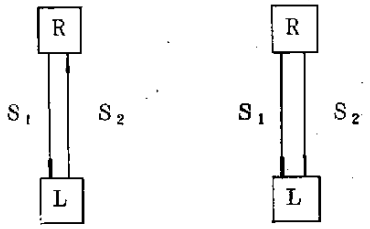
$l (\in L) \ni (r,\ r')$

ここで　$r \in R$,　$r' \in R$

$\underline{L}\ (R,\ R)$ には，次の2つのタイプがある。

a)　1 : n　　〔図2.10　a)〕

b)　n : m　　〔図2.10　b)〕

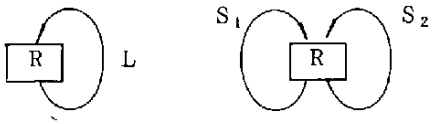


a) 1 : n　　b) n : m

図2.10　リンクレコード型L

DDL78ではこれはセット型として表わせる。

即ち，S（R，R）



a) 1 : n　　b) n : m

図2.11　図2.10に対するDDL78表現

2.3.2　内部スキーマレベル

DBTGのDDLによる内部スキーマ記述では，a) 直接アクセスとb) 逐次アクセスの両方をサポートしている。以下に，DBTGモデルにおける直接アクセス法と逐次アクセス法について述べる。

① 直接アクセス方法

DBTGモデルにおける直接アクセス方法の指定方法には次の2つがある。

i) 主インデクス：$v \rightarrow t$

CALC項目又は集合でDNAのもの。すなわち，インデクスの値に対応する組（tuple）が唯一である。

ii) 2次インデクス：$v \rightarrow \{ t_1, t_2, \cdots\cdots, t_n \}$

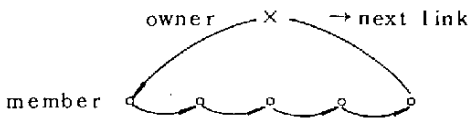
CALC項目又は集合でDAのもの。すなわち，インデクスに対応する組が複数ある場合である。

② 逐次アクセス

DBTGモデルにおける逐次アクセス方法の指定方法には次の3つがある。

NEXT，PRIOR，OWNERリンクによる論理的順序によるアクセスパスの指定方法がある。

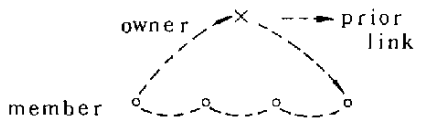
－NEXT LINK



1つのセットオカランスのシーケンスに対応したリンク（ポインタ）をもたせる。

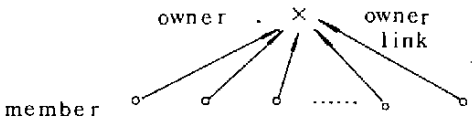
DBTG DMLのFIND NEXT ……コマンドはこのリンクに従って，現在のオカランスの次のオカランスをアクセスする。

－PRIOR LINK



PRIORリンクは，NEXTリンクの逆方向につくられる。このリンクによって，現在の直前のオカランスに直接もどる。

－OWNER LINK



OWNERリンクは，子レコードオカランスから，現在のセットオカランスの親レコードオカランスのリンクである。これによって，FIND OWNER…… コマンドは，直接にOWNERリンクをたどることによって実行できる。

セットオカランスは，この中の子レコードオカランスに対してある順序を与えている。この順序としては，次の2つがある。

1）ある属性値に対する順序（降順，昇順）

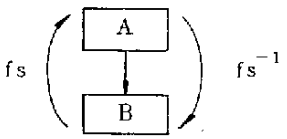
2）システムによる任意の順序

さらに，セットオカランス内で，ある属性値がuniqueである（DNA）であることも指定できる。セット型として単一セット型を用いると，レコード型の全オカランスに対しても順序づけできる。この順序はアクセスの最適化において重要である。

オカランスの格納方法も重要である。

○ クラスタリング

$$\begin{cases} a \in A \text{ と } fs^{-1}(a) = \{ b_1, \cdots\cdots, bn \} \Rightarrow loc(a) = loc(b_1) \\ b \in B \text{ と } fs(b) = a \Rightarrow loc(b) = loc(a) \end{cases}$$



クラスタリングは互いにアクセスされあう頻度の高いデータを物理的に近接に配置しておくことにより，アクセス効率を向上させることである。

2.3.3 言　語

DBTG DML は，次の様なDML コマンドによっての検索，更新，削除等を行なう。

FIND
GET
MODIFY
CONNECT
DISCONNECT
ERASE
STORE

これらのDML コマンドは，親言語（COBOL，PL／I）に埋め込んで使用する。まず最初に，これらのDML コマンドを使用するのに重要な意味をもつcurrency の概念について述べ，各コマンドの詳細について述べることにする。

A．Currency（現在子）

DBTGモデルの言語の特徴としてnavigational であることから，セットオカランスのどこに現在レコードオカランスがあるのか重要な意味を持っている。

そこで，各プログラムが動作している間，DBMSは'currency status indicator'のテーブルを維持している。indicator は，最も最近にアクセスされたレコードオカランスを表わすデータベースキーの値である。currency status indicatorは，各レルム，各レコード型，各セット型，レコード型に対して維持される。

1）各レルムに対するcurrency status indicator

レルムRに対して，"current of R"又は"current of realm R"は，Rの中で最も最近にアクセスされたレコードオカランスである。

2）各レコード型に対するcurrency status indicator

レコードT型に対して，"current of T"又は"current T occurrence"は最も最近アクセスされたTオカランスである。

3）各セット型に対するcurrency status indicator

セット型Sに対しては，Sの最も最近にアクセスされたレコードオカランスはオーナもしくはメンバーのオカランスである。これを，"current of S"と呼ぶ。

また最も最近にアクセスされたセットオカランス（親と子オカランスの集合）は，"current S occurrence"と呼ばれる。

4）レコード型に対するcurrency status indicator

'current of run － unit'はタイプに関係なく，最も最近にアクセスされたレコードオカランスである。

5）currency とDMLコマンドとの関係

currency と DML コマンドの間には次のような関係がある。

① FIND 既存のレコードオカランスに位置づけ，それを run － unit の current として定める。

② GET run － unit の current を retrieve する。

③ MODIFY run － unit の current を更新する。

④ CONNECT 1つもしくは多数のセットオカランスに run － unit の current を挿入する。

⑤ DISCONNECT 1つもしくは多数のセットオカランスから run － unit の current を切り離す。

⑥ ERASE run － unit の current を削除する。

⑦ STORE 新しいレコードオカランスを生成し run － unit の current として定める。

①～⑦のコマンドの中で，FINDはSTOREを除く他のコマンド以前に常に要求されるコマンドであることがわかる。すなわち，FINDが重要な役割を果すことがわかる。

B. DML (Data Manipulation Language)

ここでは，データベース上のデータがユーザワークエリア（UWA）(COBOLでは record area に該当する）に読み込まれ，DMLによってオペレーションされる。DMLは親言語に埋め込んで使用されるので親言語と互換性をもっていなければならない。以下に主なDML コマンドについて，サプライヤー部品データベース（図2.12）を例として使用方法とみることにする。

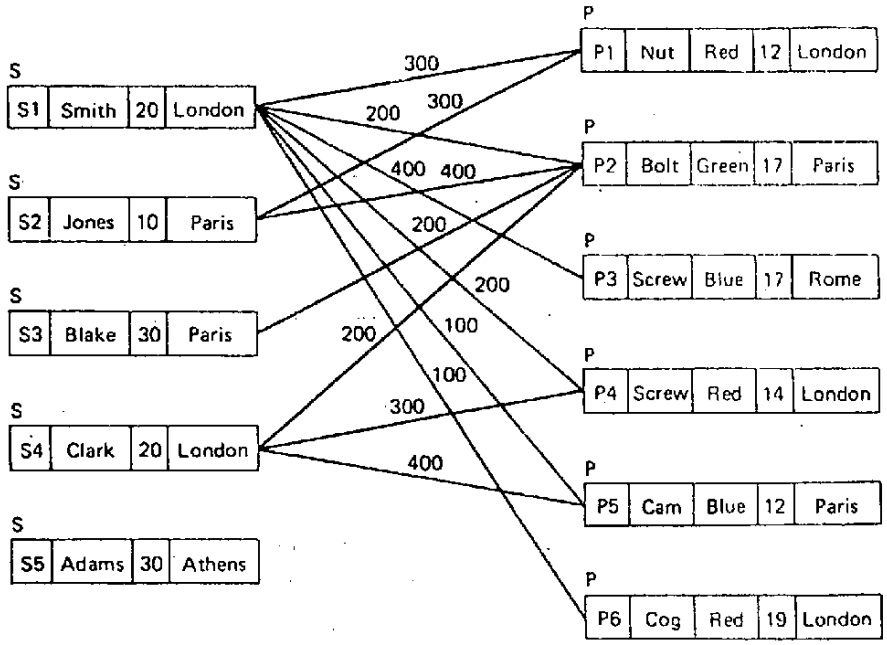


図2.12 サプライヤー部品データベース

1) FIND コマンド

FINDは，DBTGモデルのアクセスパスを用いてレコードオカランスの位置決めを行なう。

DBTGモデルのアクセスパスとしては，次のものがある。

① 直接アクセス

i) DB－key（データベース　キー）	FIND　DB－KEY
ii) CALC キー with DNA	FIND　ANY
iii) CALC キー with DA	FIND　DUPUCATE

② navigational アクセス

i) next，prior リンク	FIND { NEXT / PRIOR } WITHIN
ii) owner リンク	FIND　OWNER　WITHIN
iii) メンバーレコードの項目 with DA	FIND　DUPLICATE
iv) UWAのレコードの項目	FIND　CURRENT　USING

この他にFINDコマンドには，run－unitのカレントとしてセットのカレントを定めるFIND CURRENT ― WITHIN ― というコマンドがある。

次にFINDコマンドの例として次の様なものを考えてみる。

a) 直接アクセスの例

レコードSPのデータ項目SNOがCALC項目であるとき，サプライヤS4に対する全SPオカランスを直接アクセスで見つけるのは次の様に表わせる。

```
    MOVE 'S4' TO SNO IN SP.
NXT.
    FIND DUPLICATE SP.
    IF NOTFOUND = 'YES' GO TO QUIT.
    (Get SP and process it)
    GO TO NXT.
```

b) navigational アクセスの例

サプライヤ'S4'によって供給される部品番号を求めるには，次の様にDMLをつくる。

```
    MOVE 'S4' TO SNO IN S.
    FIND ANY S.
    IF S－SP EMPTY  GO TO NON－SUPPLIED.
NXT.
    FIND NEXT SP WITHIN S－SP.
    IF END SET = 'YES' GO TO ALL－FOUND.
    GET SP.
    (add PNO IN SP TO result list)
```

GO TO NXT.

2) GET コマンド

GETは，FINDで位置づけられたレコードオカランスをUWA内のレコード型のロケーションにGETする。

例えば，サプライヤS4の全情報を得たい場合，次のように書く。

```
MOVE 'S4' TO SNO IN S.
FIND ANY S.
GET S.
```

3) MODIFY コマンド

MODIFYはレコード型の項目の変更及びメンバレコードをあるセットオカランスから別のセットオカランスへ接続させるオペレーションを行なう。

① データ項目の内容変更

MODIFY { レコード名 / データ項目名 IN レコード名

この場合，UWA内の値でrun－unitのcurrentを置き換える。

例えば，「サプライヤS4のステータス値に10を加える」という変更は次のようにDMLを作成する。

```
MOVE 'S4' TO SNOIN S.
FIND ANY S.
GET S.
ADD 10 TO STATUS IN S.
MODIFY S.
```

次の2つの書き方の場合，メンバーシップとしてFIXEDでないことが条件となる。

② セットオカランスの変更

MODIFY レコード名 ONLY セット名 MEMBERSHIP.

この場合，run－unitのcurrentを別のセットオカランスに変更させる。

③ データ項目の変更とカセットオカランスの変更

MODIFY レコード名 INCLUDING セット名 MEMBERSHIP.

この場合，run－unitのcurrentのデータ項目の内容を変更するとともに，別のセットオカランスに変更する。

例えば，〔サプライヤS4によって供給されている部品S2の量（200）をサプライヤS3が代って供給する」という変更の場合，DMLは次のように書く。

```
MOVE 'S4' TO SNO IN S.
FIND ANY S.
FIND FIRST SP WITHIN S－SP.
```

```
MOVE 'S3' TO SNO IN S.
MOVE 'S3' TO SNO IN SP.
MODIFY SP INCLUDING S-SP MEMBERSHIP.
```

4) CONNECT コマンド

CONNECTは，あるセットオカランスにメンバレコードとして run－unit の current オカランスを挿入する。この場合，メンバシップはOPTIONALでなければならない。

まず，挿入しようとするセットオカランスに位置づける。次に挿入するレコードオカランスを run－unit の current オカランスにする。そこで次のようなCONNECT コマンドをつくる。

```
CONNECT レコード名 TO セット名.
```

E. DISCONNECT statement

DISCONNECT statement は，run－unit の current をそれを含んでいる セットオカランスから切り離す。但し，この場合切り離すレコードは，OPTIONALメンバー でなければならない。切り離されたレコードオカランスは，データベース上に存在するが，セットオカランスにははいっていない。

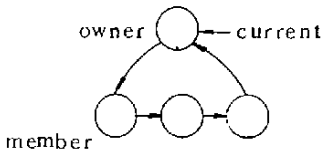
```
DISCONNECT レコード名 FROM セット名.
```

F. ERASE statement

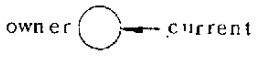
ERASE statement は，run－unit の current のオカランスをデータベース上から削除する。ERASEには次のような4つの形式がある。

(1) ERASE レコード名

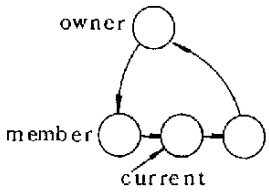
この場合，からでないセットオカランスのオーフでない run－unitの current を削除する。



i) current＝owner
この場合，削除されない



ii) current＝owner
メンバレコードオカランスがないので，削除されない。

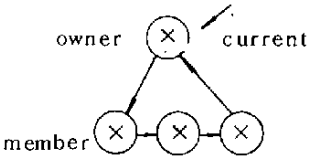


iii) current＝menber
常に削除される。

(2) ERASE レコード名 PERMANENT.

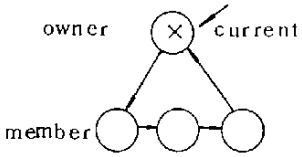
この場合，run－unit のカレントをオーナとするセットオカランスがあれば，そのメン

バーオカランスがMANDATORY（or　FIXED）であるものと共に run－unit のカレントは削除される。



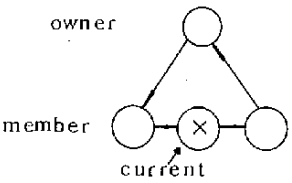
(MANDATORY)

i) current = owner & membership = MANDATORY このセットオカランスにあるすべてのレコードオカランスは削除される。



(OPTIONAL)

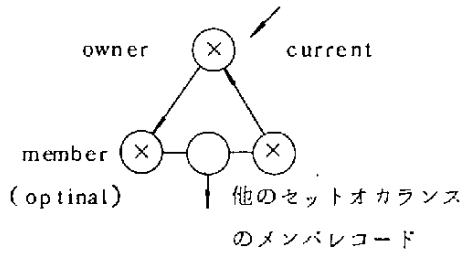
ii) current = owner & membership = OPTIONAL current であるオーナレコードオカランスのみ削除される。



iii) current = member current であるメンバレコードオカランスのみ削除される。

(3)　ERASE　レコード名　SELECTIVE

この場合，(2)とほとんど同じであるが，MembershipがOPTIONALの場合，他のセットオカランスの子になっていないレコードオカランスは削除する。



(4)　ERASE　レコード名　ALL

この場合，run－unit の current のオカランスが削除され，もしそのレコードオカランスがセットオカランスのオーナであれば，すべてのメンバレコードオカランスも削除される。

2.3.4　外部スキーマレベル（サブスキーマ）

DBTG モデルの外部モデルは，与えられたスキーマ（概念スキーマ）に対する部分集合としてサブスキーマを定義することができる。

DBTG モデルの外部モデルは，リレーショナルに較べて自由度がない。サブスキーマはスキーマの部分集合であるために次のような定義をすることができない。

①　1つ以上にまたがるエリアの宣言

②　1つ以上にまたがるセットの宣言

③　1つ以上にまたがるレコードの宣言

④　1つ以上にまたがるデータ項目の宣言

また，サブスキーマには自己矛盾があってはいけない。例えば，セット宣言で参照している場合，レコード宣言を省略することができない。

スキーマとサブスキーマの間には次のような相違がある。

①　エリア，セット，レコード，データ項目に対して個人的な名前をつけることができる。

②　データ項目に対して異なったデータタイプを与えることができる。

③　レコード内のデータ項目の相対的順序を変えることができる。

④　セットに対してスキーマとは異なるSET　SELECTIONを与えることができる。

概念スキーマの変更のうち，既存のレコード型に新しいデータ項目を追加することや新しいレコード型あるいはエリアを追加することや新しいセット型を追加することによりサブスキーマを変更することは必要ない。

サブスキーマの概念の重要性は，スキーマの全てをユーザに見せないことにより，Securityを守ることおよびユーザが使い易くなることにある。

2.4 内部スキーマモデル－SDDL

概念／外部レベルではリレーショナル等のモデル化が進められている。しかし，内部スキーマレベルでは，フォーマルなモデル化の試みはあまりない。

内部スキーマレベルのモデル化は，概念－内部マッピングにおいてOptimization（最適化）や設計に関して必要である。また完全なデータ独立性を実現するためにも必要である。

今までの試みとしては次の様なものがある。

1) CODASYLのDSDL〔 CODAS 78 〕
2) Wong, E.〔 WONGE 79 〕－Access path model
3) Yao, S.B.〔 YAO S76 〕－Attribute-based model

ここでは，CODASYLのDSDLを発展させたSDDL（Stored Data Definition Language）について述べることにする。

2.4.1 SDDL（Stored Data Definition Language）

CODASYL では，データ構造の記述と変換のためのモデルと言語に関するタスクグループ（SDDTTG, Stored Data Definition and Translation Task Group ）を設けている。このタスクグループのレポート〔 CODAS 77 〕で述べられたデータ構造の記述のモデルについて述べることにする。

データ構造には，論理的様相と物理的様相がある。DBTGモデルでは，この区別が明らかでなくデータ独立性が達成されないという欠点がある〔 CODAS 73 〕。データ構造を論理的構造と物理的構造に明確に区別して記述することをこのタスクグループは試みている。

このデータ構造を記述するモデルとしてDIAM（Data Independent Access Model）のStringモデル〔 SENKM 73 〕を用いている。

この理由としては，このモデルが

(1) 全体的なデータの明確な定義
(2) 一般化されたデータのtranslator
(3) データの相互関係

を与えることができるためである。

DIAMモデルは，次の4つのモデルから成っている。

1) 論理（事象）モデル
2) Stringモデル
3) Encodingモデル
4) 物理装置モデル

論理モデルでは，情報を記録しようとする実世界の概念である事象（属性名と属性値の組の集合）が扱われる。String モデルでは，属性値や属性値の集合の結合を単方向オペレータであるStringによってアクセスパスを表わしている。Encodingモデルは，物理構造に対してlinear

storageにおけるデータの格納レイアウトや物理構造をモデル化したものである。物理装置モデルは，物理的な格納装置の特徴や様相の仕様を表わすモデルである。

SDDLでは，Stringモデルと Encodingモデルを使い，データ構造の一般モデル化を試みている。SDDLではデータ構造を論理構造と物理構造に明確に分離して記述する。

これは内部スキーマのモデル化として，データ処理の performanceを高めるための アクセスパスのモデル化を目的としている。アクセスパスのモデルとしてのアクセス構造として次の2つがある。

1) 論理アクセス構造

2) 物理アクセス構造

論理アクセス構造（Logical Access Structure，LAS) は，データの構成（項目とグループ）とデータ構成間の相互関係とを定義したものである。相互関係とは，データがどのように実現されているかに無関係に，データに対する全アクセスパスを記述したものである。

物理アクセス構造（Physical Access Structure，PAS）は，格納媒体の物理構造である。ここでは格納媒体の格納位置や，これに対する物理的なアクセスパスが記述される。

この様に論理アクセス構造と物理アクセス構造とを明確に分離した理由は，物理構造に影響を与えるようなデータ構造の変更が，論理構造記述の変更を必要としなくするためである。

この2つの記述モデルであるStringモデルについてまず説明し，その後に論理アクセス構造と物理アクセス構造について述べることにする。

2.4.2 Stringモデル

Stringモデルとは，属性値や属性値の集合を相互に結合する単方向オペレータであるStringによってアクセスパスを表わすモデルである。

このStringモデルによって論理アクセス構造と物理アクセス構造が記述される。String モデルは次のような構成要素からなる。

i) 要素（Senkoの属性に対応）

非可分の構成要素

ii) グループ（リレーショナルモデルのリレーション，DBTGモデルのレコードに対応）

要素の集合であり，要素間の結合はStringによって表わされる。

このグループ上に2つのタイプの gathering mechanismがある。

iii) collection関係（1つのグループからなるcollectionはファイルに対応）

要素，グループ，collectionもしくはaggregate関係上の同種の gathering mechanism である。要素や要素の集合間の結合はStringによって表わされる。

要素の値によって順序づけられた結合を表わすことができ，これは順ファイルやRepeating group に対応する。

iv) aggregate関係（DBTGモデルのセットに対応）

要素，グループ，他の collection，aggregate 関係上の同種でない gathering mechanism である。要素や要素間の結合は String によって表わされる。

これらの構成要素間の関係性を図示すると，図 2.13 の様になる。

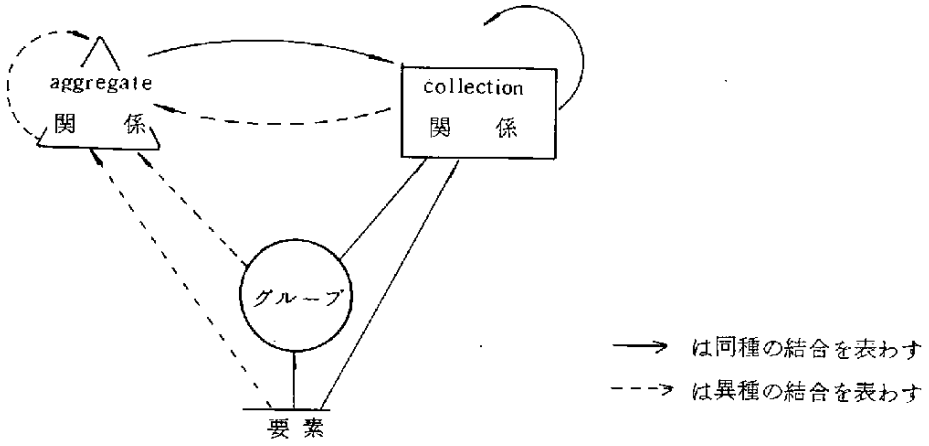
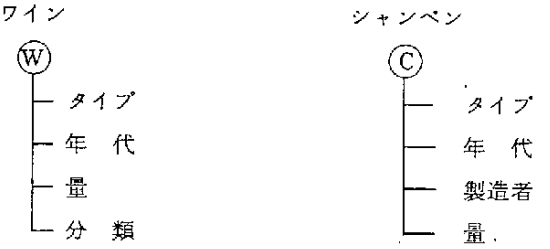


図 2.13　構成要素間の関係図

これらの構成要素を次のような記号で表わすとする。

いま，ワイン工場のモデル化の例を考えて見る。次のようなグループを定義する。



タイプ，年代，量，分類，製造者は各々要素である。ワイングループは要素（タイプ，年代，量，分類）から成るグループである。シャンペングループは要素（タイプ，年代，製造者，量）から成るグループである。

この２つのグループを使って collection 関係と aggregate 関係の例についてみてみよう。

1)　collection 関係

①　赤ワイン（年代順）

ワイングループ上にRWというcollectionを定義する。RW collection を分類が赤であるという同種のワイングループの集合で，年代の昇順に並べるものと定義する。赤ワインのスキーマとその実体は図2.14の様に示される。

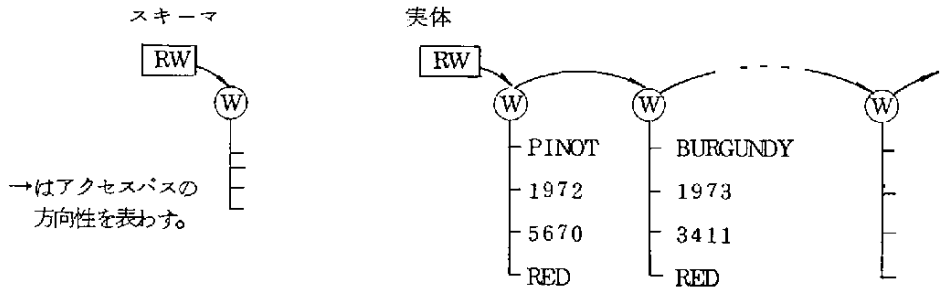


図2.14　赤ワインのスキーマと実体

② 種々のシャンペン（タイプ順）

2つのcollectionを定義する。CT collction をシャンペングループで同種のタイプの集合で，年代の昇順に並べたものと定義する。またCH collectionは，CT collection上に定義し，タイプの昇順（PINK, WHITE, … ）に並べたものと定義する。種々のシャンペンのスキーマとその実体は図2.15の様に示される。

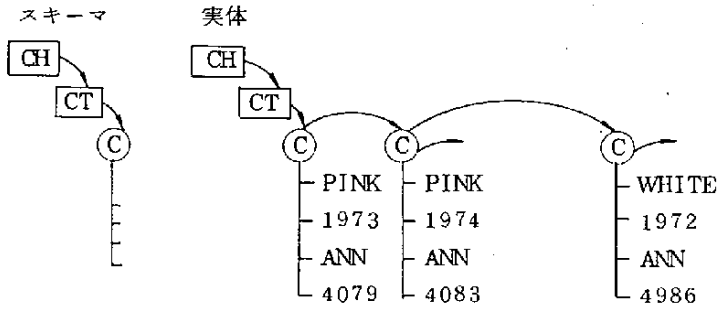


図2.15　種々のシャンペンのスキーマと実体

2) Aggregate関係

年代別のシャンペンとワインというaggregate関係YRを定義する。Aggregate関係 YRはWNとCH collection関係上に定義し，同じ年代によって集合をとり，年代の昇順に並べたものと定義する。またaggregate YRは年代の昇順に並んだ単一集合としてDBによって集められるものとする。

年代別のシャンペンとワインのスキーマと実体は図2.16の様に示される。

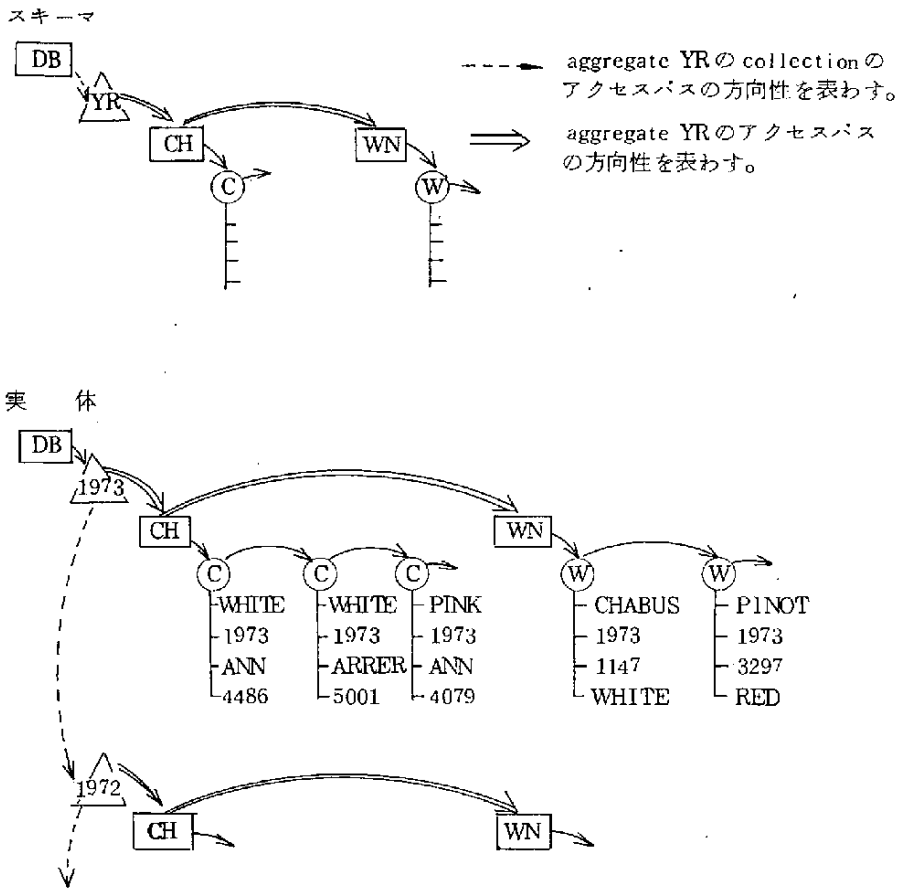


図 2.16　年代別のシャンペンとワインのスキーマと実体

2.4.3 論理アクセス構造（LAS）

論理アクセス構造の構成要素は名前，タイプ，許容値クラスなどのような属性を記述することによって定義される。仕様は，どの論理項目を集めて論理グループがつくられるのか，どの論理グループが関係しているのか，またその実態がどのように集まってデータベースを構成しているのかを示す。

論理アクセス記述を定義するのに使用する構成要素は次の５つである。

① 属　　性

② 論理項目

③ 論理グループ

④ 論理 collection

⑤ 論理 aggregate

以下にこの詳細について述べる。

1) 属　性

属性一値の集合が定義され，属性 section に名前が与えられる。この名前は属性 section でそれと関連する属性一値の集合を表わしているだけである。スキーマ上の各論理項目，グループ，collection と aggreate 関係はそれと関連するそれの先に定義された属性からなる。

〔例〕 属性名　　属性の値

PARTY-NAMES　　（／WHIG，FEDERALIST，POPULIST，
REMOCRAT，REPUBLICAN，NULL ／）

2) 論理項目

論理項目はLASの不可分の単位である。すべての他のconstructは究極的には項目上に定義される。各項目と関連するものは validation 属性であり，各属性の値集合を項目定義で与えなければならない。

項目スキーマの主属性は，legal value のそれと関連する集合である。項目実体は legal value の集合から引き出される値の１つである。

〔例〕 項目名　　validation 属性

ITEM PARTY　　VALIDATION IS PARTY-NAMES

3) 論理グループ

１つ以上の論理項目の関連で表わされる。

〔例〕

```
LOGICAL ITEM DESCRIPTION
    ITEM NAME VALIDATION IS NAMES
    ITEM DATE-OF-BIRTH VALIDATION IS DATES
    ITEM PARTY VALIDATION IS PARTY-NAMES
    ITEM RELIGION VALIDATION IS RELIGION
LOGICAL GROUP DESCRIPTION
LOGICAL GROUP IS PRESIDENT          ← 論理グループ名
    DEFINED OVER NAME DATE-OF-BIRTH PARTY RELIGION.
                  ↑ 構成する項目名
```

論理グループは構成項目に属性をもつかもしれない。Key属性はその１つである。項目や項目の連鎖は，Keyとして宣言される時，論理グループの全実体で項目の値がユニークでなければならないと制限される。

論理グループもまた関連する validation 規則をもつかもしれない。例えば，Parsonグループでは，もし性別が＂男性＂ならば，pregnancy（妊娠）の数は０でなければならない。

4) 論理 collection

論理 collection は，次の４つで定義される。

a) 論理 collection の名前

b) collecte される論理項目，グループ，aggregate や collection 関係の名前

c) 選択規則

d) 順序規則

〔例〕 赤ワイン collection の定義

```
COLLECTION RED-WINES                    〔 collection の名前 〕
    OVER WINES                          〔 collect される論理グループ名 〕
    SUBSET BY WINES. TYPE = "RED"       〔 選択規則 〕
    ORDERE BY YEARS ASCENDING           〔 順序規則 〕
```

もし選択規則として ENTIRE が与えられたら，Construct の全実体は論理 collection の単一の実体のメンバーとなる。例えば，タイプや年代に関係なく量の降順にすべてのワインをアクセスするパスは次の通りである。

```
COLLECTION ALL-WINES
    OVER WINES
    SUBSET IS ENTIRE
    ORDER BY QUANTITY DECREASING
```

選択規則は，construct の実体を collection の実体の可変数に分割することにも使用される。

これは，選択に対して UNIQUE VALUE を記述することで実行される。UNIQUE VALUE で分割されるとき，分割する construct のユニークな値と同じ数だけ論理 collection の実体が存在する。

例えば，与えられた年代に対して，タイプに関係なく WINES の全実体を通るアクセスパスは次のとおりである。

```
COLLECTION YEARS-WINES
    OVER WINES
    SUBSET IS UNIQUE VALUE OF YEAR
    ORDER IS QUANTITY DESCENDING
```

論理 collection も関連する validation 規則をもつかもしれない。これらの規則は一般に，ある論理 collection の実体内の構成の全実体の強制的な参加としてか，論理 collection 実体の construct の最小と最大値の限界として扱わなければならない。

5) 論理 aggregate

論理 aggregate は次の３つで定義される。

a) aggregate の名前

b) aggregate が定義される construct の名前

c) 照合規則

例えば，WINE PRODUCT データベースで次のような論理グループが存在すると仮定する。

GROUP GROWER

OVER NAME BIRTHPLACE FATHERS-NAME

項目PRODUCE-NAMEを含んだグループWINESを仮定する。論理collection として次のものを定義する。

COLLECTION WINE-BY-GROWER OVER WINES

SUBSET IS UNIQUE VALUE OF PRODUCER-NAME

ORDER IS IMMATERIAL

GROWERからWINES-GROWNのcollectionまでのアクセスパスは次のように定義される。

AGGREGATE WINE･GROWN

ORDER GROWER , WINE-BY-GROWER

MATCH GROWER. NAME = WINES.PRODUCER-NAME.

2.4.4 物理アクセス構造（PAS）

物理アクセス構造は，格納媒体の構造仕様を決めるものである。PASは，格納ロケーションとアクセスパスで定義される。

① アクセスパス

アクセスパスとは，データの格納の基本単位である格納セルや格納セルの集合を相互に結合する単方向性オペレータ，Stringで表わされる。

② 格納ロケーション

格納ロケーションとは，格納属性や格納属性の集合を相互に結合する単方向性オペレータ，stringで表わされる。

③ アクセスパスと格納ロケーションの関係

アクセスパスと格納ロケーションは格納属性や格納セルの物理collection又はaggregateによって表わされる。

物理アクセス構造は次の5つで定義される。

a) 格納属性

b) 格納セル

c) 物理グループ

d) 物理collection

e) 物理aggregate

1) 格納属性

格納属性は，PASの他の要素の特徴を記述する。代表的な格納属性は，格納媒体属性とアドレス属性である。

2) 格納セル

格納セルはデータ格納の基本単位である。格納セルは自分自身の識別を持っている。

また物理的媒体に依存する。意味のある識別可能な最小単位は1文字もしくはバイトである。媒体上では，これが格納セルとして記述される。

3) 物理グループ

物理グループは1つ以上の結合された格納セルから成る。装置（I/O, ファイル）と主記憶間での転送単位である。このような単位は，通常物理レコードもしくはブロックに対応する。

4) 物理 collection

物理 collection は，与えられた物理グループ，物理 aggregate，物理 collection の1つ以上の実体の結合である。物理 collection を形成するすべての実体は同種でなければならない。これはファイルに対応する。

物理 collection は次の3つによって定義される。

a) colection の名前

b) 選択規則

c) 順序規則

5) 物理 aggregate

物理 aggregate は，2つ以上の物理グループ，物理 collection，物理 aggregate の結合である。結合は，異種のものに対しても可能である。結合の条件として照合条件が定義される。

物理 aggregate は次の2つで定義される。

a) aggregate が定義される要素の名前

b) aggregate される実体を決める規則となる照合条件

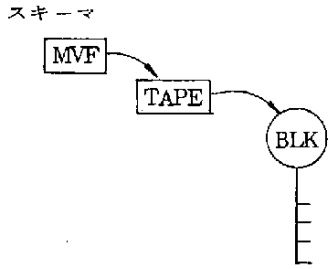
〔例1〕 マルチボリュームテープ媒体に対する物理アクセス構造

装置の主な格納属性はそのアドレスである。マルチボリュームテープ媒体では，このアドレスはVSN（ボリューム順番号）とBSN（ブロック番号）から成る。

格納セルとしては，チャンネル，フレーム，ワード etc が考えられる。

物理グループとしては，ブロックが考えられる。

格納属性	TAPE-ADDR (VSN, BSN)
格納セル	WORD (36 ビット)
物理グループ	BLK (n WORDS)
物理 collection	TAPE グループ BLK 上に定義され，VSN のユニークな値で部分集合をとり，BSN の昇順にならべる。
物理 aggregate	MVF collection TAPE 上に定義され，VSN のユニークな値で部分集合をとり，VSN の昇順にならべる。



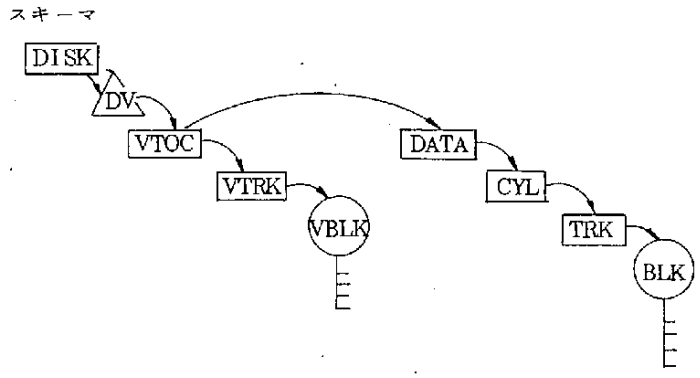
〔例２〕 ディスク媒体の物理アクセス構造

格納属性	DATA-ADDR(CYL-NO, TRK-NO, BLK-NO)
格納セル	WORD （ 36 ビット ）
物理グループ	BLK （ n WORDS ）
物理 collection	TRK グループ BLK 上に定義され，（ CYL-NO, TRK-NO ）のユニークな値で部分集合をとり，TRK-NOの昇順にならべる。
物理 collection	CYL collection TRK上に定義され，（ CYL-NO）のユニークな値で部分集合をとり，TRK-NOの昇順にならべる。
物理collection	DATA collection CYL上に定義され，singular な部分集合をとり，CYL－NO の昇順にならべる。

VTOC の記述は，グループ VBLK の追加および BLK と TRK の定義と同じようなcollection VTRK が要求される。

collection VTOC は collection VTRK 上に定義される。 aggregate DV は次のように VTOC と DATA collection 上に定義される。

物理 aggregate	DV collection DATA, VTOC 上に定義され，singular に併合させる（ i.e. collection DATA と VTOC のただ１つの実体が存在する）



2.4.5 LAS と PAS の mapping

論理アクセス構造と物理アクセス構造には各々のEncoding がある。論理アクセス encoding (LASE)は，データ構成がどのように実現されているか，アクセスパスがどのように実現 (contiguity, pointers/etc) されているかを定義している。

物理アクセス encoding (PASE)は，アドレスがどのように実現されているか，アクセスパスがどのように実現 (contiguity, pointers, hasing etc) されているかを定義している。

論理データ構造を表わすLAS/LASEと物理データ構造を表わすPAS/PASEの関係は Read／Write要素によって関係づけられる。Read/Write は手続き的な仕様によって定義される。Read process は物理データ構造から論理データ構造へマッピングを行なう。Write process は論理データ構造から物理データ構造へのマッピングを行なう。

Read process では，PASのあるアクセスパスが traverse されると，それに対応する LAS のアクセスパスが選択される。

逆にWrite process では，LASのあるアクセスパスが traverse されると，Write 仕様の論理に合ったデータ項目の値をもとにPASのアクセスパスが traverse されデータが格納される。

Read やWrite 仕様は 4 つの部分から成っている。

(1) LAS に対応する 1 つの実際の traversal 仕様

(2) PAS に対応する 1 つの実際の traversal 仕様

(3) 名前によってPAS やLAS travasals 各々を呼ぶための traversal 管理

(4) バッファのために中間ワークスペースとして使用する " local storage " の要求

Write process の仕様の例として図 2.17 の様な論理データ構造と図 2.18 の様な物理データ構造を考えてみる。

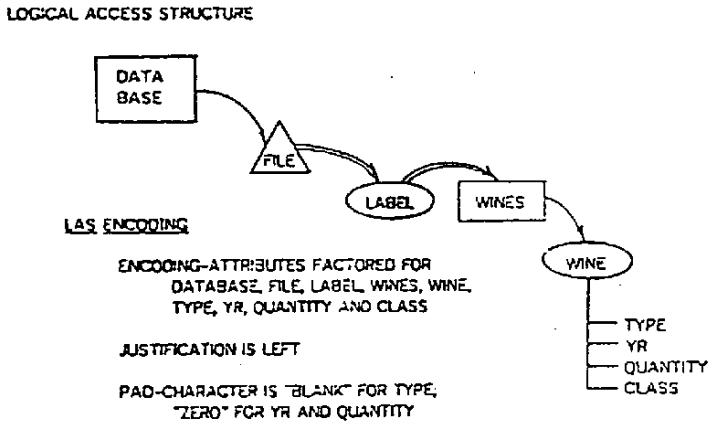


図 2.17　論理アクセス構造

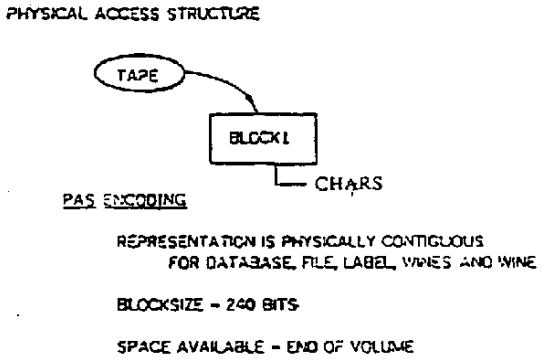


図 2.18　物理アクセス構造

Write仕様の例

```
WRITING-SPECIFICATION                          ←バッファとして使用する"Local
                                                 storage"の要求
STORAGE MANAGEMENT
  BUFFER OUTBLOCK SIZE 240 BITS

LAS TRAVERSAL INITIAL-SELECT                   ←LASに対応するtraveral仕様
  SELECT DATA-BASE

LAS TRAVERSAL FILE-SELECT                      ←FILEアクセスが出来たら，LASの
  SELECT FILE WITHIN DATA-BASE                   LABEL-SELCTを呼び出す
  ON SUCCESS INVOKE LAS LABEL-SELECT

LAS TRAVERSAL LABEL-SELECT                     ←OUTBLOCKにLABELのEncodingを
  SELECT LABEL WITHIN FILE                       使用してLABELをEncodeする。
  ENCODE LABEL USING LABEL-ENCODING INTO OUT-    終ったら，PASのPUT-IN-BLOCK,
    BLOCK                                        LASのWINE-SELECTを呼び出す
  ON COMPLETION INVOKE PAS PUT-IN-BLOCK, LAS
    WINE-SELECT

LAS TRAVERSAL WINE-SELECT                      ←FILE内のWINESを管理したら，
  SELECT WINES WITHIN FILE                       NEXT-WINEを呼び出す
  ON SUCCESS INVOKE LAS NEXT-WINE
```

```
LAS TRAVERSAL NEXT-WINE
  SELECT NEXT WINE WITHIN WINES
  ON END RETURN
  ENCODE WINE USING WINE-ENCODING INTO OUT-
    BLOCK
  ON COMPLETION INVOKE PAS PUT-IN-BLOCK, LAS
    NEXT-WINE
```

←WINES内のNEXT WINEを選択し，ENDならもどる。
ENDでない場合，OUTBLOCKにEncodeを使用してWINEをEncodeする。
終ったら，PASのPUT-IN-BLOCK，LASのNEXT-WINEと呼び出す。

```
PAS TRAVERSAL INITIAL-BLOCK
  SELECT TAPE
  SELECT NEXT BLOCK WITHIN TAPE
```

←PASのtraversal仕様
TAPEを選択
TAPE内のNEXT BLOCKを選択

```
PAS TRAVERSAL PUT-IN-BLOCK
  SELECT CURRENT BLOCK WITHIN TAPE
  BLOCK BLOCK1 FROM OUTBLOCK AT POSITION
    NEXT-AVAILABLE-SPACE
```

←TAPE内の現在ブロックを選択
OUTBLOCKからBLOCK1をNEXT-AVAILABLE-SPACEの位置にブロックする。

```
TRAVERSAL MANAGEMENT
  INITIALIZATION SECTION
    INVOKE LAS INITIAL-SELECT
    INVOKE PAS INITIAL-BLOCK
  TRAVERSAL SECTION
    DO UNTIL END OF PATH WINES
      INVOKE LAS FILE-SELECT
    END.
```

traversal管理
LASとPASの初期アクセスパスの選択
アクセスパスWINESが終るまでLASのFILE-SELECTを呼び続ける。

2.5 まとめ

この章では，データベースシステムのアーキテクチュアをANSI／X3／SPARCの3層スキーマ構造でとらえ，現在の代表的データモデルのリレーショナルモデルとCODASYL DBTGモデルについて論じた。リレーショナルモデルでは，意味構造を表現する従属性，更新異常をなくすための正規化，非手続き型である言語と外部スキーマとしてのViewについて触れた。

CODASYL DBTGモデルでは，モデルのフォーマルな定義を試み，内部スキーマレベルとしてのアクセスパス，手続き型言語と外部スキーマであるサブスキーマについて述べた。これによって，リレーショナルスキーマとDBTGスキーマとの変換（同種化）が可能となる。

また内部スキーマモデルとして，Stringモデルによる論理アクセス構造と物理アクセス構造の記述によるデータ構造の一般化モデルの試みをみてきた。

上記によって，次の様な事が言える。

リレーショナルモデルにおいて，意味構造を表わす関数従属性（2.2.4で述べられた従属性以外にも色々な従属性が導入されている）だけではうまく表現されていない。リレーショナルモデルに

おいて意味構造を表現する試みとして，付記Ｉの様なオブジェクトモデルなどの試みがある。

またリレーショナルモデルにおいて，ユーザとのインタフェースを考える上でViewの概念は非常に重要である。７章で述べるように検索の問題は問合せ変形（query modification）手法において実行することが可能である。しかし更新の問題に関しては諸々問題が残されたままである。これに対して，オブジェクトモデルとこれに基づいたデータ抽象化技法（付記Ｉ）の適用を考える必要があろう。

DBTGモデルのフォーマルな定式化により，リレーショナルモデルからDBTGモデルへモデル変換などが実行できるようになった。

内部スキーマモデルでの論理アクセス構造と物理アクセス構造モデル化は，アクセスパスの最適化とデータ独立性を達成するためには研究がもっと進められなければならない。

3. オフィス情報処理におけるデータ・ベースシステムモデル ―フォームフローモデル〔TANAY 81〕

3.1 はじめに

近年，生産業務と比較した時のオフィス業務の生産性改善の遅れが問題となっている。オフィス業務の生産性向上と高度化とを計算機と通信システムを用いた情報システムによって行なう試みはオフィス情報システム（以降OISと記す）と呼ばれている〔COOKC80，BAUMS80，ELLIC80，LADDI80〕。〔COOKC80〕は，情報制御ネット（ICN）モデル〔ELLIC80〕を用いて，各オフィス業務間の仕事の流れ（制御フロー）と，各仕事の必要とする情報の格納媒体の関連（情報フロー）とを表わし，オフィス業務の解析と合理化（streamlining）とを試みている。〔LADDI80〕は，オフィスにおけるフォーム（オフィスでの情報単位e.g.書類）の流れに着目して，その流れをメッセージフロー問題として解析し、オフィス生産性を定量化しようとしている。〔BAUMS80〕は，自動化された仕事の流れと，管理者による制御との関係をペトリネットによって表わそうとしている。

上述した現在までのOISへの試みにおける問題点は，情報システムの核となるデータベースシステム（以降DBSと記す）とオフィス情報処理（業務）との関連が明らかでないことである。本論文では，この問題の解決を目指している。OISにおけるDBSの第1の特徴は，従来の統合DBSと比較して，データベース内の全てのデータが，オフィス全体で共有される必要がない，即ち全データを1つに統合する必要がない点である。各オフィス業務内で用いられるデータは，主にこの業務内でローカルに利用される。このデータ共有の単位を部局と呼ぶ。部局とは，例えば部，課，プロジェクト，グループ等である。データは部局内では，一致性を保つために統合管理される。しかし，概して部局間では，データの一致性に対する制約はゆるい。例えば，ある部から他の部に書類が届けられた時，送られた部でのこの書類への更新は前の部に知らされる必要はない場合である。この様に，OISでは部局ごとに必要とするデータを統合し，部局間で共有するデータに対して比較的ゆるい統合をオフィス全体で行なえばよいことになる。又，部局は互いにオーバラップすることも階層関係を持つことも許される。以上のことは〔HAMM80〕におけるfederated DBS概念と類似している。

オフィスにおいて処理される情報単位をフォーム（form）と呼ぶ。フォームは，オフィス従業員から視ると，書式づけられ印刷された書類であると共に，他従業員又は部局に対する情報転送単位である。フォームとは，情報の表現であるとともに，情報伝達単位である。

OISにおけるDBSを考える上で他の重要な点は，データの共有関係である。ある部局から他の部局にフォームが転送された後，このフォームに対する一方の処理が他方に影響を与えるかどうかの問題である。フォームのコピーを取り，これを更新する場合には，フォームの共有は生じない。一方，フォームを共有している場合には，ある部局による更新は，他の部局によっても視られることになる。この共有と非共有の関係を明らかにすることは，オフィスでのDBSを考えるための基本となる。

以上より，OISにおけるDBSを考える上で，次の3点が重要な概念となる。

1) データベースのオフィス全体での統合から，必要な部局ごとでの統合化（統合DBSからfederated DBSへ）。

2) オフィス情報処理の単位としてのフォームについて，その情報表現と情報の転送（フロー）とを考えねばならない。

3) フォームには，部局間で共有されるもの（共有フォーム）と共有されないもの（非共有フォーム）とがある。

上記3点に基づいて，本論文ではOISにおけるDBSとしてのフォームフローモデルについて論じる。3.2ではこのモデルの思想と概要を述べる。3.3ではこのモデルの数学的基礎づけを行なう。3.4では，フォームフローモデルの例を示す。

3. 2　フォームフローモデルの思想と概要

オフィス情報システム（OIS）における情報処理単位はフォームである。OISのDBSの位置づけを明らかにするためには，フォームとそのフローと，DBSとの関連づけが必要になる。

OISでのDBSの第1の特徴は，データが各部局内で高い局所性をもって用いられることである。部局内で必要なデータの一致性は，部局内において保たれる必要があるが，部局間で共有されるデータの一致性保持要求は比較的ゆるいものである。例えば，他の部局に廻った書類の内容変更を瞬時に知る必要はない時が多いし，知る必要性さえない時もある。これ等のことは，DBSの立場から見ると，データベースから導出されたスナップショットが新たなデータベースとして共有されていく場合である。即ち，データは各部局内で一致性を保つために統合される必要があるが，部局間で共有されるデータに対してはそれ程強い統合を求められない。オフィス全体のDBSは，従来の中央集権的な統合DBSではなく，分権的なfederated DBS〔HAMM80〕が必要となる。

オフィスで処理される情報単位はフォームである。フォームという言葉には，次の2つの意味がある。

i) 出力フォーム（output form）

ii) フォームリレーション（form relation）

前者は，日常我々が用いる書式づけられた書類に対応している。このフォームは，書類としての情報表現であると共に，他との情報転送単位でもある。

後者は，フォームの持つ情報のシステム表現としてのリレーションである。システム内でフォーム処理と管理とは，このフォームリレーションに対してなされる。一般に1つのフォームリレーションは，複数の出力フォームを持ち得る。フォームリレーションと出力フォームとのマッピングでは，特にrepeating groupの処理が問題になる。マッピングとしては，以下の点が問題になる。

1) フォームリレーションから出力フォームの定義

2) 出力フォームを通してのフォームリレーションのアクセス

3) 出力フォームと出力紙の制御

4) フォームアクセスの制御(authorization と authentication)

フォームは，オフィス情報処理の単位であるとともに，部局間又は部局内の情報転送単位である。例えば，ある部員から部長への書類の提出がこれに当たる。このフォーム転送をフォームフローと呼ぶ。フォームフローの単位はフォームリレーションである。フォームフローは，目的部局での新たな情報の生成をもたらす。

フォームとしては，次の2種を明確に分ける必要がある。

i) 共有フォーム

ii) 非共有フォーム

共有フォームとは，ある書式を通して複数部局又は従業員が同一の情報を視れるものである。例えば，会議の出欠案内の様に他人の出欠情況を視れるものは共有フォームである。共有フォームを表わすフォームリレーションは，リレーションのビューに対応している。即ち，ある従業員によるフォームの更新は，同じフォームを保持している他の従業員にも伝搬することになる。非共有フォームとは，これを保持している従業員によって，ローカルに更新出来るものである。これを表わすフォームリレーションは，リレーションから導出されたスナップショットに対応している。このスナップショットを再び共有することも可能である。

3. 3 フォームフローモデル

前節に述べた思想に基づき，OISのためのデータモデルを提案する。本論文では，フォームの情報単位としての性質を論じ，出力形態としてのフォームに関しては別の機会に論じる。したがってフォームはリレーションとしてモデル化し，これをフォームリレーションと呼ぶ。

以下に述べるフォームフローモデルは，OISにおける次の3つの概念を取り込んでいる。

i) federated DBS

ii) フォームリレーションの共有と非共有

iii) フォームリレーションの流れ

ここで言うフォームの流れとは，次の2種類のことを意味する。

i) 他のフォームリレーションから，新たなリレーションをビューとして定義(共有フォームリレーションの定義)

ii) 他のフォームリレーションから，新たなリレーションをsnapshot として導出(非共有フォームリレーションの導出)

i)では，フォームの転送後，このフォームを作成するのに用いたもとのフォームリレーションを変更すると，転送先でのフォームの内容も変更され，逆に転送先でこのフォームを変更すると，転送先のフォームリレーションも変更される。これに対して ii)では，転送先のフォームリレーションと，転送されるフォームとはまったく独立である。 i)で転送されるフォームを仮想フォームと呼び，ii)のそれを実フォームと呼ぶ。仮想フォームリレーションに対する更新問題は，オブジェクトモデルに基づいたデータ抽象化技法〔 TANAY80 〕によって解決することができる。

この2種のフォームリレーションは，任意のフォームリレーション上に階層的に定義され得る。

フォームフローモデルは，フォームリレーションを表わすノードとフォームリレーションの定義と導出の関係を表わすアークとからなる高階グラフによって表わされる。ノードを情報単位と呼び，アークをフォームフローと呼ぶ。この高階有向グラフをフォームスキーマ図と呼ぶ。

我々は，オフィスにおける事務処理を，

1. 処理や管理の単位となる情報単位の集合　I
2. 情報の流れの集合　F
3. 部局の集合　D
4. 情報処理を行う行動の集合　A
5. 情報処理行動を情報の流れに対応づける写像　α
6. 各部局で見ることのできる情報と，許される情報処理を規定する写像　β

の6つの基本概念から構成されているものとしてとらえ，6項組（I，F，D，A，α，β）でこれをモデル化する。

Iは情報単位と呼ばれる要素の有限集合である。Iの各要素nには，その表現ℓ(n)を考えることができる。ℓとしてはIを関係の集合Rの中へ写す写像

$$\ell : I \rightarrow R$$

を考える。ℓ(n)はnに対するフォームリレーションである。

Iは互いに排反なIe，Ia，Ivに分割される。

$$I = Ie \oplus Ia \oplus Iv$$

Ieはオフィス外の情報単位の集合を表わす。外部情報単位は内部の情報単位を導出したり，逆に内部の情報単位より，外部情報単位が出力として導出される。このようにIeは，オフィスの入出力情報を表わす。スキーマ図では，外部情報単位を三角形で囲んで表現する。

Iaは実情報単位と呼ばれる要素の集合である。実情報単位nに対するℓ(n)は実フォームリレーションであり，実データとして，DBMSによって物理的に1つの関係として蓄積されている。実情報単位は正方形で囲んで表現する。

これに対するIvは仮想情報単位と呼ばれる要素の集合である。仮想情報単位nに対するフォームリレーションℓ(n)は，システム中に物理的に1つの関係として蓄積されているのではなく，他の情報単位n_1，n_2，……，n_rに対応する関係$\ell(n_1)$，$\ell(n_2)$，……，$\ell(n_r)$より定義されるビューの1つである。スキーマ図において，仮想情報単位は菱形で囲んで表現する。

Fはフローの集合と呼ばれ，（I，F）はIをノードの集合，Fを

$$F \subset 2^I \times I$$

なる高階の有向辺の集合とする高階有向グラフと定義する。

Fは情報のフローの集合F_Iとフレーム（書類の書式の枠組）のフローの集合F_Fとに分割される。

$$F = F_I \oplus F_F$$

情報フローは，実情報単位をいくつかのソースの情報単位から導出するものである。生成された実フォームリレーションとソースのフォームリレーションは互いに独立で，情報の共有一括管理は行なわれない。情報フローは，書類の物理的な転送，コピー，ディスプレイへの情報の表示，電子郵便等に対応している。上述のことから，F_I を，

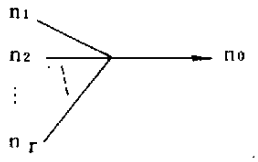
$$F_I \subset 2^I \times (I_a \cup I_e)$$

と定義する。F_I に対して，

$$(\{n_1, n_2, \cdots, n_r\}, n_0) \in F_I$$

は，ソースフォームリレーション $\ell(n_1)$，$\ell(n_2)$，…，$\ell(n_r)$ より関係演算を用いて得られるある関係が，実情報単位 n_0 に送られ，$\ell(n_0)$ の更新が送られることを意味する。

情報フローは実線の矢（→）によって表わし，高階の有向辺（$\{n_1, n_2, \cdots, n_r\}$，$n_0$）は，



と表わす。

フレームフローは，仮想情報単位を，いくつかのソースの情報単位に対するフォームリレーション上に定義されるビューリレーションとして定義する。定義された仮想フォームリレーションと，ソースのフォームリレーションとの間には情報の共有がなされ，一方における更新は，他方に影響を及ぼす。F_F は，

$$F_F \subset 2^{I-Ie} \times I_v$$

と定義され，

$$(\{n_0, n_1, \cdots, n_r\}, n_0) \in F_I$$

は，$n_0 \in I_v$ に対する $\ell(n_0)$ が，$\ell(n_1)$，…，$\ell(n_k)$ のビューとして定義されることを意味する。フレームフローは，スキーマ図では，破線の矢印（……›）で表わされる。

Dは，部局の集合である。どの部局で，どの情報単位が見え，どの情報処理行動が実行を許されるかは，後述の β が規定する。

Aは情報処理行動の集合である。

α は，各情報処理行動 $a \in A$ に，情報フローの有限系列 f_1，f_2，…，f_n を対応させる。系列中の情報フロー

$$f_i = (\{n^i_1, n^i_2, \cdots, n^i_{j_i}\}, n^i_0)$$

に対し，a は f_i を $f_i + 1$ 個の関係変数を含む関係演算式 e に写す。

$$\alpha : A \rightarrow F_I^*$$

F_I^*：F_I の要素の有限系列の集合

$$\forall a \in A$$

$$a : F_I \rightarrow E$$

E：有限個の関係変数を含む関係演算式の集合

$a \in A$に対して,

$$\alpha(a) = (f_1, f_2, \cdots, f_n)$$

とすると，この情報処理行動aは,

$$\ell(n_0^i) \leftarrow a(f_i)(\ell(n_1^i), \ell(n_2^i), \cdots, \ell(n_{j_i}^i))$$

なる更新を$i = 1, 2\cdots$, nについてのiの順に実行するという意味をもつ。

F_Fに対しては,

$$\delta : F_F \rightarrow E$$

が定義されているものとする。$f = (\{n_1, n_2, \cdots, n_r\}, n_0) \in F_F$に対して，$\delta(f)$はr個の関係変数を含む関係演算式で，ビュー$\ell(n_0)$は

$$\ell(n_0) = \delta(f)(\ell(n_1), \cdots, \ell(n_r))$$

と定義される。

βは権限を表わす写像で,

$$\beta : D \rightarrow 2^{I-Ie} \times 2^A$$

と定義される。$d \in D$, $Id \subset I-Ie$, $Ad \subset A$とし，$\beta(d) = (Id, Ad)$とすると，Idは部局dで見ることのできる情報単位の集合を表わし，Adは部局dで許される情報処理行動の集合を表わす。

実情報単位には，これを蓄えているファイルの種類を指定するファイル属性が付随する。ファイル属性には,

(1) データベース関係

(2) ワークファイル

(3) プリントファイル

(4) ディスプレイファイル

(5) 電子郵便

等があり，(3), (4), (5)の属性を持つファイルへの更新は，出力動作を意味するものとする。(3), (4), (5)の属性ファイルには，出力書式属性が与えられ，これにより，フォームの出力を行なう。実情報単位で出力ファイル(3), (4), (5)の属性をもつものは，フォームフロースキーマ図において，2重の正方形で囲んで表現する。

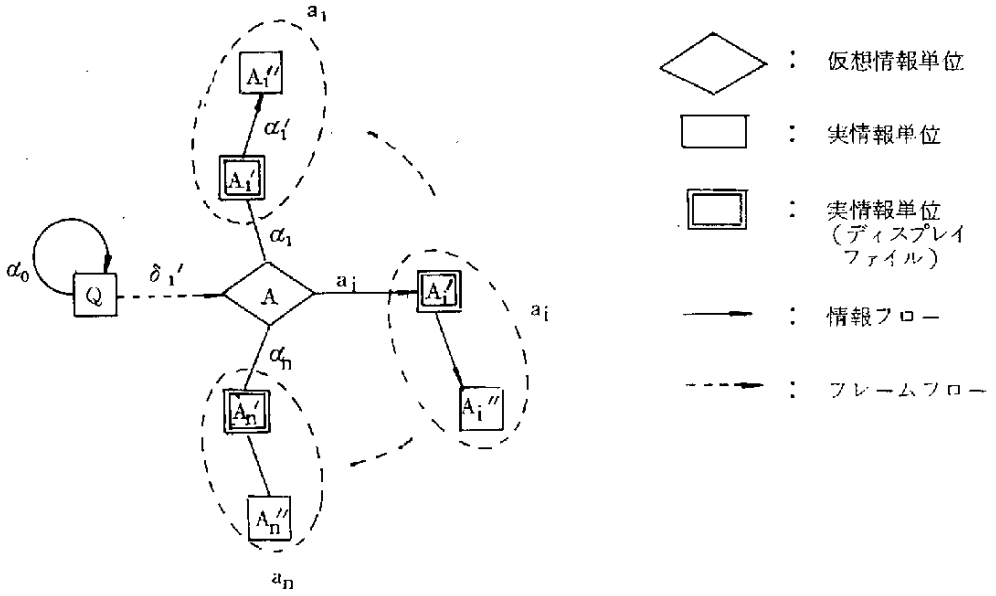


図 3.1　例1のフォームフロースキーマ図(a)

3. 4　フォームフローモデルの例

本章では，フォームフローモデルの例について述べる。

例1　会議開催案内とその出欠の回覧

部内会議案内が部の総務(a)から全部員（ a_1, …, a_2 ）に出され，各部員はその出欠を記入して総務に返す例を考えてみる。又，各部員は会議開催情報（ e.g.場所，日時，目的）のコピーを取るものとする。これを図 3.1 に図示する。 a は会議情報を実情報単位Qに生成し，これから会議案内フォーム（ A ）をつくり全員に送る。各部員（ a_i ）は，Aを通して会議案内内容を知るとともに，その時点での出欠情況を視れる。各員は，Aを通して自分の出欠をQに書き込む。この書き込みはビューAを通しての更新となる。各員は自分のディスプレイファイル(A_i')にAを通して現在のQのスナップショットを導出し，この中から必要な部分を自分のローカルファイル（ A_i'' ）に出力する。

実情報単位　Q：会議データベース

A_i'：a_i のディスプレイファイル　　A_i''：a_i のパーソナルファイル

仮想情報単位　A：会議開催案内のフレーム

情報フロー　α_0：a_1 による会議開催の起案

α_i：a_i による自分のディスプレイ（ A_i' ）への会議情報の出力

α_i'：a_i が必要とする会議情報をパーソナルファイル（ A_i' ）に格納

フレームフロー　δ_1：会議開催案内のフレームをつくる

図 3.1 (b)　　会議開催案内

例2　書類の作成

次の例は書類の作成である（図3.2）。ある部員（a_1）によって起案され，これが課長（a_2），部長（a_3）によって修正認可される。各時点で上司に対して書類を提出するときは，そのコピーを各自で保管する。部長は，各課から出されてきた書類（F_1', F_2''）からそのサマリ書類をつくり全員に見せる。各員は，上司による修正を各時点で知ることができる。

a_1から提出された書類F_1'に対するa_2の更新（修正）は，仮想フォームリレーションF_1'を通して，a_1も視ることもできる。a_2からa_3に提出された書類F_1と，a_3がまとめたFに対する更新は，実フォームリレーションF_1の更新及び，a_1とa_3とはこの更新を視ることができる。

実情報単位　F1：書類リレーション

A_1', A_2', B_1', B_2'', C_1''　：ディスプレイファイル

A_1',　A_2'',　F'　　：パーソナルファイル

仮想情報単位　F_1'：a_1の起案した書類のフレーム

F_1''：　a_2からa_3に提出した書類のフレーム

F_2''：　a_4からa_3　〃

F：a_3がF_1', F_2''からまとめた書類のフレーム

情報フロー　α_0：a_1による書類の起案

α_1：F_1'のディスプレイ（a_1へ表示）　α_2：F_1'のディスプレイ（a_2へ表示）

α_1'：F_1'のコピー　　α_2'：F'のコピー

α_3：F_1''のディスプレイ（a_3に表示）　α_5：Fのコピー（a_1のパーソナルファイル）

α_3'：F_1''のコピー　　α_4''：F_1''のディスプレイ（a_3に表示）

フレームフロー　δ_1：書類の起案フレームの定義　　δ：F_1''とF_2''とからFを定義

δ_1'：F_1''をF'から定義

図3.2(a)　書類作製（例2）

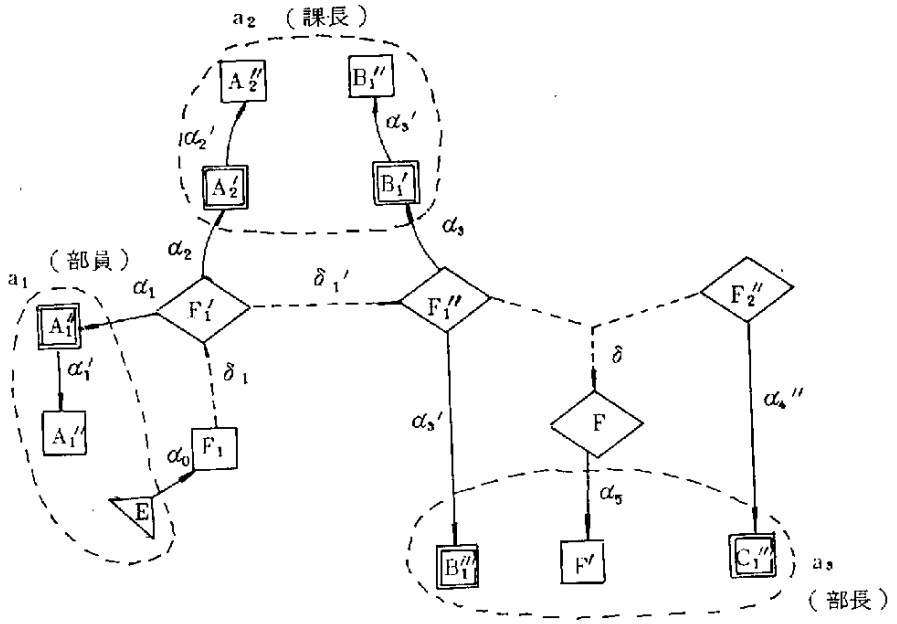


図3.2　例2のフォームスキーマ図式(b)

3. 5　まとめと今後の課題

本論文では，オフィス情報システム（OIS），CAE／CADを中心とした今後の計算機アプリケーションにおけるデータベースシステム利用のモデルとして，フォームフローモデルの提案を行なった。このモデルは，federated DBS，フォームリレーションの共有，フォームリレーションの流れの３つを基本概念としている。これによって，OIS等のDBSとの関係が不明確であったアプリケーションにおけるDBSの位置づけを明らかにできたと考えている。更にOISの設計用ツールとしても利用できる。

今後の課題としては以下の点がある。

1) OISにおけるコンカレントオペレーションの記述。全てのオペレーションを許すのではなく，各部局に対して許されるオペレーション集合の定義が必要になる。

2) SBA〔ZLOOM80〕におけるトリガ機能のモデル化。

3) 仮想フォームリレーションに対する更新に対して，オブジェクトモデルとデータ抽象化技法〔TANAY80〕の適用。

4) 分散型データベースシステムとの関連の明確化。

5) CADへの適用

4. 分散型データベースシステム－JDDBS－II
—スキーマ層モデルと通信処理モデル〔TAKIM81a〕

将来のオフィス情報システム（OIS）を始めとする計算機利用は，各サイトに分散した特性の異なったデータベース資源を通信ネットワークによって統合利用することが中心となってくると思われる。これを支える基本技術は，分散型データベースシステム（DDBS）と呼ばれるシステムである。DDBSは，データベースとこれに対する管理処理機能を有したデータベースシステム（DBS）と，これらを相互に通信させる通信ネットワークとから成るとともに，次の条件を満足するシステムである。

1) 各DBSは，互いに意味的関連性を持っているデータベースを有している（意味的関連性）。

2) 各DBSは，通信ネットワーク上に分散している（物理的分散性）。

3) DDBS上の全てのDBSは，論理的な1つのDBSに仮想化されている（論理的集中性）。

従がって，DDBSのユーザアプリケーションは，DDBSをあたかも1つの巨大DBSかのごとく，各DBSをどの様にアクセスし（異種性問題），どこに何があるか（分散問題）を意識することなく利用できるものである〔TAKIM78〕。DDBSを実現するためには，この異種性問題（heterogeneity problem）と分散問題（distribution problem）とを解決することが必要である。

本章では，当協会でモデル化を進めている分散型データベースシステムJDDBS-II（第II期Jipdec DDBS）の全体アーキテクチャについて述べる。JDDBS-IIの特徴は，これまでに研究開発してきたJDDBS－I〔TAKIM78，79，80a，b，JDDBS80〕と比較して，次の点が主要な特徴となっている。

1) ローカルネットワークの利用

2) 個人用小型データベースシステム

3) 更新機能

4.1では，DDBS研究動向を述べる。4.2ではJDDBS－IIの全体アーキテクチャを示し，4.3では次のスキーマ層モデルを，4.4では通信処理モデルを述べる。

4.1 DDBSの研究動向

分散型データベースシステム（DDBS）の本格的研究は，1976年から1977年にかけて欧米を中心に開始され，1979年から1980年にかけてそのインプリテーションを終了している。これらの例としては，SDD-1(CCA)〔ROTHJ77，80〕，LADDER(SRI)〔SAGAD78，MOORR77〕，POLYPHEME〔Univ. of Grenoble〕〔ADIBM78，80〕等をあげることができる〔表4.1〕。これらの特徴は同種DBS（i.e. リレーショナルDBS）を比較的低速（～50 kbps）な広域パケット交換ネットワークによって結合し，ユーザに対してデータの所在を意識することなくアクセスできる統一的なデータ記述のサポートを行なっている点である。更に，CCA（Computer Corporation of America）社によって開発されたSDD-1は，1979年後半にそのインプリメンテーションを終了するとともに，これの商用プロダクトとしての分散型モデル204（distributed model 204）シス

テムを1981年後半（第4／四半期）までに出荷しようとしている。上述してきたDDBS を第1期DDBSと呼ぶ。第1期DDBSを表4.1にまとめる。

1979年から1980年にかけて，上述したシステムの多くは，第2期DDBSの研究開発を開始してきている。第1期DDBSが，通信ネットワークを用いて，データベースリソースを統合利用出来ることの可能性，即ち一般的なDDBS方法論を明らかにするためのプロトタイプであったのに対して第2期DDBSは，具体的なアプリケーション（ i.e. オフィス情報システム）〔 LEBIJ 80 〕への適用を目指している。即ち， DDBSと分散処理（DPS）とが結合された分散型情報システム(distributed information system) (DISと記号) を目指していると言える。DDBSの利用形態としては，第1期DDBSが実際には検索のみがある程度実用となり得たが，OIS等の今後の アプリケーションは，より多くの更新が要求されてくると考えられる。更に多様な構成要素を組み込むために，異種DBSの統合が求められる。通信ネットワークとしては，第1期DDBSの主要なパフォーマンスボトルネックとなった広域ネットワークを替って，高速高信頼な1：N通信機能を備えたネットワーク (e.g. Ethernet , MITRENET , NBSNET)を各DBSと処理システムを結合するために用いようとしている。この背景には，光ファイバー等の通信ハードウェア技術の進歩がある。更に，ローカルネットワークの利用は， DDBSとデータベースマシン (DBMと記す) とを密接に関連するものとしてきている。ローカルネットワークについては，今後，プロトコルとホストインターフェースの標準化が重要な問題である。

上述してきた第2期DDBSの特徴を以下にまとめる。

1) 具体的アプリケーション　　オフィス情報システム (OIS)

　例　(SIRIUS)-DELTA (INRIA)〔LEBIJ80〕 KAYAK(INRIA)〔NAFFN80〕
　　DIALOG (Purdue Univ)〔WAH B80, YAOS80〕
　　(MICROBE-) SCOT (Univ. of Grenoble)

2) 異種DBSの統合　　CODASYL DBTG DBS

　例　SDD-1 (CCA)
　　LADDER (SRI)〔MOORR79〕

3) システムの効率化　　ローカルネットワークの利用

　例　SIRIUS(-DELTA)(INRIA)　　MICROBE(-SCOT)(Univ.of Grenoble)
　　DIALOG(Purdue Univ.)　　INGRES-MUFFIN(UC.B)〔STOMN79〕

4) 更新機能　　ビュー更新
　　同時実行制御

5) ユーザインタフェース　　自然語，3次元インタフェース，対話技法

表4.1　第1期DDBS

DDBS		SDD-1	LADDER	POLYPHEME
site(year)		CCA(U.S.A.)(1976-1979)	SRI(U.S.A.)(1976-1978)	Univ.of Grenoble(1976-1979)
design approach		top-down		bottom-up
application		NAVY	NAVY	
network		ARPANET	ARPANET	CYCLADES
user interface (EXS level)	model	relational model	attribute model	relational model
	language	Data language (procedural)	natural language	relational algebra
GCS level	model/ language	relational model/ QUEL(non-procedural)	relational model / relational calculus	MOGADOR(binary relational)/ relational algebra
	distributed query processing	yes	yes	yes
	concurrency control	pre-analysis/time stamp	no update	no update/no concurrency
	redundant copy	horizontal & vertical decomposition of relations	back up copy (copy of DB)	no redundancy
LCS level	model	≈relational	≈relational	relational ⟩ URANUS
	language	Data language	Data language	algebra ⟩ URANUS
	DBMS	Data computer (DEC)	Data computer (DEC)	SOCRATE

4. 2 JDDBS－Ⅱアーキテクチャ

本章では，当協会で開発を進めている分散型データベースシステムJDDBS-Ⅱの全体的なアーキテクチャについて論じる。

1977年春より1980年始めにかけて我々の開発したJDDBS-Ⅰ（第Ⅰ期JDDBS）は，パケット交換ネットワークJIPNET〔 YAMAK75 〕と，AIM（M-160），ADBS(ACOS)，INQ(ACOS)といった既存大型DBSとから成るDDBSであった〔 TAKIM78，79，80a，b，c，d，JDDBS80 〕。JDDBS-Ⅰにおける成果は次の様である。

1) DDBSの全体アーキテクチャとしての四層スキーマ構造の確立〔 TAKIM78，79 〕
2) DDBS設計システムとしての同種化システムの開発　〔 TAKIM79 〕
3) 〃　の統合化システムの開発　〔 TAKIM80b 〕
4) DDBSアクセスシステムとしての問合せ変換システムの開発　〔 TAKIM80a 〕
5) 〃　の問合せ分割システムの開発　〔 TAKIM80b 〕

これらによって，DDBSの基本概念を確立し，その一部のインプリメンテーションを行なった。

これに対して，1980年春以降開発を進めているJDDBS-Ⅱ（第Ⅱ期JDDBS）は，次の点を新たな特徴としている。

1) 小型個人用DBSの組み込み。
2) ローカルネットワークの適用。
3) オフィス情報システム（OIS）を，JDDBS-Ⅱの具体的アプリケーションとする〔 TANAY81 〕
4) 異種DBS（具体的にはCODASYL DBTG DBS）の統合
（同種化と問合せ変換システム〔 TAKIM79，80a 〕に更新機能を付加する）
5) DDBSにおける統一モデル（オブジェクトモデル〔 TANAY80 〕）の設定
6) DDBSにおける更新問題の解決（オブジェクトモデルに基づいた同時実行制御方式）
7) DD/Dの知識ベース化

図4.1は，JDDBS-Ⅱの全体的なアーキテクチャを示している。図に示す様に，JDDBS-Ⅱは次の2つのモデルから構成されている。

1) スキーマ層モデル
2) 通信処理モデル

スキーマ層モデルは，DDBS内のデータの記述（又はデータの視方）に関している。即ち，通信ネットワーク上に物理的に分散して存在しているデータが，1つの論理的DBSにどの様に仮想化されているかを示している。このスキーマ層モデルとしては，我々の四層スキーマ構造（four-schema structure）〔 TAKIM78，79〕を用いる。

通信処理モデルは，DDBSに対する処理要求（検索，更新）を,DDBSを構成する通信ネットワークとデータベースシステム（DBS）とによって処理するためのモデルである。DDBSにおいては，ある要求を処理するために必要な全てのデータが同一サイトに存在しない場合が一般的である。この場合には，通信ネットワークを用いて各DBSが互いに必要なデータを通信しながら処理，即ち，

通信処理〔 TABAK80 〕, する必要がある。 DDBS の運用において, この通信処理の有効性が全システムのパフォーマンス上重要である。

スキーマ層モデルが, 分散型データベースシステム (DDBS) の仮想化概念を与えるのに対して通信処理モデルは, スキーマ層モデルに基づいた DDBS の実際の処理のモデルである。

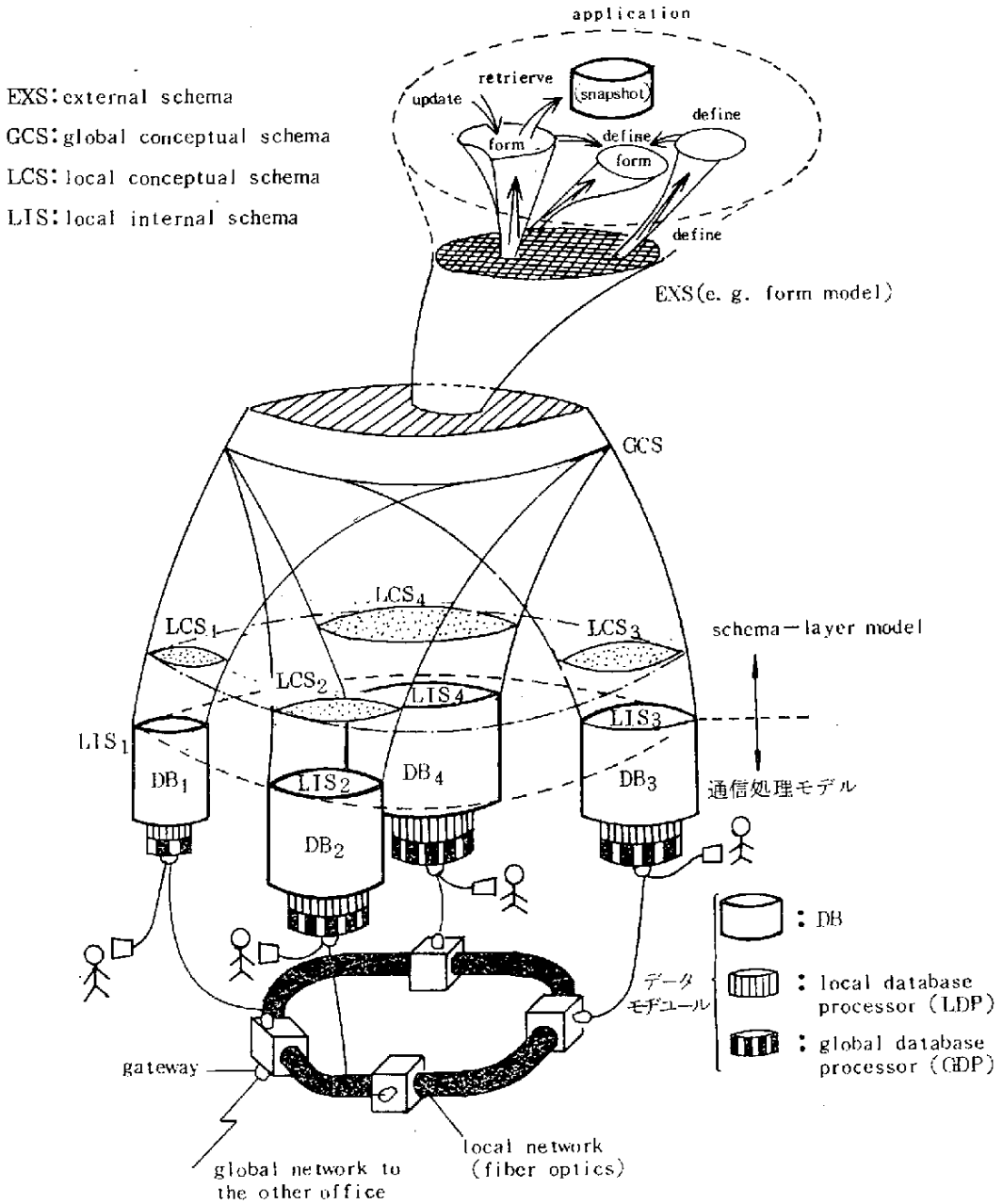


図 4.1　JDDBS－IIアーキテクチャ

4．3 スキーマ層モデル

スキーマ層モデル（ schema-layer model ）は，通信ネットワークによって結合されたデータベースシステム（DBS）内のデータをどの様に１つの論理的DBSに仮想化するかのモデルである。このモデルとして，四層スキーマ構造〔 TAKIM78，79 〕を用いる〔図4.2 〕。

第１章で述べた様に 分散型データベースシステム（ DDBS ）における主要な問題点は，次の２点である。

1） 異種性問題（ heterogeneity problem ）

2） 分散問題 （ distribution problem ）

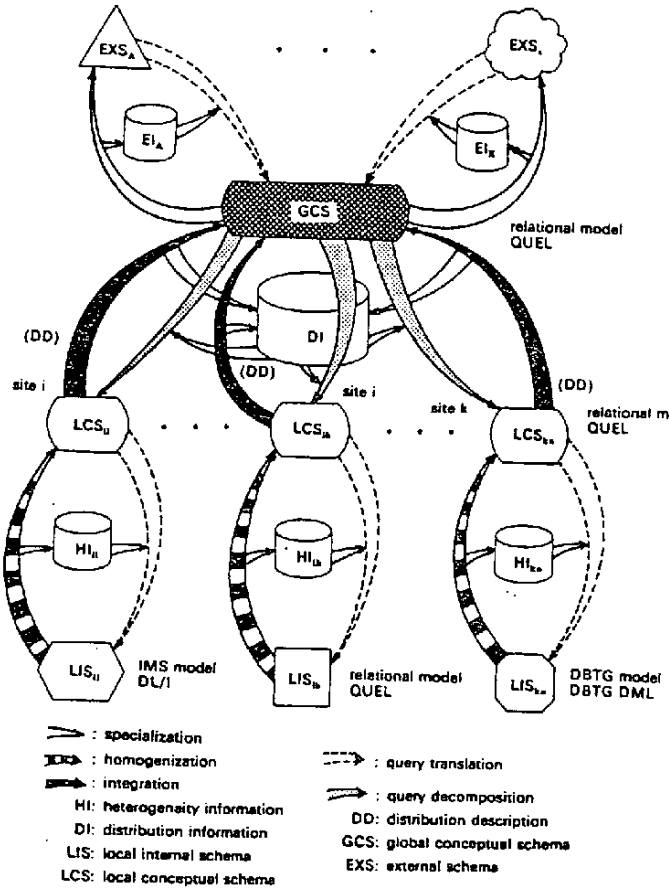


図 4.2 Four-Schema Structure (FSS)

DBSの異種性とは，DBSがDDBS の環境下で利用可能なスキーマ層が提供するデータモデルとその言語の各々の相違であると定義する。1)は，この相違を解決する問題である。2)は仮想化された１つの論理的データベース記述（即ちスキーマ）と，実際にデータベースの存在するサイトのデータベース記述との対応問題である。四層スキーマ構造は，この２つの問題の解決を目指している。

四層スキーマ構造は，次の３つの概念から構成されている。

a） ４つのスキーマ層

b） スキーマ層間のマッピング

c） DD／D

A．スキーマ層

四層スキーマ構造は，次の４つのスキーマ層から成っている。

a） ローカル内部スキーマ（LIS）

各データベースシステム（DBS）が，分散型データベースシステム（DDBS）の環境で提供可能なデータの記述である。 各DBSに固有なデータモデルと言語とによって特徴づけられる。

b） ローカル概念スキーマ（LCS）

DDBS全体で共通なデータモデルと

言語とによって，ローカル内部スキーマ（LIS）を記述したものである。共通モデルとして，リレーショナルモデル〔CODDE70〕と共通言語としてリレーショナル計算言語QUEL〔HELDG 75〕とを採用する。このスキーマ層において，各DBSの異種性問題が解決される。

c) 全体概念スキーマ（GCS）

DDBS全体に対して，ある共同体（community）にとって意味のあるデータの統一的な記述である。このスキーマ層において，DDBSは言語的な1つのDBSに仮想化されたことになる。即ち，分散問題が解決されている。

GCS層のモデルとしては，LCS層と同じくリレーショナルモデルを，言語としては QUELを用いる。

d) 外部スキーマ（EXS）

GCS層のデータの中から，共同体内のあるアプリケーションにとって意味のあるデータを，このアプリケーションに適したモデルと言語とによって記述したスキーマ層である。

例えば，フォームモデル〔TANAY80b〕がある。

B．スキーマ層間のマッピング

スキーマ層間のマッピングとしては，大別して次の2つがある。

1) 設計マッピング

2) アクセス要求マッピング

前者は，Aの4つのスキーマ層にどの様に仮想化していくかの問題であり，後者は，上位のスキーマ層から下位のスキーマ層にどの様にアクセス要求を変換していくかの問題である。

ボトムアップ的な設計マッピングとしては，次の3つのステップがある。

a) 同種化（homogenization）：LIS→LCS〔TAKIM79〕

LISからLCSの生成である。

b) 統合化（integration）：{LCS}→GCS〔TAKIM80b〕

LCSの集合からGCSの生成である。

c) 専用化（specialization）：GCS→EXS

GCSからEXSの生成である。

アクセス要求マッピングとしては，次の2種がある。

a) 問合せ変換（query translation）：LCS→LIS

：EXS→GCS

異なったモデルに基づいたアクセス言語間の変換である。〔TAKIM80a，ch．7〕

b) 問合せ分割（query decomposition）：GCS→{LCS}

同一モデル言語間の変換である。GCS問合せを各LCS上の問合せに変換する〔TAKIM80b，ch．6〕

C．DD/D

DD/Dは，Bのスキーマ層間のマッピング情報と，Aのスキーマ層自身の情報との記述である。

DD/Dとしては，スキーマ層と層間マッピングに対応して次の3種の情報がある。

a) 異種性情報（heterogeneity information or HI）：LIS←→LCS

LISとLCS自身の記述

LISとLCSとの対応情報

b) 分散情報（distribution information or DI）：{LCS}←→GCS

GCS自身の記述

GCSとLCSとの対応情報

LCSの存在サイト（データモデュール）情報

c) 外部情報（external infomation or EI）：GCS←→EXS

EXS自身の記述

EXSとGCSとの対応情報

DD/Dは，LCSレベルでLCSリレーションとして管理される。異種性情報（HI）は，各データベースシステム（DBS）サイトで管理される。分散情報（DI）は，全てのサイトで完全冗長に管理される。外部情報（EI）はアプリケーションの存在するサイトで，完全冗長に管理させる。DD/Dの情報管理の原則は，この情報を用いる処理プロセスの存在するサイトに必要な種類のDD/D情報を配置することである。HIは，問合せ変換システムの存在するところであり，DIは問合せを受けつけるところである。

4．4　通信処理モデル

4.3節で述べたスキーマ層モデルに基づいた（i.e. 四層スキーマ構造）アクセス要求を，どの様にシステムとして処理していくかがDDBS実現上重要となる。DDBSでは複数のデータベースシステム（DBS）が，通信ネットワークによって互いに結合されており，DDBSに対する要求は，一般にこれらの複数のDBSにまたがったものになる。複数DBSサイトにまたがった要求を処理するためには，少なくともあるサイトが必要とする他方のサイトのデータを，このサイトに通信ネットワークを用いて転送しなければならない。このことは，データを"流し"ながら処理させていくという通信と処理との融合，即ち，通信処理〔TABAK80〕の必要性を示している。

本節では，通信処理のモデル化を試みる。まず，分散型データベースシステム（DDBS）は，次の要素から構成されているとする〔図4.1〕。

1) データモデュール（data module or DM）

2) 通信ネットワーク

データモデュール（DM）は，次の2つの要素から成っている。

a) 論理プロセッサ（PS）

b) データベース（DB）

データベース（DB）は，論理的なデータの集合である。論理プロセッサ（PS）は，このデータベース（DB）内のデータに対する処理（e.g. 関係代数演算）機能を備えている。この様にDMは，デ

ータの集合とある処理能力との対から成っている。

通信ネットワークは，各DMを相互に通信させる機能を有している。

四層スキーマ層の各スキーマ層に対して，スキーマモデルをスキーマとその仮想的な外延として定義する。上位層のスキーマモデルが，下位層に存在する論理的に異なったスキーマモデルから構成されているとする。下位層の各スキーマモデルに対して，１つのデータモヂュール（DM)が対応しているとする。この時，下位層に発せられたアクセス要求は，一般に下位層の複数のデータモヂュール（DM）の協働（cooperation）によって処理される必要がある。この協働はDMの処理能力とDM間の通信ネットワークとの通信処理によって行なわれる。通信処理としては次の２種がある。

1) DM自身による通信処理（ DM内通信処理）

DM内のDBに対しての処理である。例として，初期ローカル問合せ処理〔HEONA78a，TAKIM80b 〕，サーチエンジン〔TANAY81〕がある。

2) DM間の通信処理（ DM間通信処理）

他のDB内のデータと自分のDB内のデータとの処理である。例として，サイト間結合処理〔TAKIM80b〕，半結合処理〔BERNP79〕がある。これは，DM間を相互に結合する通信ネットワークを用いて，必要な情報（データ，コマンド）をやりとり（通信）することによって達成される。

A．データモヂュール（DM)

DMは，あるスキーマ層（例えば，LCS層）のスキーマモデル（論理的なデータベース）としてのDBと，処理能力を備えた論理プロセッサ（PS）とから成っている〔図4.3 〕。

i.e. DM ≒ PS ＋ DB

論理プロセッサ（PS）は，次の２種のプロセッサから成っている。

a) DM内通信処理プロセッサ（DPS）

b) DM間通信処理プロセッサ（CPS）

即ち，

PS ≒ DPS （  in Fig.4.1 ）＋CPS （  in Fig.4.1 ）

DPSは，DM内のDBに対する処理を行なう。例えば，DB内のリレーションの結合，制限，更新演算を行なう。CPSは，他のDMから転送されてくるリレーションを受信するとともに，自分のDB内のリレーションの必要な処理（ e.g. 結合）を行ない，必要な結果を他のDMに送る。この様にCPSは，他のDMとの通信とともに処理を行なう。

DBは，あるスキーマモデルに対応している。分散型データベースシステム（DDBS)では，DBはローカル概念スキーマ（LCS）層のスキーマモデルとなる。DBの要素の集合がDM間通信処理の単位となる。例えば，DDBSでLCSがリレーショナルモデルに基づいているならば，リレーション（組の集合）が転送単位となる。DBは他のDMから転送されてくるデータのバッファの役目も持っている。

図4.1で，全体データベースプロセッサ〔JDDBS80〕(GDP）は，CPSの役目をもち，ローカ

ルデータベースプロセッサ（LDP）は，DPSの役目を持っている。

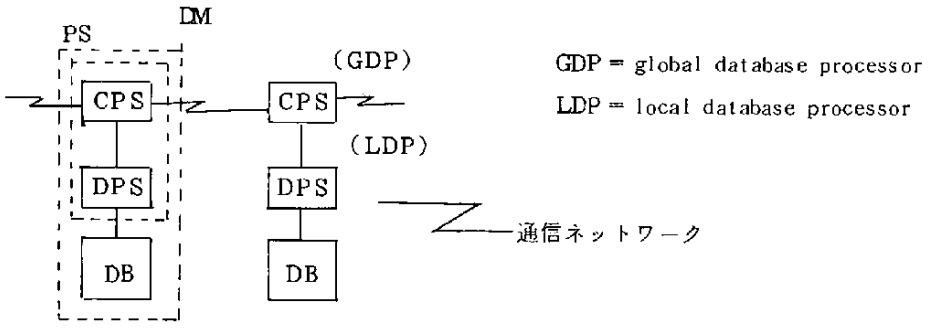


Fig. 4.3　DMの構成

B. 通信ネットワーク

通信とは，上位層（即ち，全体概念スキーマ（GCS））におけるアクセス要求を処理するために必要なデータモデュール（DM）の協働を達成するための情報（データと制御情報）のやりとりである。DB（ローカル概念スキーマ（LCS）モデル）の要素がDM間のデータ通信の単位となる。LCSがリレーショナルモデルに基づいているので，リレーションが通信処理の単位となる。DM間での通信は，この間に物理的に存在する通信ネットワークを介して行なわれる。

通信ネットワークとしては，次の2種類がある。

a) 広域ネットワーク（ global network ）

パケット交換技術を用いており，中～低速（～50 kbps ）である。通信実体間の1対1の通信を提供する。

例　　ISO／SC16の7層モデル〔BACHD78〕　DCNA(NTT)　SNA(IBM)

b) ローカルネットワーク

ローカルネットワークは，次の様な特徴を備えている〔COTTI80〕。

i) 高帯域（≧1Mbps　～15Gbps も85年頃までに可能）

ii) 高信頼性（ 10^{-12} の誤り率）　　iii) 近距離（数km以内）

iv) 単一組織による運用管理（1つのオフィス）

更に，共有媒体ネットワーク（e.g. Ethernet , MITRENET, NBSNET ）では，高速高信頼な1つの通信実体と複数の通信実体間の通信（1：N通信）が可能である。

JDDBS-Ⅱでは，共有媒体ローカルネットワークを用い様としている。この理由としては，次の点がある。

1) 広域通信ネットワークにおける通信ボトルネックの解消。

2) DBS間通信は本来的に1：Nである。

3) OIS等のデータベース利用は，1つの組織（部局）内でのローカリティが高い。

C. 通信処理の参照モデル

DDBSにおける通信処理の階層構造を，参照モデルと呼ぶ。この参照モデルは，図4.4の様に

示すことができる。応用アクセス層は，各種応用アクセスモデュールがあり，DDBSに対するアクセスを行なう。DM間通信処理層は，データモデュール間の通信処理を行なう層である。応用アクセスモデュールに対して，DDBSの分散性をinvisibleにするサービスを提供している。最下層のDM内通信処理層は，各DM内で閉じて処理を行なう層である。〔TABAK80〕は，4層の階層構造を提案している。

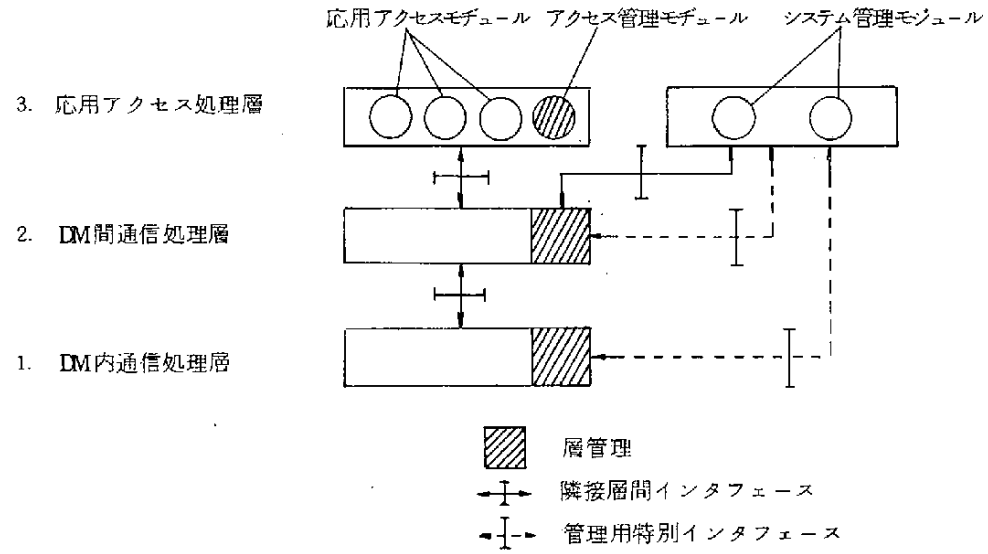


図4.4 通信処理の参照モデル

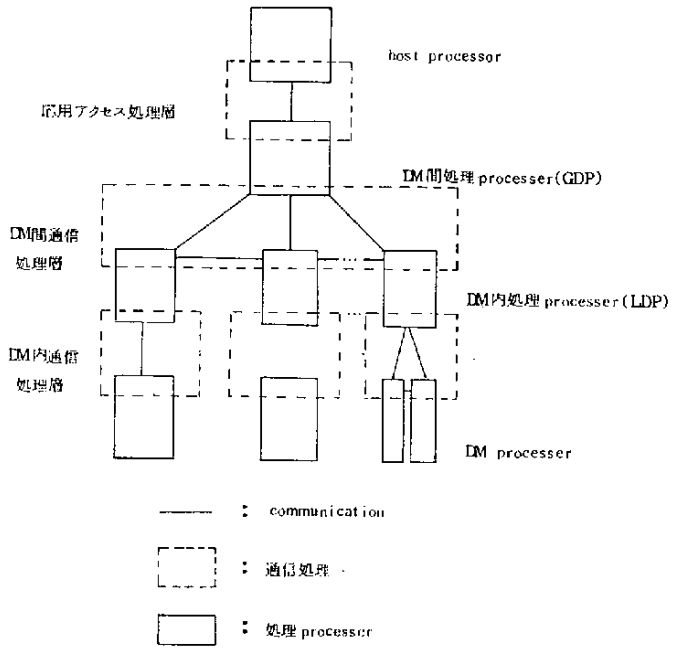


図4.5 通信処理

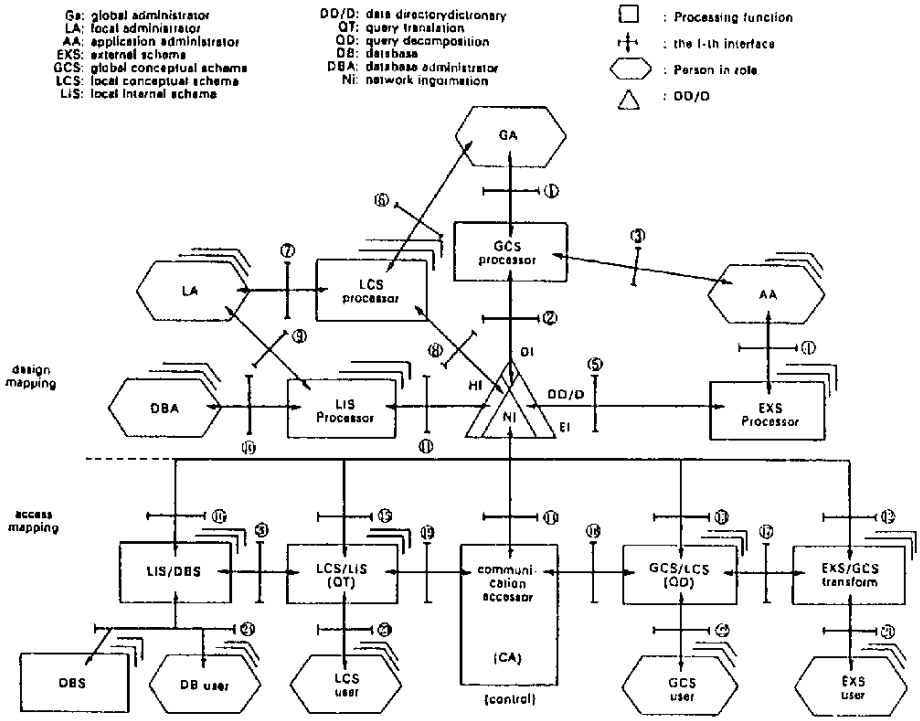


図4.6　DDBSのアーキテクチャ

5. 問合せ処理

DDBSでは，全体概念スキーマ（GCS）層に出されたアクセス要求（GCS問合せを呼ぶ）をローカル概念スキーマ（LCS）層の問合せ表現に変換し，各データモデュール（DM）間及びDM内で必要な通信処理を行ない，求める結果を得なければならない。この問合せ処理は，次の2つの主要なプロセスからなっている。

1) 問合せ表現変換

2) 通信処理

問合せ表現変換とは，GCSリレーションのみを参照するGCS問合せを，LCSリレーションを参照する問合せ表現に変換することである。GCSリレーションがLCSリレーションのビュー（view）として定義されているので，検索要求については，問合せ変形（query modification）〔STONM76〕手法を用いて変形できる。表現変換されたLCSリレーションのみを参照する問合せを，全体LSC問合せと呼ぶ。

全体LCS問合せは，一般に複数のDM上のLCSリレーションを参照している。従って，これらのDM間及びDM内での通信処理が必要となる。DDBSにおいて，通信ネットワークがその容量の点から主要なパフォーマンスボトルネックを形成している。ARPANET等の広域パケット変換ネットワークの通信速度は，高々50 kbpsであり，現在のチャネル速度が1 MB／secである。光ファイバーを用いたローカルネットワークでは，1G bps～数10Gbpsの通信路の共用が可能となるだろうが，この時高速転送された大量データに対する処理がボトルネックとなると考えられる。このため通信処理の最適化は，DDBSの運用上重要な問題となる。

本章では，DDBSにおける問合せ処理の基本仮定（5.1），処理の目標関数（5.2），戦略（5.3）について述べる。問合せ表現を5.4で，問合せの表現変換を5.5で触れる。5.6では初期ローカル問合せ処理について論じて，この中で問合せの通信処理におけるDM間通信処理とDM内通信処理の位置づけを明らかにする。前者については6章で，後者については7章で各々論じる。

5.1 基本仮定

DDBSでの問合せ処理を考えるうえで，次の仮定を設ける。

1) 問合せとしては検索のみを考える。更新は，更新すべきデータの所在を見つける検索に続く，格納演算（追加，消去，修正）と考えられる。更新を考えるうえでは，次の2点を考えねばならない。

   i) ビュー更新

   ii) 同時実行及び信頼性制御

   これらは困難な問題であり，まず基本となる検索について考え，更新は今後の課題としたい。ビュー更新については，付記1を参照されたい。

2) 問合せ言語としては，QUEL〔HELDG75〕を用いる。QUEL問合せの条件式（q1）は，次の形

式をしているとする〔TAKIM80a, b〕。

$$ql ::= qual_1 \vee qual_2 \vee \cdots\cdots \vee qual_h$$

$$qual_i ::= C_{i1} \wedge C_{i2} \wedge \cdots\cdots \wedge C_{in_i} \quad (\text{for } i = 1, 2, \cdots, h)$$

$$C_{ij} ::= c\text{-}pred_{ij1} \vee c\text{-}pred_{ij2} \vee \cdots\cdots \vee c\text{-}pred_{ij\,l_{ij}}$$

$$(\text{for } i = 1, 2, \cdots, h, \quad j = 1, 2, \cdots, n_i)$$

ここで，$c\text{-}pred_{ijk}$は，制限又は結合論理式である。

又，各C_{ij}内の$c\text{-}pred_{ijk}$($k = 1, \cdots, l_{ij}$)は全ての同一の組変数を参照していなければならない。問合せは，各$qual_i$に対応して生成され，各々独立に処理され，各々の結果の和が求める解となる。従って，以降$qual_i$を条件式に持つ問合せ（ i．e．積正規化された問合せ）を考えることにする。

3) 通信ネットワークは，高信頼な1対N通信機能を物理的に備えたローカルネットワークとする。通信コストは時間で表わすものとする。1対N通信ネットワークとは，次のコスト関数が成り立つネットワークとする。即ち，xを転送データ量とした時，$C_i\{j_1, j_2, \cdots, j_n\}(x)$を$DM_i$からn個のDM($j_1, \cdots, j_n$)に転送するコストとすると，次の式が成り立つことである。

$$C_i\{j_1, j_2, \cdots, j_n\}(x) = C_i\{j_1\}(x) = C_i\{j_2\}(x) = \cdots\cdots = C_i\{j_n\}(x)$$
$$= C_i\{j_1, j_2\}(x) = \cdots\cdots = C_i\{j_2, \cdots j_n\}(x)$$
$$= a + bx \qquad \cdots (2)$$

ここでaとbとは定数である。(2)式は，通信コストは，転送されるDMの数，距離には独立で，ただ量だけに比例することを示している。

4) 各DMはデータベース内に動的な情報を保持している。従来のDBSが検索中心の利用であったのに対して，オフィス情報システム等の今後のアプリケーションにおいては更新利用が増加していくものと思われる。本論文では，検索問合せのみを考えているが，各DMが動的データを保有することは，問合せ処理において主要な役割を持つDD/D情報の管理に影響がある。即ち，DD/D情報自体も動的なものとなる。

5) ネットワークは軽負荷とする。共有媒体上のコンテンションによる遅延は存在しないものとする。

6) ユーザのGCS問合せを受けとったDMを，結果データモデュールと呼ぶ。この問合せの結果は，最終的に結果DMに集められユーザに出力される。

5．2　目標関数

問合せ処理の目標としては，次の3点がある〔TAKIM80b〕。

1) 全通信量の最少化〔HEVNA78a〕
2) 応答時間の最少化〔HEVNA78a〕
3) DD/D情報の最少化と静化

分散型データベースシステム（DDBS）において，通信ネットワークが主要なボトルネックとなることから，通信トラフィックをなるべく減少させる必要がある。このために1)の目標達成は重要である。

DDBSを構成するデータモデュール（DM）は，データベースとともに，処理能力を備えている。このため，データベースマシンにおける様に，各DMでの処理のパラレリズムを高めることによって，応答時間を減少できる。利用者からみて，応答時間の短縮は望ましい。

最後の目標は，問合せを処理するためのDD/D情報（i.e. 分散情報と異種性情報）の管理をなるべく簡単化するためのものである〔TAKIM80b〕。問合せの処理を行なうDMは，このための情報（DD/D）を保持する必要がある。これらの情報が動的な性質を持つならば，各DM間でDD/D情報の一致性を保つための膨大な通信オーバヘッドが生じてしまう。いわゆる冗長コピーに対する同時実行制御問題〔BERNP80〕が生じてしまう。DD/D情報の中で，異種性情報（HI）は，各DMで管理されるので，分散情報（DI）の保持方法が問題となる。DIのなかで，GCSリレーションとLCSリレーションとの対応情報（我々はこれをスキーマ情報と呼ぶ）は，一般に比較的長いライフサイクルを持つために静的な性質を備えていると言える。一方，LCSリレーションのオカーランス情報（e.g. カーディナリティ，選択度）は，LCSリレーションが動的であることから，これを反映して動的なものとなってしまう。我々は，OISにおいては，より更新利用がなされると考えていることから，これらの動的な情報（パフォーマンス情報）を冗長に，かつ一致性を保持することは困難である。これらの情報に基づいた決定を正確なものとするためには，パフォーマンス情報のより正確な保持が必要になる。しかし，パフォーマンス情報の正確な保持は，より大きな通信オーバヘッドを必要とする。

我々は，DDBSの運用上から，DD/Dとしてパフォーマンス情報を持たせない方針をとっている。このことによって，DD/D管理を簡単化するとともに，各DMでのDD/Dの格納オーバヘッドを減少できる。この第3番目の目標達成は，DDBSの運用管理上最重要であると確信している。

5.3 基本戦略

以上述べてきた仮定のもとで目標を達成するための問合せ処理の基本戦略を以下に示す。

1) GCS問合せを問合せ変形手法を用いて，全体LCS問合せに変形する。
2) 問合せの処理に関連する全DMが，完全な分散制御によって処理を実行する。ローカルネットワークを用いているために，あるDMから出されたメッセージを全DMで受信できる。
このため，全てのDMは，他の全てのDMの状態をモニターできるために，どれか1つのDMをマスタとし他をスレーヴとする必要はない。
3) 通信トラフィックを減じるために，各DM内でローカルに処理できる部分をまず処理してしまう。これは初期ローカル問合せ処理〔HEVNA78a〕と呼ばれるものであり，DM内通信管理に相当している。DM内通信処理では，このDMで実行可能な言語に変換し実行させる〔第6章と〔TAKIM80a〕を参照〕。
4) 通信トラフィックを減じるために，DM間処理に必要な部分のみを転送する。このために，

semi-join〔BERNP79, GOODN79〕手法を用いる。

5) より小さいデータを転送させる。DM間での結合（join）処理においては，より小さいリレーションを，より大きいリレーションに転送する。

問合せ処理の概要を図5.1に示す。まずGCS問合せは，問合せ変形手法を用いて，対応するLCSリレーションのみから成る全体LCS問合せを生成する。

次に全体LCS問合せ内のなかで，各DMで独立して処理できる部分を処理してしまう。これは，DM内通信処理である。しかる後に，通信ネットワークと各DMとを用いて，DM間通信処理を行なう。DM内通信処理については第7章で，DM間通信処理については第6章で各々論じる。

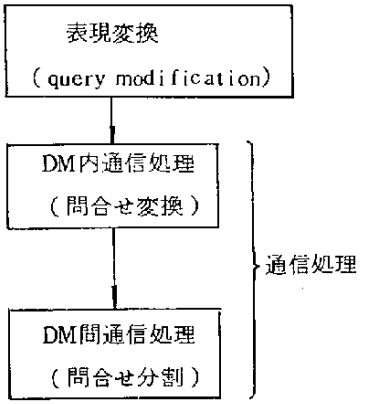


図5.1　問合せ処理の概要

5.4　問合せ表現と問合せグラフ

リレーションに対する問合せqは，QUEL〔HELDG75〕によって次の様な形式で記述される。

$$\left.\begin{array}{l} \underline{\text{range}}\ (x_1, X_1)\ (x_2, X_2) \cdots\cdots (x_m, X_m)\ ; \\ q:\ \underline{\text{retrieve into}}\ R\ (r_1 = aexp_1, \cdots, r_k = aexp_k) \\ \qquad\quad \underline{\text{where}}\ \ \text{qual}\ ; \end{array}\right\} \cdots(1)$$

ここで，　R　≙　問合せqの結果リレーション名

r_j　≙　結果リレーションRのj番目の属性（$j = 1, 2, \cdots, k$）

$aexp_j$　≙　結果属性r_jに対する$x_1, \cdots, x_m$上に定義される算術式（$j=1, 2, \cdots, k$）

X_i　≙　リレーション　（$i=1, 2, \cdots, m$）

但し，各X_iは互いに異なっている必要はない。

x_i　≙　リレーションX_iに対する組変数（$i=1, 2, \cdots, m$）

VAR(q)　≙　$\{ x_i \mid i = 1, 2, \cdots, m \}$

≙　問合せqの参照する組変数の集合

qual ≙ 問合せqの条件式

条件式(qual)は，次の形式をしている。

$$\text{qual} \triangleq c_1 \ (\wedge c_2 \wedge \cdots \wedge c_n)$$

$$c_i \triangleq c\text{-}pred_{i1} \ (\vee c\text{-}pred_{i2} \vee \cdots \vee c\text{-}pred_{il_i}) \quad \cdots(2)$$

$c\text{-}pred_{ij}$ は，2項比較述語を表わしている。比較述語としては，制限述語と，結合述語とがある。即ち，

$$c\text{-}pred_{ij} :: = rp(x_k) \mid jp(x_k, x_l) \quad \cdots\cdots(3)$$

ここで，x_k，x_l は組変数である。(i.e. x_k，$x_l \in VAR(q)$)。

rp(x_k) ≙ 変数 x_k を参照する制限述語

≙ $x_k.a \ \theta \ v$ 又は $x_k.a \ \theta \ x_k.b$

ここで $\theta \in \{<, \leq, =, \geq, >, \neq\}$

a，bはリレーションX_kの属性

jp(x_k, x_l) ≙ 変数 x_k と x_l ($x_k \neq x_l$) に関する結合述語

$x_k.a \ \theta \ x_l.b$

ここで aはx_kの，bはx_lの属性

この時，問合せの条件式は次の制約を持っている。

制約1 問合せの条件式(qual)は，積正規形をしている。

(i.e. $\text{qual} :: = c_1 \wedge c_2 \wedge \cdots \wedge c_n$)

制約2 条件式(qual.)の各conjunct (c_i) 内の全ての比較述語は，制限述語か，結合述語か，のどちらかであり，かつ同一の変数を参照していなければならない。

i.e.

$$c_i :: = rp_{i1}(x_k) \vee rp_{i2}(r_k) \vee \cdots \vee rp_{il_i}(x_k) \mid jp_{i1}(x_k, x_l) \vee jp_{i2}(x_k, x_l) \vee \cdots \vee jp_{il_i}(x_k, x_l)$$

上述してきたLCS問合せ(q)に対して，次の記号を定義する。

vatt(aexp)≙ 算術式 aexp が参照するLCS属性 ($x_i.a$) の集合

TL(q) ≙ 問合せ(q)の結果属性で定義するLCS属性(目標属性と呼ぶ)の集合

$$\triangleq \bigcup_{i=1}^{k} vatt(aexp_i)$$

$tl(x_i)$ ≙ 変数 x_i に関する目標属性

$$\triangleq \{ x_i.a \mid x_i.a \in TL(q) \}$$

$rf(x_i)$ ≙ 変数 x_i に関する制限論理式

$$\triangleq \bigwedge_i c_i \quad \text{ここで}$$

$$\begin{cases} c_i = rp_{i1}(x) \vee \cdots \vee rp_{il_i}(x) \\ \qquad \underline{if} \quad x = x_i \\ c_i = T \quad \underline{if} \quad x \neq x_i \end{cases}$$

$jf(x_i, x_j)$ ≙ 変数 x_i と x_j とに関する結合論理式

≙ $\bigwedge_k c_k$ ここで

$$\begin{cases} c_k = jp_{k1}(x, y) \vee \cdots \vee jp_{kl_k}(x, y) \\ \qquad \underline{if}\ (x = x_i \wedge y = y_j) \vee \\ \qquad\quad (x = x_j \wedge y = x_i) \\ c_k = T \quad \underline{otherwise} \end{cases}$$

$range(x_i)$ ≙ 変数 x_i の range リレーション

≙ X_i

これらの記号を用いて，(1)の問合せ書き直すと(4)の様になる。

$$\left.\begin{array}{l} \underline{range}\ \{(x_i, X_i) \mid x_i \in VAR(q) \wedge X_i = range(x_i)\}; \\ \underline{retrieve}\ \underline{into}\ R\ (\{\ tl(x_i) \mid x_i \in VAR(q)\ \}) \\ \underline{where} \\ \qquad \bigwedge_{\forall x \in VAR(q)} \left[rf(x) \wedge \bigwedge_{\forall g \in VAR(q)(q \neq x)} jf(x, y) \right]; \\ ここで \quad VAR(q) = \{\ x_1, x_2, \cdots, x_m\ \} \end{array}\right\} \cdots\cdots (4)$$

(4)式の様なLCS問合せは，グラフによって表わせる。これをリレーショナル問合せグラフ(RQG)と呼ぶ。RQGのノードは，LCS問合せの参照する組変数を表わし，RQGノードに関するリンクは制限式，結合式，目標属性を表わしている。RQGは次の様に定義できる。

RQG ≙ (VN , LNK)

VN ≙ $\{\ n_i \mid i = 1, 2, \cdots, m\}$

≙ RQGノードの集合

n_i ≙ (x_i , $range(x_i)$)

≙ RQGノード

組変数 x_i に対応している ($x_i \in VAR(q)$)

ここで，$node(x_i) = n_i$ なら1対1対応関数 (node:VAR→VN) を定義する。

LNK ≙ (TLNK , PLNK JLNK)

TLNK ≙ $\{\ tl_i \mid i = 1, 2, \cdots, m\}$

≙ 目標属性リンクの集合

tl_i ≙ (x_i , $tl(x_i)$)

≙ $tl(x_i) \neq \phi$ なるノード $node(x_i)$ に存在し，x_i に関する目標属性リストを表わしている。

RLNK ≙ $\{\ rl_i \mid i = 1, 2, \cdots, m\}$

≙ 制限リンクの集合

rl_i ≙ (x_i , $rf(x_i)$)

≙ $rf(x_i) \neq T$ なるノード $node(x_i)$ に存在し，x_i に関する制限論理式 $rf(x_i)$ を表わす。

JLNK ≙ $\{ jl_{ij} \mid i, j = 1, 2, \cdots, 3 \ (i \neq j) \}$

≙ 結合リンクの集合

jl_{ij} ≙ $(x_i, x_j, jf(x_i, x_j))$

≙ $jf(x_i, x_j) \neq NIL$ なるノード $node(x_i)$ と $node(x_j)$ との間に存在し，結合論理式 $jf(x_i, x_j)$ を表わす。

ここで $jl_{ij} = jl_{ji}$

これらは図5.2の記号を用いて図示できる。RQGの結合リンクと制限リンクが表わす論理式の積は，問合せqの条件式を表わしている。

即ち，

$$qual = \bigwedge_{n_i \in VN} rf(node^{-1}(n_i)) \cdot \left[\bigwedge_{\substack{Vn_j \in VN \\ (n_i \neq n_j)}} jf(node^{-1}(n_i), node^{-1}(n_j)) \right] \cdots (5)$$

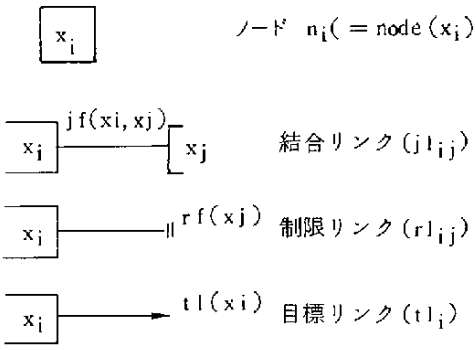


図5.2　RQGの記号

又，目標リンクの表わす目標属性の集合は，問合せqの目標属性の集合である。

即ち，

$$TL(q) = \{ tl(node^{-1}(n_i)) \mid n_i \in VN \wedge tl_i \neq \Lambda \} \quad \cdots\cdots (6)$$

従って，RQGはリレーショナルLCS問合せq(1)と等価である。

5.5　表現変換－問合せ変形

表現変換とは，GCSリレーションを参照するGCS問合せを，問合せ変形〔STONM76〕を用いて，対応するLCSリレーションのみを参照する全体LCS問合せに変形することである。

この表現変換のためのGCSリレーションとLCSリレーションとの対応情報は，分散情報（DI）内のスキーマ情報である。各DMはこの情報の完全コピーを保持している（full redundancy）。

GCS問合せを受け取った目標DMは以下のことを行なう。

1) GCS問合せの構文及び意味チェックを行ない，内部表現に変換する。
2) GCS問合せの参照するLCSリレーションを明らかにする。これらの定義式を，分散情報から見つけて，問合せ変形を行なう。
3) 2)で生成された全体LCS問合せを，5.1の(1)式の様に正規化する。
4) 問合せ条件式の各disjunct（(1)式のqual）に対応して，問合せを生成する。これらを全DMに放送（broadcast）し，通信処理を行なわせる。

問合せ変形については以下に述べる。GCSリレーションの定義式は，(1)式の様に書ける。

```
range ( l1 , L1 : s1 ) ( l2 , L2 : s2 ) … ( lm , Lm : sm ) ;
{ define Gi ( gai1 = aei1, gai2 = aei2, …, gai li = aei li )
  where qual i ;                                              | i = 1, 2, …, n }
                                                              …… (1)
```

ここで，l_i(i = 1, 2, …, n)は，LCSリレーションL_iに対する組変数である。s_iはL_iの存在するDM名である。i ≠ j なるiとjに対して$l_i = l_j$であるが，必ずしも$L_i \neq L_j$とは限らない。G_iは，GCSリレーション(i = 1, 2, …, n)である。ga_{ij}はG_iのj番目の属性であり，ae_{ij}はこれの定義式である。qualiはG_iの条件式である。nはGCSリレーション数で，$G_1, G_2, \cdots, G_n$の参照する全組変数は$\{ l_1, l_2, \cdots, l_m \}$である。

この時，GCSリレーション$G_1, \cdots, G_n$に対するGCS問合せは，(2)式の様になる。

```
range   ( g1, G1 ) ( g2, G2 ) …… ( gn, Gn ) ;
retrieve into   R ( r1 = gae1 , r2 = gae2 , …, rk = gaek )        (2)
      where   qual ;
```

ここで，置換$\sigma = \{ t_1/v_1, t_2/v_2, \cdots, t_p/v_p \}$を考える。Eをある表現とすると，$\sigma$Eは，E内の変数$v_i$を式$t_i$(i = 1, 2, …, p)によって同時に置き換えた式となる。
これを用いると(2)のGCS問合せを変形したものは，(3)式の様に表わせる。

```
range   ( l1, L1 : s1 ) ( l2, L2, s2 ) …… ( lm, Lm , sm ) ,
retrieve into   R ( r1 = σ•gae1, r2 = σ•gae2 , rk = σ•gaek )      (3)
      where   σ•qual ∧ qual1 ∧ qual2 ∧ … ∧ qualm ;
ここで  σ = { aeij / gi.gaij | i = 1, 2, …, m,  j = 1, 2, …, li }
```

問合せ(3)は，更に次の様に正規化させる。

```
range   ( l1, L1 : s1 ) …… ( lm, Lm : sm ) ;
retrieve into   R ( r1 = aexp1, r2 = aexp2 , …… , rk = aexpk )
      where   qual1 ∨ qual2 ∨……∨ qualq ;  ( q ≥ 1)                 (4)
ここで  qual i :: = ci1 ∧ ci2 ∧ … ∧ ci li   ( li ≥ 1 )
      cij :: = predij1 ∨ predij2 ∨ … ∨ predijmij  ( mij ≥ 1 )
```

2.1で述べた様に，各c_{ij}は同一変数を参照する述語($pred_{ijh}$)から成り立っている。
この時，(4)式から次の問合せが生成される。

$$\left\{\begin{array}{l}\underline{\text{range}}\ \ (l_1, L_1 : s_1)\cdots(l_m, L_m : s_m)\ ;\\ \underline{\text{retrieve into}}\ \ R_i(r_1 = \text{acxp}_1, \cdots, r_k = \text{aexp}_k)\\ \qquad\underline{\text{where}}\ \ \text{qual}_i\ ;\\ \text{この時}\ \ R = R_1 \cup R_2 \cup \cdots \cup R_q\end{array}\ \middle|\ i = 1, 2, \cdots, q\right\} \quad \cdots\cdots (5)$$

(5)内の各問合せは独立に処理され，問合せ結果は，個々の問合せ結果リレーションの和となる。以降，(5)内の1つについて考えることにする。この問合せを新ためて全体LCS問合せと呼ぶ。

5．6　初期ローカル問合せ処理

問合せ変形によって生成された全体LCS問合せは，結果DMから，この問合せに必要なLCSリレーションを所有している全DMに転送される。全体LCS問合せを受信した各DMは問合せを解析して，自分のDM内でローカル処理出来る部分に対して，初期ローカル問合せを生成する。

この初期ローカル問合せは，各DMで次の様にして行なわれる。

1) 全体LCS問合せから問合せグラフを生成する。
2) 問合せグラフをサーチして，自分のDM（s_iとする）内の変数ノードを集める。
3) これらの変数ノード間の結合リンクをサーチして，結合リンクによって連結なノードの集合をつくる。このs_i内のj番目のグループ（G_{ij}）内のノードをv_{ijk}とすると，この連結なグループG_{ij}に対する初期ローカル問合せq_{ij}は，(1)式の様になる（$k=1, 2, \cdots, m_{ij}$）。
4) 生成されたq_{ij}を実行する。DMがリレーショナルDBSでなければ，問合せ変換〔7章〔TAKIM 80a〕〕によって実行可能な言語に変換して実行させる。〔DM内通信処理〕。
5) q_{ij}の実行は，結果リレーションR_{ij}をDM内に生成する。
6) R_{ij}についてDM間通信処理を行なう。

各DMでの初期ローカル問合せq_{ij}の実行は，問合せグラフの縮退をもたらす。縮退された問合せグラフは，ノードとしてR_{ij}をもち，ノード間の結合リンクが存在する。他のノードとの結属性は$jtl(l_{ij^1}) \cup \cdots \cup jtl(l_{ijm_{ij}})$である。ただし，制限リンクは存在しない。

$$\left.\begin{array}{l}\underline{\text{range}}\ \ (l_{ij^1}, L_{ij^1} : s_i)\cdots(l_{ijm_{ij}}, L_{ijm_{ij}} : s_i)\ ;\\ q_{ij}:\ \underline{\text{retrieve into}}\ \ R_{ij}(\{tl(l_{ij^1}) \cup tl(l_{ij^2}) \cup \cdots \cup tl(l_{ijm_{ij}}) \cup\\ \qquad\qquad jtl(l_{ij^1}) \cup\ jtl(l_{ij^2}) \cup \cdots \cup jtl(l_{ijm_{ij}})\}\\ \qquad\underline{\text{where}}\ jc(l_{ij^1}, l_{ij^2}) \wedge jc(l_{ij^1}, l_{ij^3}) \wedge \cdots\cdots\\ \qquad\qquad\qquad\qquad \wedge\ jc(l_{ijm_{ij-1}}, l_{ijm_{ij}}) \wedge\\ \qquad\qquad rc(l_{ij^1}) \wedge rc(l_{ij^2}) \wedge \cdots \wedge rc(l_{ijm_{ij}})\ ;\end{array}\right\} \cdots\cdots (1)$$

ここで,

l_{ijk} ≜ DM s_i 内の互いに連結な j 番目のグループ (G_{ij}) 内の k 番目の組変数
($k = 1, 2, \cdots, m_{ij}$, $m_{ij} \geq 0$)

L_{ijk} ≜ l_{ijk} の range リレーション

$tl(l_{ijk})$ ≜ l_{ijk} に関する目標属性集合 $tl(l_{ijk}) \in TL(q)$
目標属性の存在しない時は, $tl(l_{ijk}) = \phi$

${}_jtl(l_{ijk})$ ≜ $\{ l_{ijk}.a \mid l_{ijk}.a \in var({}_jc(l_{ijk}, l_{i'j'k'})) \wedge (i, j, k) \neq (i', j', k') \}$

≜ l_{ijk} の属性で, 他のグループ又は他の DM のリレーションとの間の結合属性であるものの集合。

6. 問合せ分割 — DM間通信処理

本章では，データモヂュール（DM）間での問合せの通信処理について論じる。

DM内通信処理によって，問合せは，DM間の結合（join）と結果属性リンクとだけからなる形に縮退されている。この問合せは，次の様に表わせる。

range $(l_{11}, L_{11} : S_1)(l_{12}, L_{12} : S_1) \cdots\cdots (l_{2}, L_{21} : S_2) \cdots\cdots (lmp_m, Lmp_m : Sm)$;

retrieve into $R(t1(l_{11}), \cdots, t_l(lmp_m))$

where $\bigwedge_{i=1,m} \bigwedge_{j=1,p_i} \bigwedge_{k=i+1,m} \bigwedge_{l=1,p_k} jc(l_{ij}, l_{kl})$;

ここで

$jc(l_{ij}, l_{kl})$は，l_{ij}とl_{kl}間での結合式である。存在しなければtrueを真理値として取る。

$t1(l_{ij})$は，l_{ij}に関する結果属性のリストである。存在しなければφである。

この問合せを処理するためには，DM間を結合する通信ネットワークを用いてデータを転送し，結合演算を行なう必要がある。この通信処理は，5.2節で述べた目標を達成する様に行なわれる必要がある。通信ネットワークとしては，1：N通信機能を備えたローカルネットワークを用いているので，この機能を有効に利用することが必要である。本章では，1：N通信を用いた場合の通信処理アルゴリズムを提案するとともに，従来の1：1通信に基づいたアルゴリズムとの比較を通して，その評価を試みる。

まず，6.1では，DM間結合処理の基本となる半結合（semi－join）〔BERNP79〕について述べる。6.2では，単一の結合属性のみから成る単純問合せの場合の通信処理アルゴリズムを示し，その例を6.3に示す。6.4では〔HEVNA78b〕の結果との比較を行なう。6.5では更に一般問合せに対するアルゴリズムを示す。

6.1 半結合（Semi－join）〔BERNP 79〕

R_1とR_2とを2つのリレーションとする。AとBを各々R_1とR_2との属性とする。この時R_1とR_2のAとBとのθ－joinは，次の様に定義される。

$$R_1 [A \theta B] R_2 = \{ r_{12} \mid r_{12} \in R_1 \times R_2 \wedge r_{12}[A] \ \theta \ r_{11}[B] \} \quad \cdots\cdots (1)$$

これに対して，R_2によるR_1のAとBについての半結合は，次の様に定義される。

$$R_1 \ltimes_{A \theta B} R_2 = \{ r_1 \in R_1 \mid (\exists r_2 \in R_2)(r_1[A] \ \theta \ r_2[B]) \} \quad \cdots\cdots (2)$$

よって $R_1 \ltimes_{A \theta B} R_2 = R_1 [A \theta B] (R_2[B]) = R_1 [A \theta B] (R_2[B])$ ……(3)

ここで，R_1とR_2とが，各々DM_1とDM_2（$DM_1 \neq DM_2$）に存在するとすると，(2)は次のことを意味している。

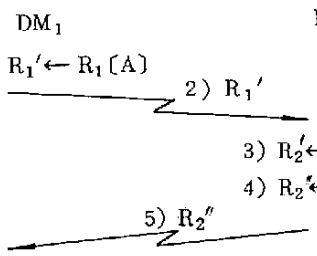


図 5.1　半結合 $R_1 \langle A \theta B \rangle R_2$

1) R_1 のAについての射影をつくる。

$R_1' \leftarrow R_1 〔A〕$;

2) R_1' を DM_2 に転送する。

3) DM_2 で，R_1' と R_2 との結合を行なう。

$R_2' \leftarrow R_1' 〔A \theta B〕 R_2$;

4) この結果のAについての射影をつくる。

$R_2'' \leftarrow R_2' 〔A〕$;

5) R_2'' を DM_1 に転送する。

6) DM_1 で，R_1 と R_2'' との結合を行なう。

$R_1'' \leftarrow R_1 〔A=A〕 R_2''$;

7) R_1'' を結果属性（RAT）について結合を行なう。

$R_1'' \leftarrow R_1'' 〔RAT〕$;

これに対して，(1)の結合では，図 6.2 の様に，一方のリレーションが，他方に転送され，そこで2つのリレーションの結合を行なわねばならない。

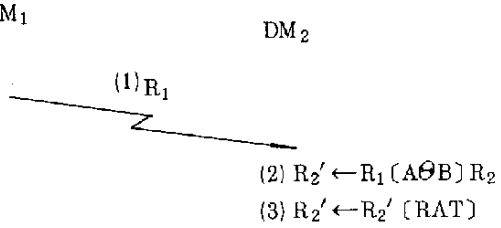


図 5.2　結合 $R_1 〔A \theta B〕 R_2$

従って，半結合の場合の通信コスト C_{sj} は，次の様になる。

$$C_{sj} = |R_1'| + |R_2''|$$

ここで　$R_2'' \subseteq R_1'$ であるので，

$$C_{sj} = |R_1'| + |R_2''| \leq 2 \cdot |R_1| = 2 \cdot |R_1〔A〕| \quad \cdots\cdots (1)$$

ここで　R は，リレーション R の通信コスト(i.e. サイズ)を表わすものとする。

結合の時の通信コスト C_j は，次の様になる。

$$C_j = |R_1| \quad \cdots\cdots (2)$$

R_1 の全属性集合を Ω_{R1} とすると，$|R_1〔\Omega_{R1}-A〕| \geq |R_1〔A〕|$ が成り立つとすると，

$$2|R_1〔A〕| \leq |R_1|$$

従って　$C_{sj} \leq C_j$ となり，半結合の通信コストは，結合の場合よりも安価になる。

リレーションの属性Xについての θ-join に対する選択度〔HEVNA78a, SELIP79〕を，

$P_{RX\theta}$ と記すと，C_{sj}は，次の様になる。

$$C_{sj} = |R_1〔A〕| + P_{R_1A\theta} \cdot |R_2〔B〕| \quad \cdots\cdots (3)$$

最終的に生成されるR_1''のサイズは次の様になる。

$$|R_1''| = |R_1| \cdot P_{R_2B\theta} \quad \cdots\cdots (4)$$

6.2 単純問合せの通信処理アルゴリズム ― BSアルゴリズム

単純問合せとは，ただ1つの結合属性から成る問合せである。

$L_1, L_2, \cdots\cdots, L_m$ を，各々 $DM_1, DM_2, \cdots\cdots, DM_m$ 上のリレーションとし，属性aを結合属性とする。この時，問合せは，次の様に書かれる。

range　$(l_1, L_1 : S_1)(l_2, L_2 : S_2)\cdots\cdots(l_m, L_m : S_m)$;

retrieve into　$R(tl(l_1), \cdots\cdots, tl(l_m))$

where　$\bigwedge_{i=1, m-1} \bigwedge_{j=i+1, m} l_i.a\ \theta\ l_j.a$;　$\cdots\cdots$ (1)

以降，問題を簡単にするために等価結合（$\theta \in \{=\}$）についてのみ考えることにする。(1)式は，問合せグラフによって，図6.3の様に表わせる。図からも解かれるように，結合式の推移性によって，問合せグラフは完全グラフとなる。

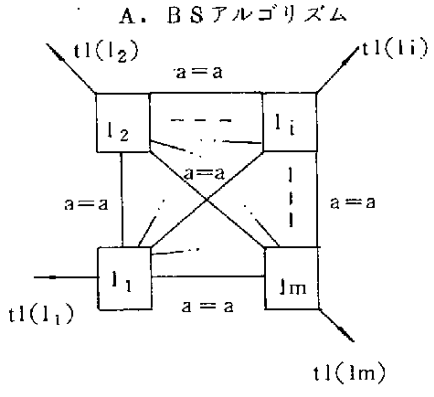


図6.3　(1)の問合せグラフ

通信ネットワークが物理的に1：N通信機能を備えていることから，あるリレーション l_i を転送すると，全DMで l_i を受信出来る。このことは，全DMで，l_i との結合，即ち $l_i.a = l_j.a$ ($j = 1, \cdots\cdots, i-1, i+1, \cdots, m$) を，DMの数に独立に行なえることを意味している。

又，このことは，リレーションの転送は，同時にただ1つだけ送られることも意味している。複数のリレーションを，同時に転送することは，転送しているDMで，転送と同時に他のリレーションの受信が行なわれることになり，結合によって転送されるリレーションのサイズを縮小出来ないことになる。

以上のことより，1：N通信を用いた結合処理は次の様な特徴をもつことが解かる。

1）同時，1つのリレーションが転送される。

2）このリレーションを全DMで受信できる。即ち，結合演算は並行して行なえる。

従って，問合せ処理のための全通信コストと応答時間とは，1：1通信と異なり等しくなる。

以下に，単純問合せに対する通信処理アルゴリズム（BSと呼ぶ）を示す。アルゴリズムBSは，全体LCS問合せに関連する全データモヂュール（DM）によって行なわれる。

アルゴリズムＢＳ

① ＤＭ$_i$について考える。

1. 結果ＤＭが送られてきた全体ＬＣＳ問合せをもとにして，4.6で述べた初期ローカル問合せ処理（ＤＭ内通信処理）を行なう。結果として生成されるリレーションL_iの結果性aについてのパフォーマンス情報（i.e. card(L_i〔a〕)）をＡＣＫにのせて全ＤＭに放送する。
L_iをaについてソートする。

2. 他の全てのＤＭから，ＡＣＫを待つ。

3. 全ＤＭからＡＣＫを受信したならば，card(Lk〔a〕)(k∈{1, 2, ……, m})が最少のL_kを見つける。

3.0 あるＤＭkからcard(Lk〔a〕)=0が送られてくれば， go to 3.6；

3.1 もし，card(L_i〔a〕)が最少ならば

$L_i' \leftarrow L_i$〔a〕； L_c'を全ＤＭに放送する。

3.1.1 L_iが結果属性を持つならば go to 2；

3.1.2 L_iが結果属性を持たないならば go to 3.5；

3.2 card(L_i〔a〕)が最少でないならば，他のＤＭ$_j$からcard(L_j〔a〕)が最少であるL_j'（$=L_j$〔a〕）の到着を待つ。

3.2.1 L_j'を受信したならば

$L_i \leftarrow L_i$〔a=a〕L_j'；

3.2.2 L_iが結果属性を持てば，（tl(L_i)≠∅） go to 3.4；

3.2.3 L_iが結果属性を持たなければ，（tl(L_i)=∅）

3.2.3.1 $|L_i$〔a〕$| < |L_j'|$ならば go to 3.4；

3.2.3.2 $|L_i$〔a〕$| = |L_j'|$ならば go to 3.5；

3.3 全L_k〔a〕が同一のカーディナリティを持つならば，〔全ての結果処理を終了した時〕
（i.e. $(\forall k)(L_k$〔a〕$=C)$）

3.3.1 L_iが結果属性を持つならば（tl(L_i)≠∅）

3.3.1.1 ＤＭ$_i$が結果ＤＭでなければ L_i〔tl(L_i)〕を結果ＤＭに転送する。
go to 3.5；

3.3.1.2 ＤＭ$_i$が結果ＤＭであれば，結果属性を持つL_k〔tl(L_k)〕の到着を待つ。
受信したならば $L_i \leftarrow L_i$〔tl(L_i)〕L_k〔tl(L_k)〕*)
全L_kを受信したならば L_iをユーザに出力して terminate；

3.3.2 L_iが結果属性を持たなければ（tl(L_i)=∅） go to 3.5；

3.4 〔ＡＣＫの放送〕
card(L_i〔a〕)を求めて，ＡＣＫにセットし，全ＤＭに放送する。
go to 2；

3.5 〔L_iのdelete〕

---

*) Ａ，Ｂを２つのリレーションとすると，ＡＢはＡの第ｉ組とＢの第ｉ組とをconcatenateすることである。

L_iをdeleteする。　　terminate ;

3.6 〔joinの失敗〕

3.6.1 DM_iが結果DMならば，ユーザに答がないことを知らせる。

go to 3.5 ;

3.6.2 DM_iが結果DMでなければ　　go to 3.5 ;

ここで，card(R)は,リレーションRのカーディナリティを表わす。

L_{j1} L_{j2},……L_{jp} のなかで，$L_{i_{k+1}}$以降の転送リレーションとして選択される可能性のあるリレーションは，

$L'_{j_k}〔a〕 \subset L_{i_k}〔a〕$ なる L_{j_k}だけである。これを

$$\beta i_k = \{ L_{k1}, L_{k2}, \cdots\cdots, L_{kpk} \} \quad \cdots\cdots (6)$$

となる。この時 $P_k \leq P$ である。

以上のことを用いると，アルゴリズムBSは次の様に書ける

```
        β0 ← { L1, L2, ……, Lm } ;
for k ← 1   step 1   while βk ≠ ∅
 do
  begin
        Lik ← L st.  L ∈ βk-1 ∧
                    (∀L' ∈ Bk-1)(|L〔a〕| ≦ |L'〔a〕|) ;
        (∀L ∈ βk-1  st. L ≠ Lik)(L ← L〔a = a〕Lik) ;        …(7)
        βk ← βk-1 − { Lik } − { L | L ∈ βk-1 ∧
                                |L〔a〕| = |Lik〔a〕| ∧
                                t1(L) = ∅ }
  end
 doend
```

B．通信コスト

ここで，$L_{i_k} \in \Phi_{BS}$の結合属性aの選択度〔HEVNA78a〕を$P_{L_{ik}a}$とすると,結合(4)によって生成されたリレーション L_{j_k}のカーディナリティは，(8)のようになる。

$$card(L_{j_k}) \cdot P_{L_{ik}a} \quad (k = 1, 2, \cdots\cdots p) \quad \cdots\cdots (8)$$

これを用いて，転送シーケンスΦ_{BS}における通信コスト$C\Phi_{BS}$を求めてみると，(9)式の様になる。

$$C\Phi_{BS} = |L_{i_1}〔a〕| + P_{L_1}a \cdot |L_{i_2}〔a〕| + P_{L_1}a \cdot P_{L_2}a \cdot |L_{i_3}〔a〕| + \cdots\cdots + P_{L_1}a \cdot P_{L_2}a \cdot \cdots\cdots P_{L_{n-1}}a \cdot |L_{i_n}〔a〕|$$

$$= \sum_{k=1}^{n} |L_{i_k}〔a〕| \cdot \prod_{l=0}^{k-1} P_{L_l}a \quad \cdots (9)$$

ここで $P_{L_0}a \triangleq 1$

(7)で示す様に，アルゴリズムＢＳでは，あるリレーション$L_{i_k} \in \Phi_{BS}$の次の$L_{j_{k+1}}$の候補となり得るリレーションの集合β_kには，$L\langle a\rangle \cap L_{i_k}\langle a\rangle \neq 0 \wedge L_{i_k}\langle a\rangle - L\langle a\rangle \neq \emptyset$たるリレーション$L$がふくまれている。従って，$\Phi_{BS}$は，各$L_i\langle a\rangle$（$i=1, 2, \cdots\cdots, m$）の集合論的関係によって決まる。以下に，いくつかの場合について検討してみる。

1) $\forall k \in \{1, 2, \cdots\cdots, m\}$　$L_1\langle a\rangle \subseteq L_k\langle a\rangle$の時(best case)

この時，リレーション$L_i\langle a\rangle$の転送によって，全ＤＭに$L_1\langle a\rangle = L_2\langle a\rangle = \cdots\cdots = L_m\langle a\rangle$なるリレーション$L_1, \cdots\cdots, L_m$が生成される。即ち，ただ$L\langle a\rangle$の転送によって，必要な結果が得られることになる。従って

$$\Phi_{BS} = \{ L_i \} \qquad \cdots\cdots (10)$$

ここで　$|L_i\langle a\rangle| = \min_{k \in \{1, 2, \cdots\cdots, m\}} (|L_k\langle a\rangle|)$

通信コストは　$C\Phi_{BS} = |L_i\langle a\rangle|$　$\cdots\cdots (11)$

アルゴリズムＢＳは，結果として，転送されたリレーションのシーケンスを生成することになる。このシーケンスを　Φ_{BS}　とすると，次の様になる。

$$\Phi_{BS} \triangleq \{ L_{i1}, L_{i2}, \cdots\cdots, L_{im} \} \qquad \cdots\cdots (2)$$

ここで　$i_k \in \{1, 2, \cdots\cdots, m\}$　$(k = 1, 2, \cdots\cdots n)$

$i_k \neq i_{k'}$　$(k \neq k'$ で　$i_{k'} \in \{1, 2, \cdots\cdots, m\})$

$n \leq m$

(2)式は，すべてのｋ，ｋ′$\in \{1, 2, \cdots\cdots, n\}$に対して，$k > k'$　なる　L_{i_k}と$L_{i_{k'}}$がある時，$L_{i_{k'}}$は　L_{i_k}の転送後に転送されることを示している。これを

$$L_{i_k} > L_{i_{k'}} \qquad \underline{iff} \qquad k > k' \qquad \cdots\cdots (3)$$

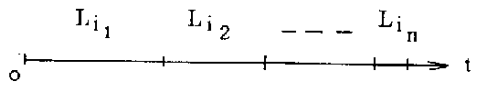


図6.4　ネットワーク上に転送されるリレーションのシーケンス$\Phi = L_{i1} \cdots, L_{in}$

となる。これは，ネットワーク上に現われるリレーションは，時間軸に対して，図6.4の様に示されることである。

あるリレーション$L_{i_k} \in \Phi_{BS}$が送られるリレーションを$L_{j_1}, L_{j_2}, \cdots, L_{j_p}$とする。各　j_kでは，次の結合処理がなされる。

$$L_{j_h} \leftarrow L_{j_h}\langle a = a\rangle L'_{i_k} \qquad (h = 1, 2, \cdots\cdots, p) \qquad \cdots\cdots (4)$$

ここで　$L'_{i_k} = L_{i_k}\langle a\rangle$

この結合の結果　L'_{ik}とL_{i_h}には，次の様な関係がある。

即ち

$$L_{j_h}\langle a\rangle \leq L'_{i_k}\langle a\rangle \qquad \cdots\cdots (5)$$

ϕ_{BS} として，L_{i_k} の次にとられるリレーション $L_{i_{k+1}}$ は，L_{j1}，……，L_{jP} のなかで，$|L_{jh}(a)|$ (h=1，…，P) が最少のものである。全ての $h \in \{1, 2, \cdots, P\}$ に対して，$|L_{jh}(a)|$ が等しければ

$$L_{i_k}(a) = L_{j_1}(a) = L_{j_2}(a) = \cdots\cdots = L_{jP}(a)$$

となり，これが求める解となる。

2) $\forall i, i', i'', i''' \in \{1, 2, \cdots\cdots, m\}$ ($i \neq i' \neq i'' \neq i'''$)

$$\begin{aligned}&((L_i(a) \cap L_i'(a) \neq \phi \wedge L_i(a) - L_i'(a) \neq \phi) \wedge \\ &(L_i(a) \cap L_i'(a)) \wedge (L_i''(a) \cap L_i'''(a)) \neq \phi \quad \wedge \\ &(L_i(a) \cap L_i'(a)) - (L_i''(a) \cap L_i'''(a)) \neq \phi)\end{aligned}$$

(worst case)

この場合には，m－1 個のリレーションが通信される。

$$\phi_{BS} = \{L_{i_1}, L_{i_2}, \cdots\cdots, L_{i_{m-1}}\} \qquad \cdots\cdots\cdots\cdots\cdots\cdots \quad (12)$$

ここで　$i_k \in \{1, 2, \cdots\cdots m\}$ ($k = 1, 2, \cdots\cdots, m-1$)

$i_k \neq i_{k'}$ に対して　$L_{i_k} \neq L_{i_{k'}}$

通信コストは，次の様になる。

$$C\phi_{BS} = \sum_{k=1}^{m-1} |L_{i_k}(a)| \prod_{l=0}^{k-1} P_{L_{i_l}} a \qquad \cdots\cdots\cdots\cdots\cdots\cdots \quad (13)$$

ここで　$P_{L_{i_0}} a \fallingdotseq 1$

$$\begin{aligned}C\phi_{BS} = {} & |L_{i1}(a)| + P_{Li1} a \, |L_{i2}(a)| + P_{Li1} a \cdot P_{Li2} a \cdot \\ & |L_{i3}(a)| + \cdots\cdots + P_{Li1} \cdot \cdots \cdot P_{Li_{m-2}} \cdot |L_{i_{m-1}}(a)|\end{aligned}$$

6.3　ＢＳアルゴリズムの例

本節では，ＢＳアルゴリズムを例を用いて示す。例として，図6.4の様な問合せを考える。ノード上に書かれた数は，リレーションのサイズを表わし，カッコ内の数字は，結合属性の選択度を表わす。各リレーションは，結合属性 a だけから成るとする。以下にアルゴリズムＢＳを示す。

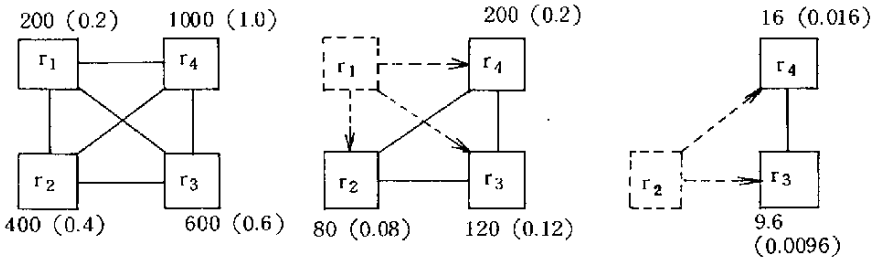


図 6.4　問合せ処理例

1. $|r_1|$ が最少なので，r_1 を放送する。よって $T_1^{r_1} = |r_1| = 200$

2. r_2，r_3，r_4 と r_1 との結合処理を行なう。結合処理後のリレーションのサイズと選択度は次の様になる。

$$|r_2| = 0.2 \times 400 = 80 \qquad P_2 = 0.2 \times 0.4 = 0.08$$
$$|r_3| = 0.2 \times 600 = 120 \qquad P_3 = 0.2 \times 0.6 = 0.12$$
$$|r_4| = 0.2 \times 1{,}000 = 200 \qquad P_4 = 0.2 \times 1.0 = 0.2$$

3. r_2 のサイズが最少なので，r_2 の放送を行なう。よって

$$T_2^{r_2} = |r_2| = 80$$

4. この結果は，次の様になる。

$$|r_3| = 0.08 \times 120 = 9.6 \qquad P_3 = 0.08 \times 0.12 = 0.0096$$
$$|r_4| = 0.08 \times 200 = 16 \qquad P_4 = 0.08 \times 0.2 = 0.016$$

5. 従って $T_3^{r_3} = |r_3| = 9.6$

最後に $T_4^{r_4} = |r_4| = r_4 = 16 \times 0.0096 = 0.1536$

よって $\phi_{BS} = \{ r_1, r_2, r_3, r_4 \}$

$$C\phi_{BS} = T_1^{r_1} + T_2^{r_2} + T_3^{r_3} + T_4^{r_3}$$

$$= 200 + 80 + 9.6 + 0.1536 = 289.75$$

Y_{a0} のアルゴリズムによると〔HEVNA78b〕

応答時間　　t = 400

全処理時間　　t = 700　　となる。

6.4 評　　価

図6.5から，図6.8に Y_{a0}〔HEVNA78a，b〕のアルゴリズムPによる全処理時間と応答時間と，選択度が正しいとの仮定のもととのアルゴリズムBSとの比較を示す。図6.9から図6.12には，アルゴリズムBS（6.2の(3)式)と，Y_{a0} のアルゴリズムとの時間の比を示している。

これらの図から解かる様に，応答時間については，ほとんど同様な結果を得ている。全処理時間は2～4倍 Y_{a0} の方が大きくなっている。これは，彼のアルゴリズムPが,転送のパラレリズムを高めるように，働らくからと考えられる。

結合をとるリレーション数が増加するにつれてアルゴリズムBSの結果と，彼の応答時間とは差がなくなってくる。我々のアルゴリズムBSでは，転送されるサイトDM数は，後のステージ程少なくなっていき，1：1通信に近くなっていくためと思われる。

全体として，1：N通信ネットワークでは，全処理時間を，大幅に縮小できるが応答時間に対しては，1：1通信と同様となることが解かる（通信容量を同一とした時）。

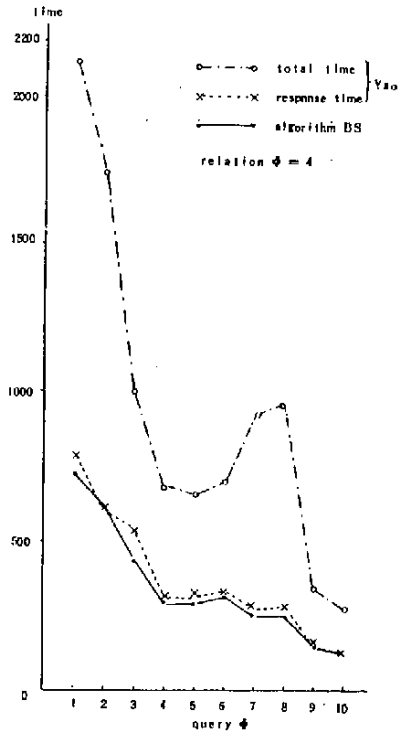


図 6. 5

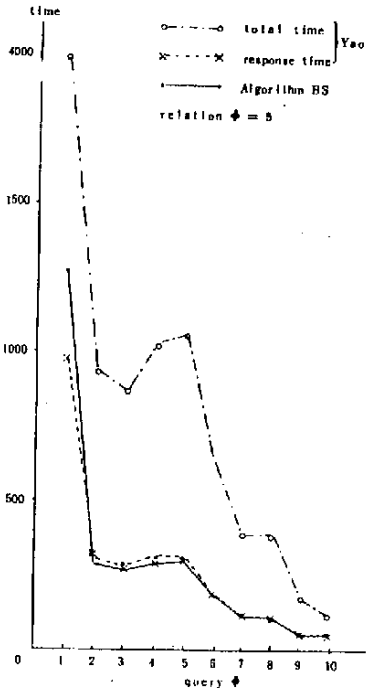


図 6. 6

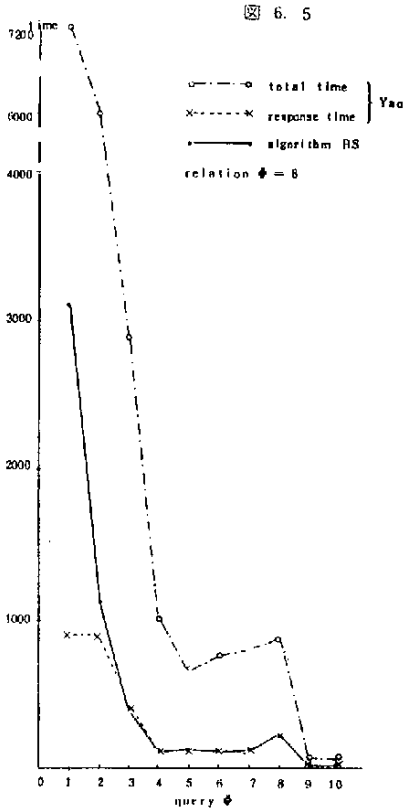


図 6. 7

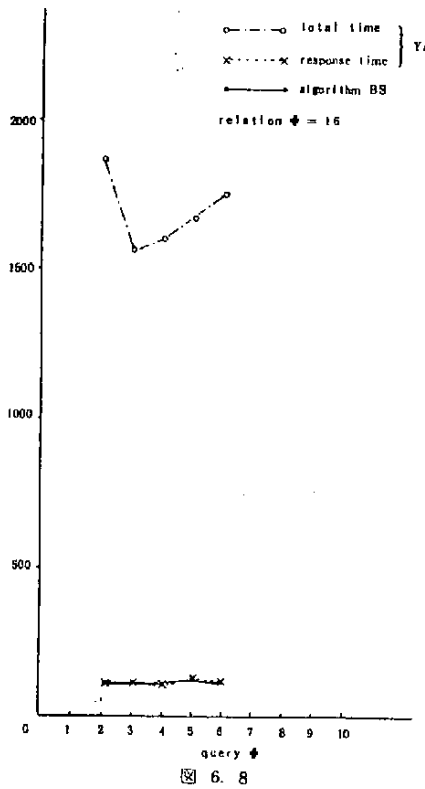


図 6. 8

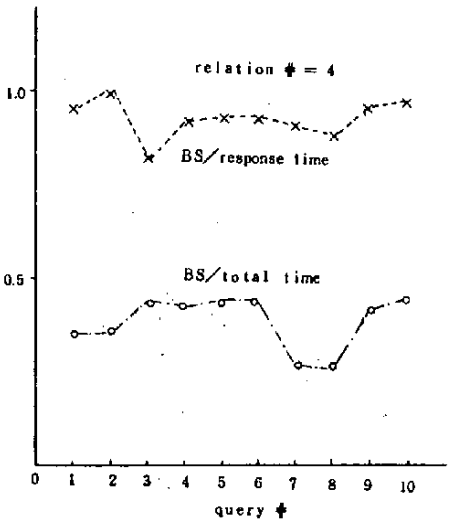


図 6. 9

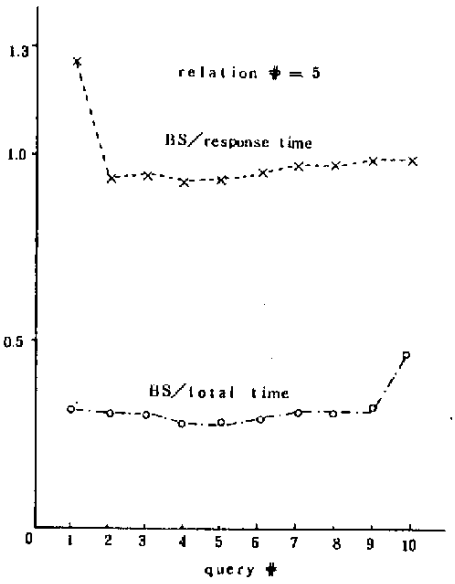


図 6. 1 0

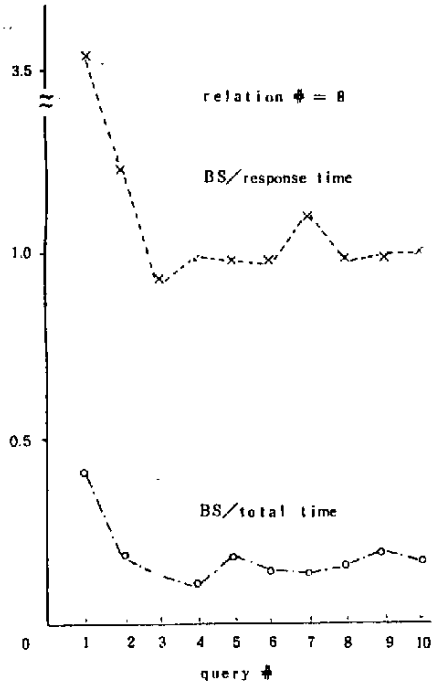


図 6. 1 1

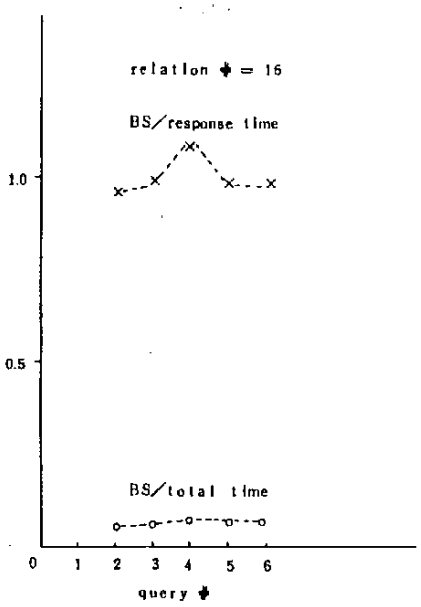


図 6. 1 2

しかし，共有媒体ネットワークでは，複数リレーションをパラレルに転送できない。このため，Y_{a0}のアルゴリズムを用いると，応答時間は全処理時間となってしまう。このため，我々のＢＳアルゴリズムは，ローカル共有媒体ネットワーク上で有効である。

6.5 一般問合せ処理アルゴリズム

次に，複数の結合属性を持つ場合を考えてみる。ここで以下の様な記法を用いる。

R_i ≡ リレーション （ $j=1, 2, \cdots\cdots, n$ ）

A_i ≡ リレーションR_iの属性集合 （ $i=1, 2, \cdots\cdots, m$ ）

TL_i ≡ 問合せのR_iに関する結果属性の集合

$$i.e.\ TL_i \subseteq A_i$$

$\mathcal{Q}$ ≡ 問合せの結合属性の集合

a_j ≡ 結合属性 （ $j=1, 2, \cdots\cdots, m$ ）

即ち $\mathcal{Q} = \{ a_1, a_2, \cdots\cdots, a_n \}$

ここで，R_iのa_jについての射影を次の様に定義する。

$$R_i[a_j] = \begin{cases} R_i[a_j] & \underline{if}\ \ a_j \in A_i \\ \phi & \underline{otherwise} \end{cases}$$

一般問合せの処理は，ＢＳアルゴリズムを単純に拡張することによって行なえる。一般問合せの処理アルゴリズムをＢＧアルゴリズムと呼ぶ。ＢＧアルゴリズムの概要は，次の様になる。

ＢＧアルゴリズム

0. ローカル問合せ処理（ＤＭ内通信処理）を，各ＤＭで行なう。
1. 各ＤＭは，生成された結果リレーションの各結合属性についての射影のサイズ（ $\forall_{ij} \mid R_i[a_j] \mid$ ）をもとめ，その情報を，他の全てのＤＭに転送する。
2. 全ＤＭからのＡＣＫを受信したならば，全ての $\mid R_i[a_j] \mid$ （ $i=1, 2, \cdots\cdots, m$, $j=1, 2, \cdots\cdots, n$ ）のなかで最少のものをもとめる。これをR_iとa_jとする。
3. 自分のＤＭがR_iとa_jとを持つならば

   $R_i' \leftarrow R_i[a_j]$ ； R_i'を全ＤＭに転送する。

   3.1 結果属性も，他の結合属性も持たなければ

   （ $i.e.\ A_i - \{a_j\} \cap \mathcal{Q} = \phi \ \wedge\ TL_j = \phi$ ），R_iを消去。

   3.2 全てのＤＭからのＡＣＫを持つ。 go to 2 ；
4. 自分のＤＭがR_i, a_jを持たないならば，$R_i[a_j]$の到着を持つ。R_i'（ $= R_i[a_j]$ ）を受信したならば，このＤＭ内の全てのリレーションR_kに対して，

   $R_k \leftarrow R_k[a_j = a_j]\ R_i'$ ；

   4.1 R_kが結果属性も，他の結合属性も持たない時

   かつ $\mid R_k[a_j] \mid = \mid R_i' \mid$ の時，R_kを delete。

   4.2 go to 1 ；

このアルゴリズムは図6.13の様に記せる。

```
β ← β0 ← { Ri〔al〕 | i∈{ 1, 2, ……, m } ∧
                    l∈{ 1, 2, ……, n } ∧ al ∈Ω } ;
for     t := 1    step 1    while βt ≠ φ
 do
  begin
       L ← Rk〔al〕   st. Rk〔al〕∈Bt-1 ∧
                  | Rk〔al〕| = min | L' |   ;
                                  ∀L' ∈Bt-1
       βt ← βt-1 − { Rk〔al〕} ;
  for    ∀L'∈βt
   begin
        L' ← Rk〔al〕〔al = al〕L'   ;
    if  ( Ω∩( AL' − { al } ) = φ ∧ TLL' = φ ∧
                  | L'〔a〕| = | Rk〔al〕| ),
      then
       begin    βt ← βt − { L'〔al〕} ;
                β ← β − { L'〔al〕}
       end
   end
end
```

図6.13　アルゴリズムBG

6.6　まとめと今後の課題

本章では，問合せを処理するためのDM間の通信処理について論じた。我々のアルゴリズムは，選択度等の統計的情報に基づいた静的な通信処理スケジュールの決定ではなく，問合せ処理に関するDMの完全な分散制御のもとでの動的なスケジュール決定に基づいている。この理由としては，次の点がある。

1）選択度といった統計情報の正しさに対する疑問

第7章で述べる様に，選択度は，値が均一分散している仮定に基づいているために，この仮定が成り立たないデータに対して，その正しさが問題となる。

選択度といったパフォーマンス情報が，スキーマ情報と比して動的であることから，各DMでこれらの情報を管理するオーバヘッドが問題となる。

2) ローカル共有媒体ネットワーク

このネットワークが，1：N通信を物理的に提供していることから，各DMが他のDMの状態を同時に知り得る。

これらによって，各ステップで，最も通信コストの小さいリレーションを転送できる。この各ステップでの最適性は，処理全体の最適性を意味していない。しかし，本質的にリレーション処理の中間結果(e·g· 結合，制限，射影)を正確に見積もることは，リレーションの値の分散状態に依存していて，不可能であることから，我々は，全体最適性を達成することは出来ないと考えている。各ステップで，最適を簡単なアルゴリズムによって求めることは，feasible な解となると考えている。

DM間通信処理の今後の課題としては，以下の点がある。

1) 更新処理

GCS層に対する更新要求の意味を，LCS層のモデル要素によって表わさねばならない。この問題は，ビュー更新問題と呼ばれている困難な問題である。我々はこの問題を解決するために，付記Iに示すデータ抽象化技法が有効であると考えている。即ち，ビューに対して許されるオペレーションを，あらかじめ定義しておき，このオペレーションを通してのみビューのアクセスを許すものである。

更新の第2の問題は，いわゆる同時実行制御である。同時実行制御のなかのコミットメント制御に対して，ローカルネットワークの持つ1：N通信機能を用いることは有効である。即ち，異なったDMにある複数の関連するデータに対して，更新要求を物理的に同時に転送できるからである。1：1通信ネットワークにおいて，パフォーマンス上問題であった同時実行制御を，1：Nネットワークでは，より容易に，かつ有効に行なえる。

2) 問合せとして限量子(∀，∃)，aggregate関数の処理

現在，我々の問合せは，aggregate関数(即ち，∀)を含まない。これらの処理のための手法の開発が今後必要となる。

7. 問合せ変換 － ＤＭ　通信処理

ローカル初期問合せ処理によって生成されたローカル問合せ〔5.6節の(1式〕は，１つのデータモデュール（ＤＭ）内で独立に処理できる。即ち，ＤＭ内で通信処理される。ローカル問合せは，そのＤＭ内のＬＣＳリレーションのみを参照するリレーショナル問合せ（ＬＣＳ問合せと呼ぶ）である。ＤＭが，リレーショナルＤＢＳではなく，異種ＤＢＳである場合には，ＬＣＳ問合せをそのＤＭで実行可能なアクセス表現に変換し，実行させる必要がある。これを問合せ変換〔TAKIM 80 a〕と呼ぶ。問合せ変換としては，特にQUEL LCS問合せから,CODASYL DBTGプログラムへの変換と，その実行について論じる。この問合せ変換システムは，次の様な位置づけをもっている。

1) ＤＤＢＳにおいて，CODASYL DBTG DBSとなる。（既存大型ＤＢＳ）に対する共通インタフェースとなる。

2) CODASYL DBTG DBS のエンドユーザ向けリレーショナルインタフェースとなる。

本章で論じる問合せ変換システム（ＱＴＰ－Ｖ２）は，〔TAKIM80 a〕に対して次の点で機能拡張を試みている。

1) 実際のCODASYL DBTG DBS（ＡＩＭ on Facom M－１６０）での実働

2) キー属性間の不等価結合の処理

3) 非キー属性間の結合処理

4) 出力問題（フォーム出力）

5) ＬＣＳに対する階層的なビュー定義と，ビューの問合せ処理

7.1では，問合せ変換を考えるうえでの基本仮定を論じる，7.2では，問合せ変換の概要を論じる。7.3では構造変換を，7.4ではアクセス生成を，7.5ではＤＭＬの生成について述べる。7.6では結合処理を，7.7ではビューの処理を述べ，7.8では本システムの評価について論じる。

7.1 基　本　仮　定

本節では，問合せ変換を考えるうえでの基本仮定について述べる。

A．ＬＣＳ問合せ

ローカル概念スキーマ（ＬＣＳ）に対する問合せを，ＬＣＳ問合せと呼ぶ。ＬＣＳ問合せは，ＬＣＳ問合せに対して，次の様な仮定を設ける。

1) 検索のみを考える。

2) aggregate関数をもたない。（aggregate － free）

3) ＬＣＳ問合せは積正規化されている。

4) 問合せの中の主キーについての結合は，等価（＝）又は非等価（≠）結合だけが許される。

5) 非主キー属性についての結合は任意のものを許す。ただし，結合されるリレーションは，主キーによる結合によって連結でなければならない。

MAX，MIN，AVER，COUNTといったaggregate関数は，値の集合に対する統計的な処

理を行なう。aggregate関数のなかで，group by 機能を用いないものに対しては，処理は比較的容易である。即ち，aggregate関数を独立した問合せとして処理してその結果に対して，対応するaggregate処理を行なえばよい。その後，初めの問合せ内のaggregate関数を，処理結果として得られた値で置き換える。例えば，(1)の様な比較述語が問合せ条件式内に現われたとする。

…… ＜a－exp＞＜cop＞＜g－func＞ …… …… (1)

ここで ＜g－func＞::＝＜g－func-name＞(＜a－exp＞$_t$ where＜qual＞)

…… (2)

この時(2)に対応して，(3)の様な問合せが生成される。

retrieve into ＜temp-rel＞(＜temp.att＞＝＜a－exp＞$_t$)

where ＜qual＞ ； …… (3)

(3)の結果＜temp-rel＞に対して，＜g－func-name＞に対応する統計処理を行なう。この結果をCとすると，(1)は，(4)の様になる。Cは，ある数値である。

…… ＜a－exp＞＜cop＞ c …… …… (4)

ＬＣＳ問合せ(q)は，5.4節の(1)式の様に，積正規化されているとする。

range (x_1, X_1)(x_2, X_2) …… (x_m X_m) ；

q： retrieve into R(r_1 = a exp$_1$, r_2 = a exp$_2$, ……, r_k = a exp$_k$)

where qual ；

qual ≜ c_1 〔$\wedge c_2 \wedge \cdots\cdots \wedge c_n$〕 (1)

c_i ≜ c-pred$_{i1}$〔$\vee$c-pred$_{i2}\vee\cdots\cdots\vee$c-pred$_{il_i}$〕

ここで $\forall j \in \{1, 2, \cdots\cdots, l_i\}$ c－pred$_{ij}$ = rp$_{ij}$(x) | jp$_{ij}$(x, y)

ここで x, y ∈ { x_1, x_2, ……, x_m }

c_i 内(i＝1，2，……，n)の全ての比較述記 c－pred$_{ij}$ (j＝1，2，……，l_i)は，同一の変数を参照していなければならない。

ＬＣＳ問合せの主キーについての結合としては，＝と≠だけを許す。我々は，結合には次の２種あると考えている。

ｉ）構造を表わす結合

ii）値の関連を表わす結合

ＥＳリレーションの各組は，実体(entity or object)を表わし，ＲＳリレーションの組は実体間の関係性を表わしている。主キーについての結合とは，ＥＳリレーションとＲＳリレーションとの間でなされ，なお実体と他の実体との関連(ＲＳの表わす関係性をもつかもたないか)を示している。従がって，結合は＝と≠とについてしかあり得ない。この概念は，オブジェクトモデル〔TANAY80〕〔付記Ｉ〕概念に近いものである。

一方，ii)は，任意のリレーション間の属性の値の関連であるので，任意の結合を許す。しかし，結合を行なう組は，ｉ）の構造結合によって結合されている必要がある。

B. CODASYL DBTG モデル

変換目標のCODASYL DBTGモデルに対しては，次の仮定を設ける。

1) 〔CODAS73〕に準拠する。
実際のインプリメントの対象としては，FACOM の AIM*)を用いる。

2) CODASYL DBS に対する応答時間は，アクセスされるオカーランス数に比例するとする。アクセスコストとしては，アクセスされるオカーランス数をmetricとする。この仮定の正しさについては，評価において論じる。

7.2 問合せ変換システムの概要

問合せ変換システムは，次のことを行なうシステムである。

1) QUEL問合せを入力として受けとる。

2) QUEL問合せを，COBOL DMLプログラムに変換する。

3) 変換されたプログラムを，対応するCODASYL DBTG データベースシステム上で実行させる。

4) 結果を，ユーザに出力する。

QUEL問合せを，COBOL DMLプログラムに変換するためには，まず，この2つの差を明らかにせねばならない。QUEL問合せと，COBOL DML プログラムとの相違点は，次の2点である。

1) 互いが参照するデータモデルが異なっている。QUEL問合せは，リレーショナルモデルに基づき，COBOL DMLプログラムは，CODASYL DBTGモデルに基づいている。

2) QUEL問合せは，非手続き的であるが，COBOL DMLプログラムは，手続き的である。

従って，問合せ変換(query translation)とは，これらの2つの相異を，解決することであると言える。

第1の問題は，QUEL問合せの意味をCODASYL DBTGモデルによって，非手続的に表わすことである。このためには，問合せの参照するリレーショナルモデル要素(i・e・リレーション，属性)と，問合せが表わしている要素間の関係性(i・e・リレーション間の結合(join))とを，CODASYL DBTG モデルの対応する要素と関係性とによって置き換えることが必要である。このプロセスは，DBTGスキーマから，リレーショナルスキーマへの変換，即ち同種化の逆過程である。リレーショナルモデル要素と，CODASYL DBTG モデル要素との対応関係は，異種性情報(HI)内に保持されている。この情報を用いて，リレーショナル問合せの意味を，CODASYL DBTG モデルによって表わすことができる。この過程を，構造変換(structure transform)

---

*) AIMは，〔CODAS73〕には準じていない部分がかなりある。特に，更新に対するインテグリティ機構(e. g. メンバシップクラス) がそれである。しかし，今回は検索のみを考えているので，AIMでも十分と考えている。来年度以降〔CODAS73〕に準拠しているADBS(NEC)に対するインプリメントを予定している。

と呼ぶ。又，構造変換によって生成されたＤＢＴＧモデルによる問合せは，ＤＢＴＧ問合せグラフ（ＤＱＧ）と呼ばれるグラフによって表わされる。ＤＱＧは，ＱＵＥＬ問合せの意味を，ＤＢＴＧモデルによって非手続的に表わしたものである。

ＤＱＧは，非手続的な問合せ表現であるので，次に，CODASYL DBTG データベースシステム上で実行可能な手続きを生成せねばならない。この手続生成をアクセスパス生成（access path generation）と呼ぶ。アクセスパス生成では，目標データベースシステム（ＤＢＳ）上で，最適なアクセス手続きを生成することが必要になる。この最適化の目標関数としては，次のものがある。

1) 中間結果を最少化する。

2) 応答時間を最少化する。

この目標関数を達成する様に生成されたアクセス手続きは，アクセス木（access tree）と呼ばれる木構造によって表わされる。

アクセス木から，自動的にCOBOL DMLプログラムが生成される。これをＤＭＬ生成（ＤＭＬ generation）と呼ぶ。目標DBTG DBS の DML 情報は，異種性情報（ＨＩ）内のＤＭＬ情報（ＤＭＬＩ）内に格納されている。このＤＭＬＩを，各DBTG DBS ごとに生成することによって，容易に，CODASYL DBTG タイプのＤＢＳに適応できる。

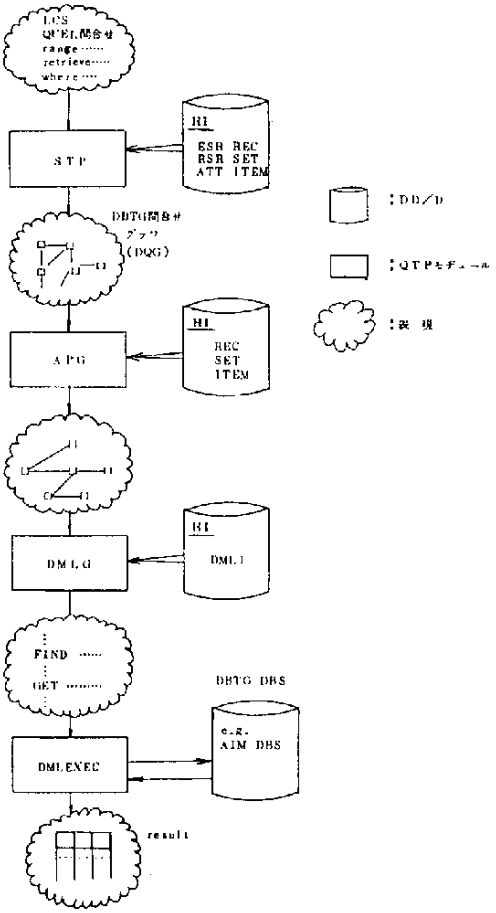


図7.1　ＱＴＰ の 概 要

7.3 構造変換（Structure Transformation）

構造変換プロセッサ（ＳＴＰ）は，ＱＵＥＬ問合せから，異種性情報（ＨＩ）を用いて，問合せの意味のＤＢＴＧモデルによる非手続き的表現への変換を行なう。異種性情報（ＨＩ）は，QUELの参照するローカル概念スキーマ（ＬＣＳ）リレーションと，ＤＢＴＧモデルのローカル内部スキーマ（ＬＩＳ）要素との対応情報を格納している。ＱＵＥＬ問合せは，リレーショナル問合せグラフ（ＲＱＧ）と呼ばれるグラフによって表わされる。一方，ＳＴＰはこのＲＱＧから，ＤＢＴＧ問合せグラフ（ＤＱＧ）と呼ばれるグラフを生成する。ＤＱＧは，ＱＵＥＬ問合せの意味を，DBTGモデル要素によって非手続的に表現したものである。

本章では，ＲＱＧからＤＱＧへの変換について論じる。7.3.1では，リレーショナル問合せグラフ（ＤＱＧ）を論じる。7.3.2では，ＤＢＴＧ問合せグラフ（ＤＱＧ）を述べ，7.3.3ではＲＱＧからＤＱＧへの変換について論じる。

7.3.1 リレーショナル問合せグラフ

ＬＣＳ問合せ(q)の結合には，次の２つの意味がある。

1）構造の表現

2）値の関連

1)は，ＥＳリレーションの表わすオブジェクトの関連を，ＲＳリレーションを通して視ることである。これは，リレーションのキー属性による結合として表わされる。この結合を主結合（primary join）と呼ぶ。7.1の仮定で述べた様に主結合は，オブジェクト間の関連を表わすものであるから，等価（＝）と非等価（≠）結合のみが許される。変数 x_i と x_j との間の主結合を表わす述語を $p_{jp}(x_i, x_j)$ とする。このなかで等価主結合，非等価主結合を表わす述語を，各々 $pejp(x_i, x_j)$，$pnjp(x_i, x_j)$ と記すことにする。

一方，値の関連は，任意の属性（ただし，定義域が等しくなければならない）間の結合によって表わされる。これは，全ての比較演算子（＜，≦，＝，≧，＞，≠）に対して可能である。これを非主結合（non-primary join）と呼ぶ。x_i と x_j との間の非結合述語を $njp(x_i, x_j)$ と表わす。ただし，7.1で述べた様に，x_i と x_j とは，主結合によって連結でなければならない。

この時，x_i と x_j との間の結合論理式 $jf(x_i, x_j)$ は，次の形式をしていなければならない。

$$
\begin{aligned}
&jf(x_i, x_j) :: = p_{jp}(x_i, x_j) \wedge njf(x_i, x_j) \\
&njf(x_i\ x_j) :: = NJF \\
&\quad NJF :: = njf(x_i, x_j) \mid NJF \wedge NJF \mid NJF \vee NJF \\
&\quad p_{jp}(x_i, x_j) :: = pejp(x_i, x_j) \mid pnjp(x_i, x_j) \\
&\quad pejp(x_i, x_j) :: = x_i.a = x_j.a \\
&\quad pnjp(x_i, x_j) :: = x_i.a \neq x_j.a \qquad \cdots (1) \\
&\qquad\qquad \text{ここで } a \text{ は } x_i \text{ と } x_j \text{ の主キー属性}
\end{aligned}
$$

$njp(x_i, x_j)$ :: = $aexp_i$ θ $aexp_j$

ここで，$aexp_k$ (k = i, j)は，x_kの任意の属性上の算術式

ここで $\theta \in \{ <, \leq, =, \geq, >, \neq \}$

aは range(x_i)の属性

bは range(x_j)の属性

aとbとは，同一の定義域に属している

主結合と非主結合とに対応して，ＲＱＧ(5.4)には，次の２種の結合リンクが存在することになる。

JLNK ≜ (PJLNK, NJLNK)

PJLNK ≜ (PEJLNK, PNJLNK)

PEJLNK ≜ $\{ pejl_{ij} \mid i, j = 1, 2, \cdots\cdots, m \ (i \neq j) \}$

$pejl_{ij}$ ≜ (x_i, x_j $pejp(x_i, x_j)$)

≜ node(x_i) と node(x_j) との間に存在し，等価主結合述語$pejp(x_i, x_j)$ を表わしている。

PNJLNK ≜ $\{ pejl_{ij} \mid i, j = 1, 2, \cdots\cdots, m \ (i \neq j) \}$

$pejl_{ij}$ ≜ (x_i, x_j, $pnjp(x_i, x_j)$)

≜ node(x_i) と node(x_j) との間に存在し，非等価主結合述語$pnjp(x_i\ x_j)$ を表わしている。

NJLNK ≜ $\{ njl_{ij} \mid i, j = 1, 2, \cdots\cdots, m \ (i \neq j) \}$

njl_{ij} ≜ (x_i, x_j, $njf(x_i, x_j)$)

node(x_i) と node(x_j) との間に存在し，非主結合論理式$njf(x_i, x_j)$ を表わす。

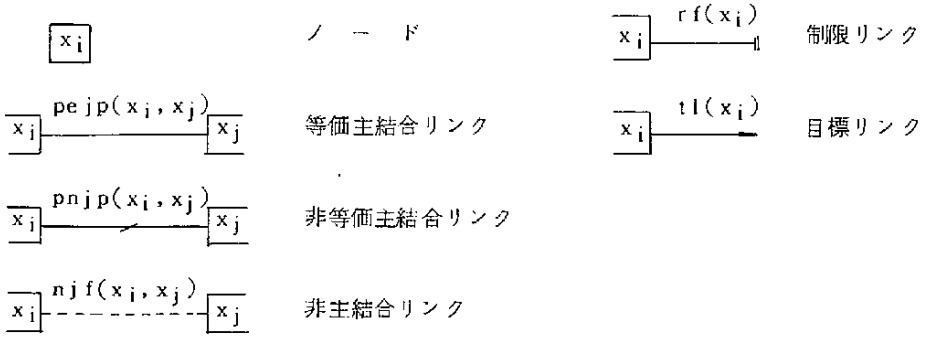


図7.2 拡張 ＲＱＧ(ＥＲＱＧ)

これらのリンクに対応したグラフは，図7.2の記号を用いて書かせる。このグラフを，拡張リレーショナル問合せグラフ（extended relational query gaph or ERQG）と呼ぶ。

例として，PRDBS〔付記Ⅱ〕に対する次の様なＬＣＳ問合せを考えてみよう。

例，ＤＢＳ又はＤＤＢＳを研究しているプロジェクトの開始年に入社した部員のなかで，このプロジェクトには属していず，かつＤＢＳと又はＤＤＢＳとに関するレポートを書いていない部員の名前と，彼の書いた論文名をもとめよ。

これは，ＱＵＥＬによって，図7.3の様に書ける。

```
range   ( e , EMPLOYEE)  ( p , PROJECT)( r , REPORTR) ;
range   ( pe, PROJ-EMP)  ( ps, PROJ-SUBJ)  ;
range   ( er, EMP-REP)   ( rs, REP-SUBJ)   ;
retrieve into   R( name=e.ename,  r.title)
where
      e.eno    ≠  ep.eno          and  e.eno = er.eno    and
      p.pno    =  ep.pno          and  p.pno = ps.pno    and
      r.rno    ≠  er.rno          and  r.rno = rs.rno    and
   rs.subjectn =  ps.subjectn      and
    p.syear    =  e .hire-date    and
  (ps.subjectn = " DBS"            or   ps.subjectn = " DDBS");
```

図7.3　ＱＵＥＬ問合せ

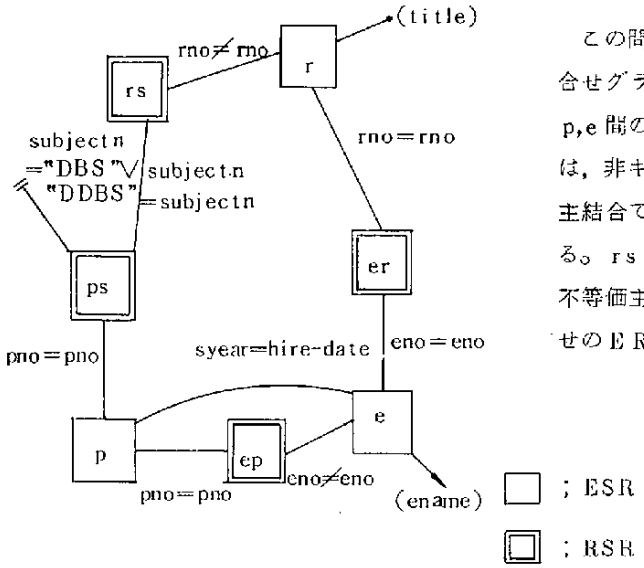


図7.4　ＲＱＧ

この問合せに対するリレーショナル問合せグラフ（ＲＱＧ）を図7.4に示す。p,e間の結合p.syear=e.hire-dateは，非キー属性に関していないので，非主結合である。他の結合は，主結合である。rsとr，及びeとep間の結合は，不等価主結合である。従って，この問合せのＥＲＱＧは図7.5の様になる。

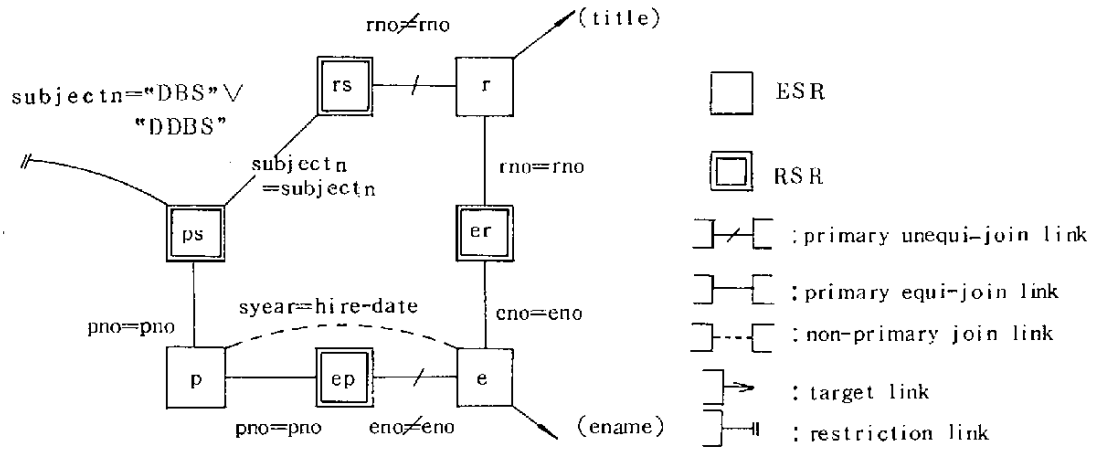


図 7.5　拡張 RQG（ERQG）

7.3.2　DBTG問合せグラフ（DQG）

DBTG問合せグラフ（DQG）は，LCS問合せの意味を，CODASYL DBTG モデル要素によって非手続的に表わしたグラフである。DQGノードは，レコード型を表わし，DQG間のリンクは，セット型（SLNK）又は，不等価主結合に対応した非セットリンク（NLNK）を表わしている。DQGは(1)の様に定義される。

DQG ≙ (DN, DLNK)
DN ≙ $\{ d_i \mid i = 1, 2, \cdots\cdots, l \}$
≙ DQGノード(d_i)を表わすDQGノード
ここで，　l＝DQGノード数
d_i ≙ レコード型　$rec(d_i)$ を表わすDQGノード
ここで
$rec : DN \rightarrow REC$
$\forall d \in DN \quad \exists r \in REC \quad rec(d) = r$
rは，DQG内でdによって表わされるレコード型。　…(1)
DLNK ≙ (SLNK, NLNK, RLNK, TLNK, NJLNK)
SLNK ≙ $\{ sl_{ij} \mid i, j = 1, 2, \cdots\cdots, l, \quad i \neq j \}$
sl_{ij} ≙ (d_i, d_j) ≙ d_i から d_j への有向アーク
ここで　set : SLNK → SET
$\forall sl \in SLNK \quad \exists s \in SET \quad set(sl) = s$
sは，DQG内でslによって表わされるセット型。

この時　$rec(d_i) = set(sl_{ij})$ の親レコード型

$rec(d_j) = set(sl_{ij})$ の子レコード型

$NLNK \triangleq \{ nl_{ij} \mid i, j = 1, 2, \cdots\cdots, l, \quad i \neq j \}$

$nl_{ij} \triangleq (d_i\ d_j)$ ≙ d_i から d_j への有向アーク

ここで

$rec(d_i) = rec(d_j)$

d_i は正ノード，　d_j は反ノード。

$RLNK \triangleq \{ rl_i \mid i = 1, 2, \cdots\cdots, l \}$ …(1)

$rl_i \triangleq (rf(d_i))$

≙ ノード d_i に関する制限論理式 $rf(d_i)$ を表わす。

ここで　$d_i \neq$ 反ノード

$TLNK \triangleq \{ tl_i \mid i = 1, 2, \cdots\cdots, l \}$

$tl_i \triangleq (tl(d_i))$

≙ ノード d_i に関する目標属性リスト

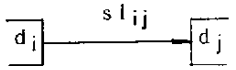
ここで　$d_i \neq$ 反ノード

$NJLNK \triangleq \{ njl_{ij} \mid i, j = 1, 2, \cdots\cdots, l, \quad i \neq j \}$

$njl_{ij} \triangleq (nif(d_i, d_j))$

≙ ノード d_i と d_j 間の非主結合論理式

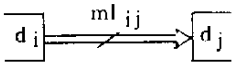
DQGは，図7.6の記号によって，図示できる。



: set-link(sl_{ij})

$rec(d_i)$ = an owner record-type of set(sl_{ij})

$rec(d_j)$ = a member record-type of set(sl_{ij})

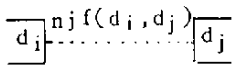


: non-set-link(nl_{ij})

r = $rec(d_i)$ = $rec(d_j)$

d_i = a pro-node of r

d_j = an anti-node of r



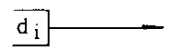
: non-primary join link (njl_{ij})

: record-type

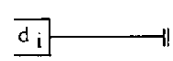
: link record-type

: pro-node

: anti-node



: target link



: restriction link

図7.6　DBTG　Query Graph

7.3.3 構造変換 RQG → DQG

拡張リレーショナル問合せグラフ（ＥＲＱＧ）は，ＬＣＳ問合せの意味を，グラフ表現したものである。ＥＲＱＧは，次の特徴をもっている。

1) ＬＣＳリレーションを参照し，

2) リレーション間の構造的関係づけを主結合リンクで示し，

3) リレーション間の値の関連を非主結合リンクで行なっている。

リレーショナル・スキーマ（ＬＣＳ）とCODASYL DBTG スキーマ（ＬＩＳ）との対応は，同種化においてなされている。構造変換は，同種化の逆過程としてある。同種化で生成された異種性情報（ＨＩ）を用いて，ＥＲＱＧ内のリレーション要素を，CODASYL DBTG モデル要素に置き換えねばならない。

一般に，ＥＳリレーションは，１対１にＤＢＴＧモデルのレコード型に対応づけられる。又，ＲＳリレーションのうちのＬリレーションは，リンクコード型に１対１対応づけを行なえる。問合せが参照するＤＢＴＧ構造（アクセスパス）は，ＬＣＳ問合せ内の主結合によって表わされる。従って，主結合は，対応するセット型によって表わされる。

A．隠れ構造

ＥＲＱＧを，ＤＱＧに変換する以前に，ＬＣＳ問合せの構造を調べてみよう。ＬＣＳ問合せ内の主結合は，ＬＣＳリレーションのキー属性に対してなされる。あるキー属性ａは，ＥＳリレーションと，これと他のＥＳリレーションとを関連づけるＲＳリレーション内に現われる。従って，キー属性についての主結合としては，図7.7の2種が考えられる。図の b) は，意味的には，c) の様に解釈できる。即ち，ある事象（オブジェクト）ａについての関係性ｒとｓとを示している。ユーザの発するＬＣＳ問合せは，図の b) の様に，ＥＳリレーションを省略した形式で書ける。我々はこの様なＥＳリレーションを隠れ構造（ＨＳ）と呼ぶ。即ち，隠れ構造とは，ＥＲＱＧ内の主結合リンクがＲＳリレーションに対応するノード間に存在している時，ＲＳリレーションの主結合属性を持つＥＳリレーションである。

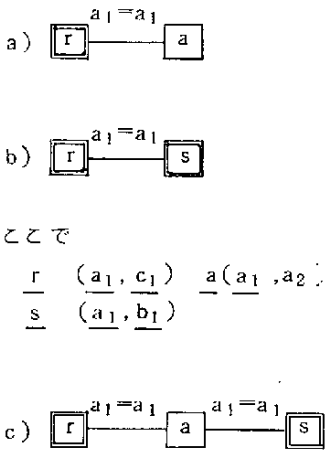


図7.7 主結合のパターン

ここで，ある主結合リンク pjl_{ij} を考える。主結合属性をａとする。ＥＲＱＧ上で，主結合属性ａを持つ主結合リンクで，リンクの推移関係を通して互いに連結な主結合リンクの集合を PJLa とする。

$$PJLa \triangleq \{ pjl_{ij} \mid pjl_{ij} = r_i.a\ \theta\ r_j.a \ \wedge\ a \in pkey(r_i) \ \wedge$$

$a \in pkey(r_j)$ }

ここでPJLa内の全ての主結合リンク（$pjl_{ij} \in PJLa$）は、連結である。

ここで $\forall v_i \in VAR_q$ $pkey(v_i) = \{ a \mid a$は，v_iのrange LCSリレーションの主キー $\}$

この時，隠れ構造をみつけるアルゴリズムを（HSFアルゴリズム），以下に示す。

1) aを主キーとするLCSリレーションをAとする。これは異種性情報（HI）のATTリレーション〔付記2〕を用いて得れる。

2) PJL_a 内の全ノードn_kをサーチして，$range(n_k) = A$ なるノードをさがす。

3) もしみつかれば（$n_k.a \ \theta \ r_j.a$），PJL_a からこれを消去する。
みつからなければ，4）へ。
次に，$n_k.a \ \theta \ r_l.a$ たるリンクをみつけ，PJL_a から消去する。

4) PJL_a 内のリンクを縦型にサーチする。
マークされていない各リンク $r_i.a \ \theta \ r_j.a$ に対して，$r_i.a \ \theta \ A.a$ と $r_j.a \ \theta \ A.a$ の結合リンクを生成し，PJL_a から $r_i.a \ \theta \ r_j.a$ を消去する。

5) $PJL_a = \phi$ となるまで4)をくりかえす。$PJL_a = \phi$ ならば，次の主キー属性aについて，1)からくりかえす。

このアルゴリズムは，等価主結合（i.e. $\theta \in \{ = \}$）に対して成り立つ。図7.8は，HSFによる隠れ構造の明確化の例を示している。しかし，$\theta \in \{ \neq \}$ に対しては，隠れ構造の発見は，図7.9に示す様な，問合せの意味のあいまいさをもたらすために出来ない。このため，我々は，非等価主結合に対しては，隠れ構造を許さないことにする。

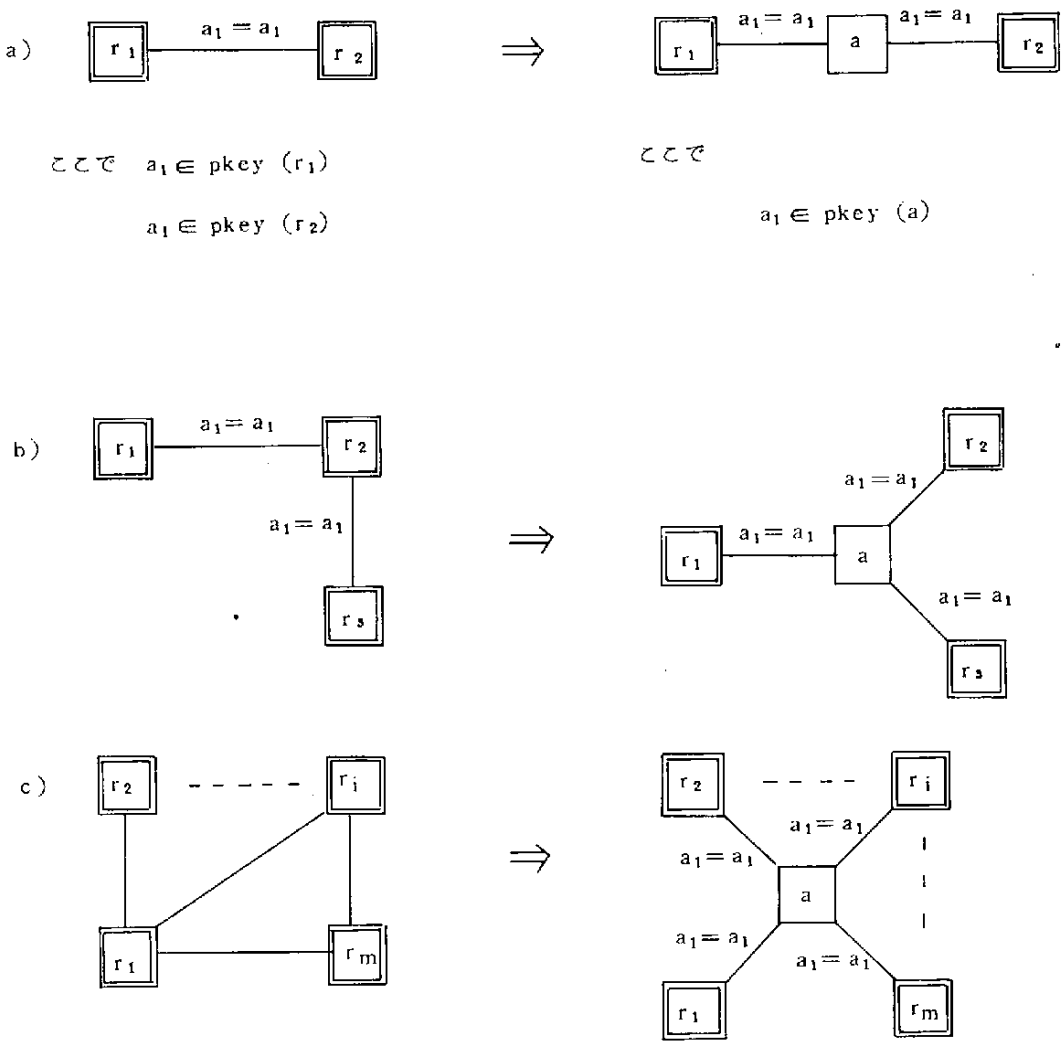


図 7.8　等価主結合の隠れ構造の例

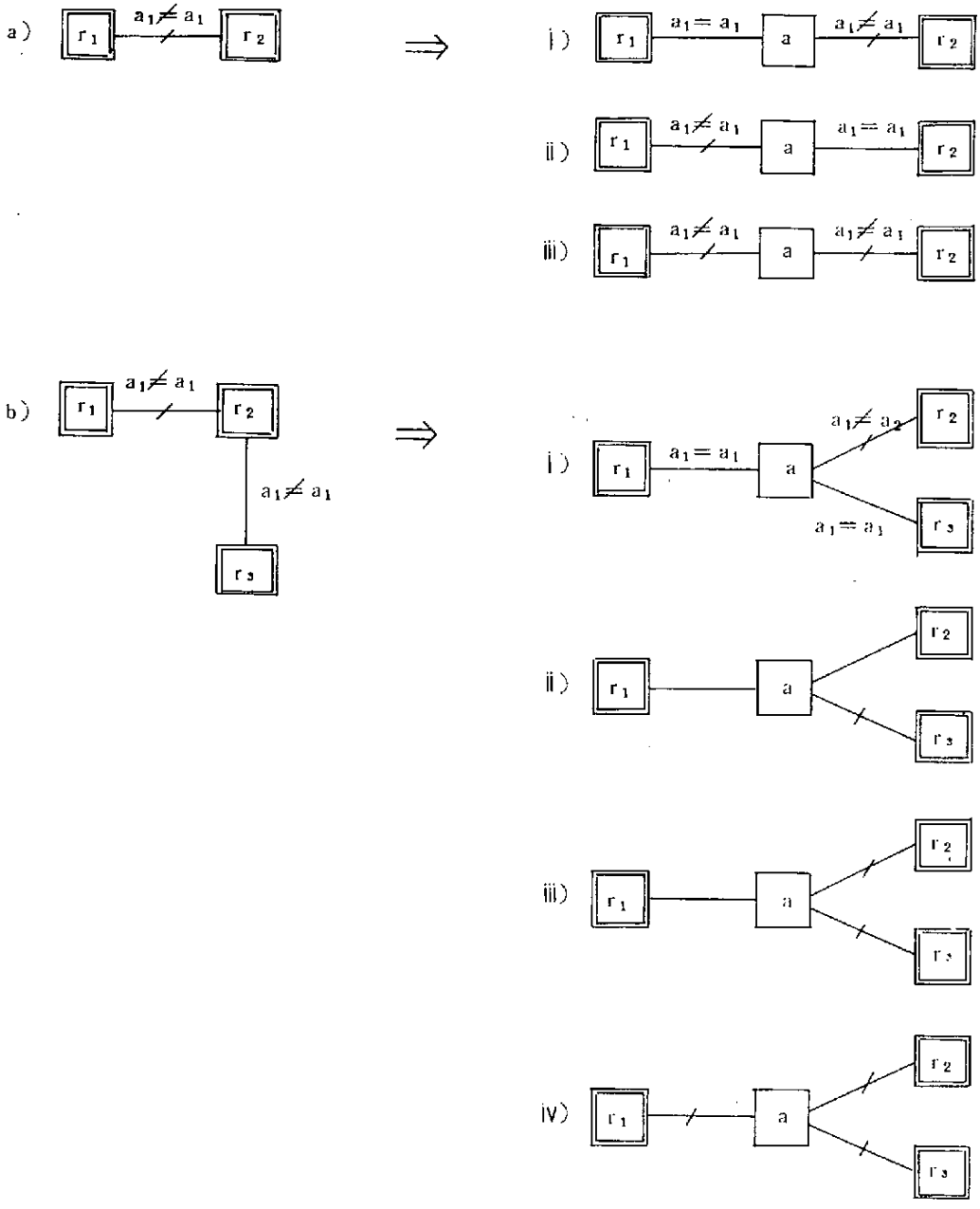


図 7.9　非等価主結合隠れ構造のあいまいさ

B．構造変換

隠れ構造が明らかにされたＥＲＱＧからCODASYL DBTG モデルによる問合せの非手続き的表現を生成する。

これは，次の様になる。

1) ＥＳリレーションを表わすＥＲＱＧノード(x_i) を，対応するレコード型を表わすＤＱＧノードｒに変える。

ここで次の x_i から対応するＤＱＧノードを生成する node procedure dnode(x_i) を次の様に定義する。

```
DQG node procedure   dnode(x_i) ;
ERQG node    x_i     ;
begin DQG node   d ;
    create  d  st. rec(d) = H_R^-1 ( range(x_i) ) ;
    tl(d) ←   σ_H  tl(x_i)   ;
    rf(d) ←   σ_H  rf(x_i)   ;
    dnode ←   d
end   of  dnode  ;
```

ここで　H_R ： REC ↔ ESR

（ＲＥＣはＬＩＳのレコード型の集合，ESRはLCS ESリレーションの集合）

H_I ： ITEM ↔ ATT

（ＩＴＥＭは，レコード型の項目集合，ＡＴＴはＥＳリレーションの属性集合）

σ_H は，ＬＣＳ属性ａを対応するデータ項目に置き換える置換オペレータ

$\sigma_H \triangleq \{ H_I^{-1}(a) / a \}$

2) $pjl_{ij} - x_j - pjl_{jk}$ について，次のことを行なう〔図7.10，図7.11〕

ここで　range(x_j) ＝ ＲＳリレーション

d_i = dnode(x_i)

d_k = dnode(x_k)　とする。

```
if     range(x_j) = RS relation
  then
    if   pjl_ij = pejl_ij ∧ pjl_jk = pejl_jk
      then
                          create-set (d_i, d_k, x_j)
  else  if   pjl_ij = pejl_ij ∧ pjl_jk = pnjl_jk
    then
```

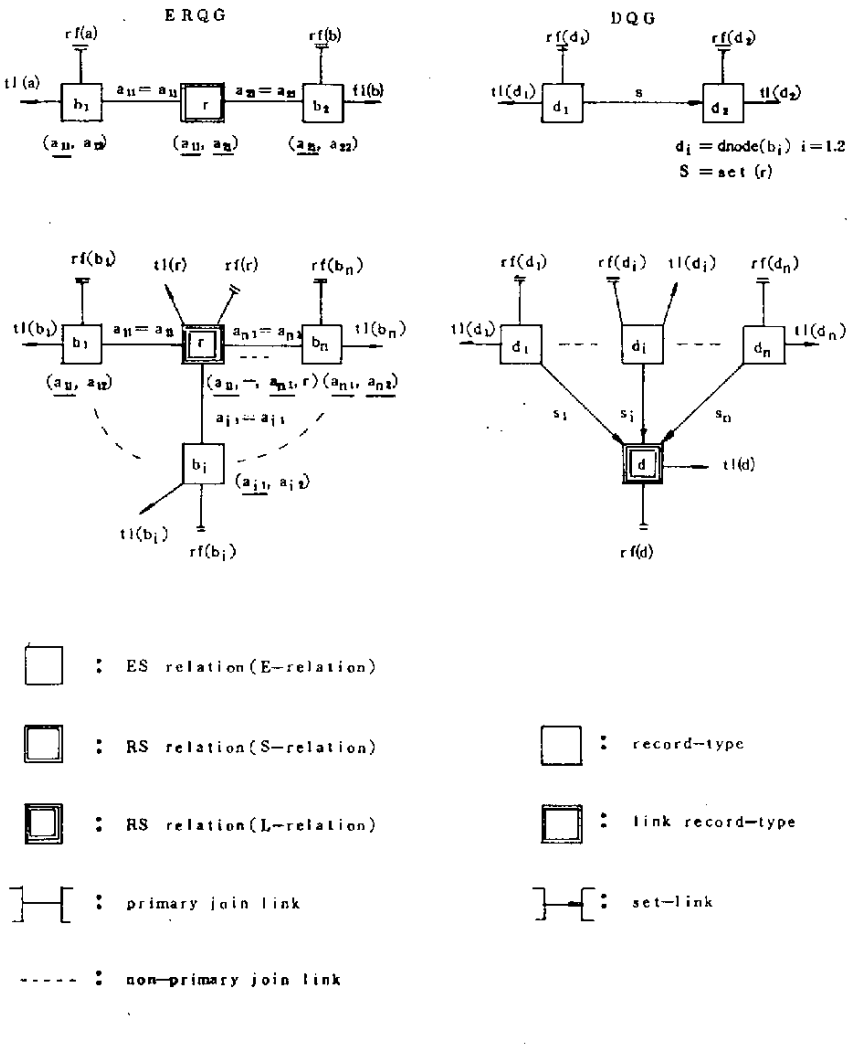


図7.10　主結合リンクの変換

```
    begin   dk' ← anti-node of dk
            create   nlkk'; create-set(dj, dk', xj)
    end
else if   pjlij = pnjljk ∧ pjljk = pejlij
  then
    begin   di' ← anti-node of di ;
            create   niii'; create-set(di', dk, xj)
```

```
    end
else if   pjl_ij = pnjl_jk ∧ pjl_jk = pnjl_jk
  then
    begin   d_i'← anti-node of d_i ;   d_k' ← anti-node of d_k ;
            create   nl_ii' ;   create   nl_kk' ;
                                  create-set(d_i', d_k', x_j)
    end
  else error
else error ;
procedure    create-set ( a, b, r ) ;
  DQG node      a, b  ;
  RQG node       r    ;
begin
  if     range(r) = S-relation
    then     〔S = set-type〕
      if     rec(a) =owner of set(r)
        then
                    create   sl_ab
        else        create   sl_ba
  else     〔r = link record-type〕
    if    dnode(r) is already created (r' = dnode(r))
      then
        begin     create  sl_ar' ;   create  sl_br'    end
      else
        begin     r' ←  dnode(r)  ;
                  create  sl_ar' ;   create  sl_br'
        end
end  of    create-set  ;
```

3) 非主結合リンク njl_{ij} に対しては

$$njl(d_i, d_j) \leftarrow \sigma_H \cdot njl(x_i, x_j)$$

図7.12と7.13には，各々図7.5のERQGとDQGを示す。

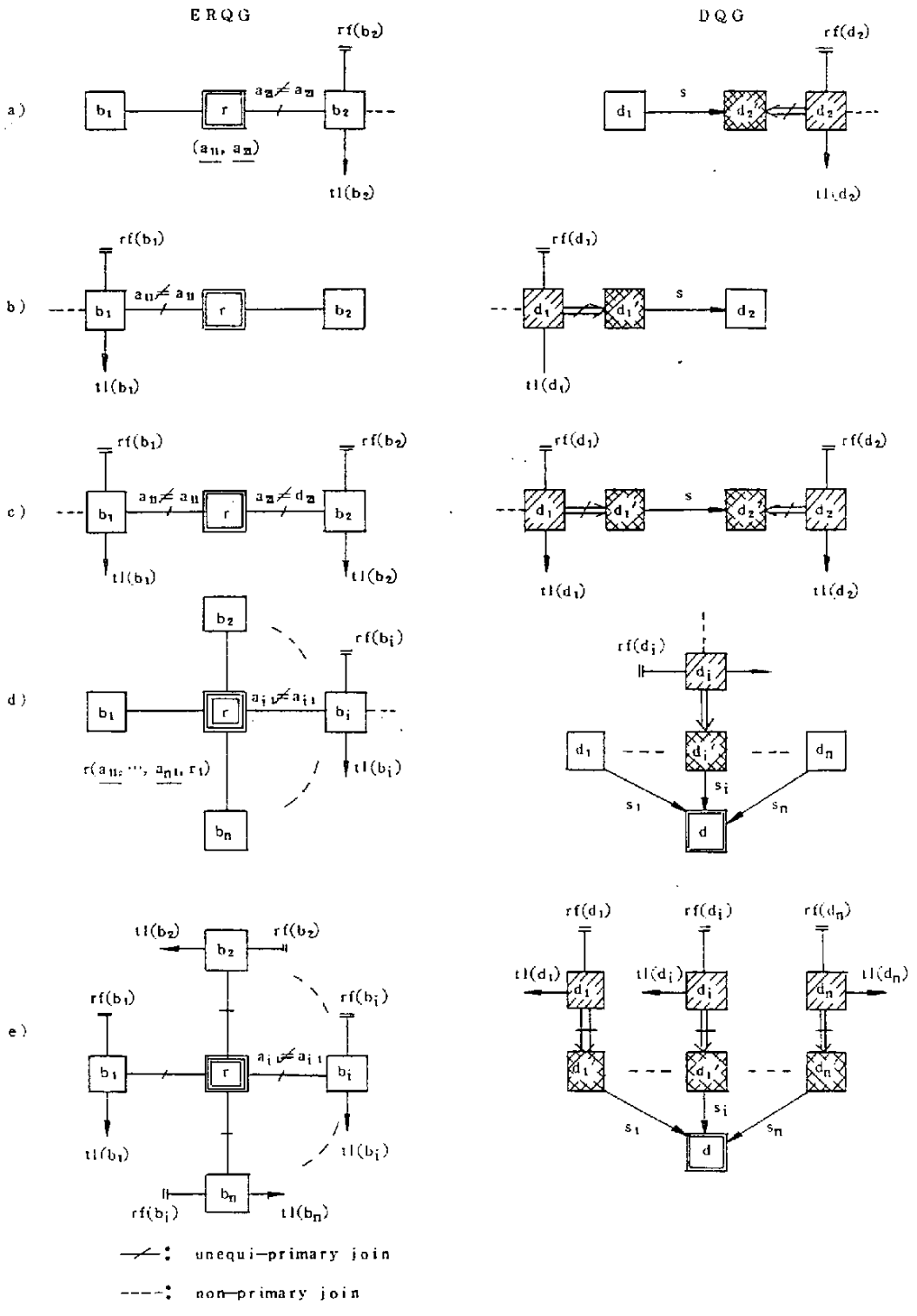


図 7.11　非等価主結合リンクの変換

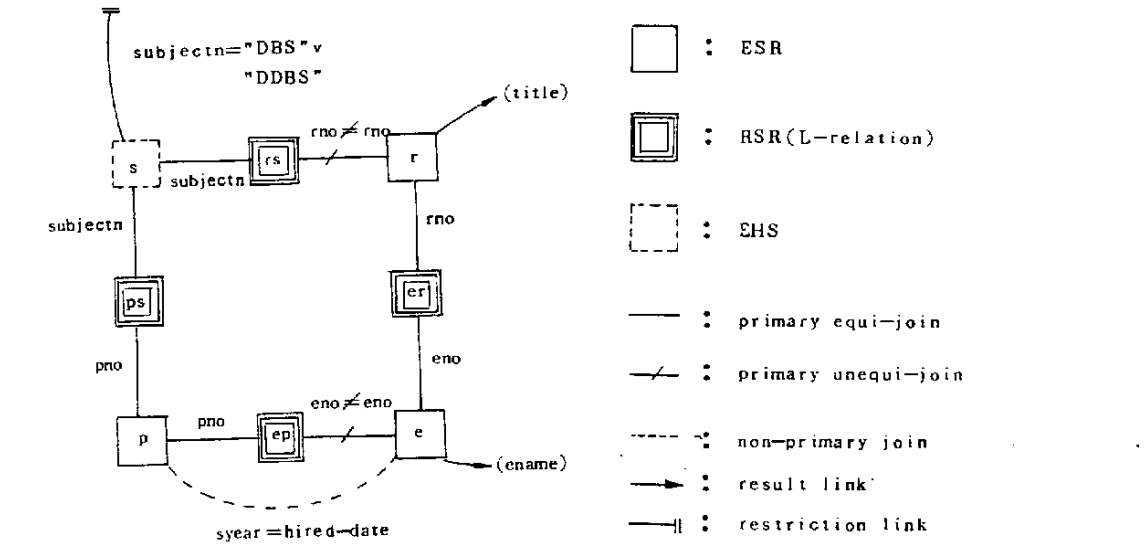


図 7.12　図 7.5 の ERQG

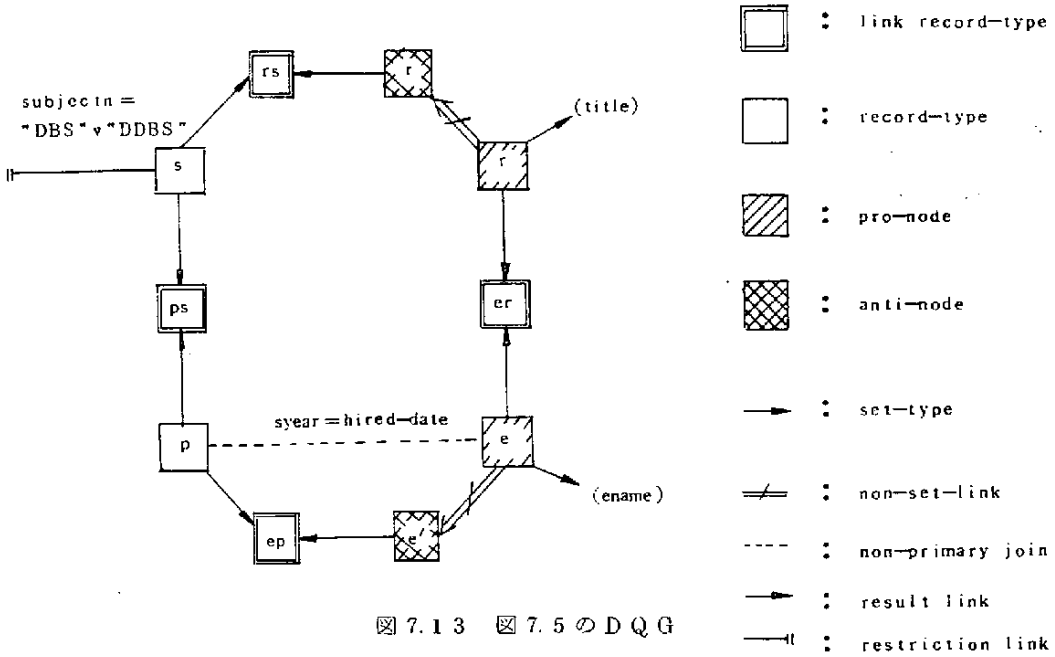


図 7.13　図 7.5 の DQG

7.4　アクセスパス生成（ＡＰＧ）

ＤＢＴＧ問合せグラフ（ＤＱＧ）は，ＬＣＳ問合せの意味を，ＤＢＴＧモデル要素によって非手続的に表わしたものである。従って，次の問題は，ＬＣＳ問合せと，ＤＢＴＧ　ＤＭＬプログラムとの第２の相違点である手続きの生成である。アクセスパス生成（ＡＰＧ）は，ＤＱＧから，最適なアクセスパスの生成を行なうモヂュールである。

7.4.1 目 標 関 数

アクセスパス生成の目標としては，次の２つがある〔TAKIM80a〕。

1) 中間結果を最少にする。

2) 応答時間を最少にする。

変換目標のCOBOL DMLは，閉包（closure）の性質を持っていない。即ち，DMLによって算出された結果に対しては，再びDMLによってアクセスすることができず，親言語によって行なわれなければならない。中間結果の量の最少化は，格納オーバヘッドを減じるうえで必要であるが，生成されるDMLプログラムの簡潔性の点から，中間結果の数の最少化が必要である。複数の中間結果は，各々，別々のファイルとして管理され，これらの結果間の集合演算機能をCODASYL DMLとは独立にインプリメントしなければならない。生成されたCOBOL DMLプログラムは，CODASYL DBTG DML言語のナヴィゲーショナルな特性を最大に生かしていることが必要である。中間結果数を増大させることは，DBTGモデル言語のナヴィゲーショナルな特徴とは反するものとなってしまう。

第２の目標は，ユーザにとって重要である。我々は，応答時間はアクセスされたオカーランスの数に比例すると仮定している。実際の応答時間は，アクセスされるページ数に比例してくるが，このページアクセスは，DBSの内部スキーマレベルの問題であると考えている。この仮定の正しさについては，我々のシステムの評価において検討する。

7.4.2 アクセスコスト

本節では，DBTG問合せグラフ（DQG）からアクセスパス生成のために必要なアクセスコストについて考える。7.1でも述べた様にアクセスコストは，アクセスされるレコードオカーランスの数である。

A. 基本定義

まず，アクセスコスト関数を算出するための基本事項の定義を行なう。

CODASYL DBTG モデルにおけるアクセスパスは，セット型と，これに関連したレコード型の対のシーケンスとして表わされる。アクセスは，このシーケンスをナヴィゲートすることによって行なわれる。アクセスシーケンスのk番目にレコード型 R_i を表わすDQGノードd_iがあり，k＋1番目に R_j を表わすd_jノードがある。更に，d_iとd_jは，セット型S_{ij}で結ばれているとする。

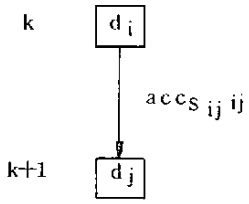


図7.14 アクセス関数

a．アクセス関数　$acc_{S_{ij}\,ij}$

ここで，ＤＱＧノードd_iから，セット型S_{ij}を介したノードd_jへのアクセス関数$acc_{S_{ij}\,ij}$を，次の様に定義する。

$$acc_{S_{ij}\,ij} : rec(d_i) \rightarrow rec(d_j) \quad (i.e.\ R_i \rightarrow R_j)$$

$$\forall r_i \in R_i \qquad acc_{S_{ij}\,ij}(r_i) = \{ r_j \mid r_j \in R_j \ \wedge \ ((r_i, r_j) \in S_{ij} \vee (r_j, r_i) \in S_{ij}) \} \qquad \cdots (1)$$

ここで，ＤＢＴＧモデルのセット型に対する制約から

$$\forall r_i, r_i' \in R_i \ (r_i \neq r_i') \qquad acc_{S_{ij}\,ij}(r_i) \cap acc_{S_{ij}\,ij}(r_i') = \phi \qquad \cdots (2)$$

$card(R_i) \triangleq R_i$ のカーディナリティ

$\triangleq R_i$ のレコードオカーランス数

この時，ノードd_iのオカーランスr_iから，アクセスパス$acc_{S_{ij}\,ij}$を介してアクセスされるノードd_i内のレコードオカーランス数は，次の様になる。

$$card\,(acc_{S_{ij}\,ij}(r_i \in R_i))$$

$$\triangleq card\,(\{ r_j \mid r_j \in R_j \ \wedge \ ((r_i, r_j) \in S_{ij} \vee (r_j, r_i) \in S_{ij}) \}) \qquad \cdots (3)$$

$\triangleq R_i$ とセットオカーランスによってリンクされているR_jのオカーランス数

b．結　合　度　　ct_{ij}

(1)と(3)式を用いて，アクセス関数$acc_{S_{ij}\,ij}$によるノードd_iに対するノードd_jの結合度（connectivity）ct_{ij}を定義する。ct_{ij}は，ノードd_iの１つのレコードオカーランスが，セット型S_{ij}を介して，平均何個のd_jオカーランスとリンクされているかを示している。

$$ct_{ij} \triangleq average\,(\forall r_i \in R_i \quad card\,(acc_{S_{ij}\,ij}(r_i)))$$

$$\triangleq \Big[\sum_{\forall r_i \in R_i} card\,(acc_{S_{ij}\,ij}(r_i)) \Big] / card(R_i) \qquad \cdots (4)$$

(4)式について，更に考えてみよう。

i) ノードd_iは，セットS_{ij}の親レコード型を表わし，d_jは子レコード型を表わす時〔図7.15〕。

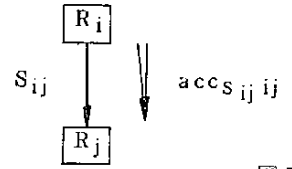


図7.15

$$\forall r_i \in R_i \qquad acc_{S_{ij}\,ij}(r_i) = \{ r_j \mid r_j \in R_j \ \wedge \ (r_i, r_j) \in S_{ij} \} \qquad \cdots (5)$$

セット型S_{ij}がTタイプ〔2.3〕の時，即ち，S_{ij}の子レコード型R_iオカーランスを持つ時，

$$R_j = \bigcup_{\forall r_i \in R_i} acc_{S_{ij}\,ij}\ (r_i) \qquad \cdots\cdots (6)$$

又，(2)式が$acc_{S_{ij}\,ij}$に対して成り立つことから結合度ct_{ij}は，次の様になる。

$$ct_{ij} \triangleq \left[\sum_{\forall r_i \in R_i} card\ (acc_{S_{ij}\,ij}(r_i)) \right] / card(R_i)$$

$$= card(R_j) / card(R_i) \qquad \cdots\cdots (7)$$

一方，S_{ij}がPタイプ時，S_{ij}の全ての子レコードオカーランスは親レコードオカーランスをもつとは限らない。従って，

$$R_i \supseteq \bigcup_{\forall r_i \in R_i} acc_{S_{ij}\,ij}\ (r_i) \qquad \cdots\cdots (8)$$

$$ct_{ij} = card(R_j') / card(R_i)$$

$$ここで \quad R_j' = \bigcup_{\forall r_i \in R_i} acc_{S_{ij}\,ij}\ (r_i) \qquad \cdots\cdots (9)$$

R_iのなかで親レコード型をもつオカーランス集合

ii) ノードd_iはS_{ij}の子レコード型で，d_jは親レコード型の時〔図2.16〕。

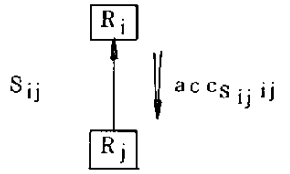


図 7.16

$$\forall r_i \in R_i \quad acc_{S_{ij}\,ij}\ (r_i) = \begin{cases} r_i \in R_i & if\ (r_j, r_i) \in S_{ij} \\ \phi & otherwise \end{cases} \qquad \cdots (10)$$

よって $0 \leq card(acc_{S_{ij}\,ij}(r_i)) \leq 1$

S_{ij}がTタイプの時，子レコードオカーランスは必らず親レコードオカーランスを持つので，

$$\forall r_i \in R_i \quad card\ (acc_{S_{ij}\,ij}(r_i)) = 1$$

従って，$card\left(\bigcup_{\forall r_i \in R_i} acc_{S_{ij}\,ij}(r_i)\right) = card(R_i)$

$$ct_{ij} = card(R_i) / card(R_i) = 1 \qquad \cdots\cdots (11)$$

S_{ij}がPタイプの時は，

$$\forall r_i \in R_i \quad 0 \leq card\ (acc_{S_{jj}\,ij}(r_i)) \leq 1$$

$$ct_{ij} = \left[\sum_{\forall r_i \in R_i} card\ (acc_{S_{ij}\,ij}(r_i)) \right] / card(R_i)$$

$$= card(R_j') / card(R_i) \qquad \cdots (12)$$

ここで，$R_j' \triangleq \bigcup_{\forall r_i \in R_i} acc_{S_{ij}\,ij}\ (r_i)$

$\triangleq$ R_iの中で，子レコード型を持つオカーランス集合

表7.1は，(7)，(9)，(10)，(12)をまとめたものである。

表 7.1　結合度 ct_{ij}

R_i	R_j	S_{ij} タイプ(α/β)*	結合度　ct_{ij}
owner	member	T(M/A, F/A)	$card(R_j) / card(R_i)$
		T(O/M)	$card(R_j') / card(R_i)$ R_j' ($\subseteq R_i$)は，R_jの中で親レコードオカーランスを持つオカーランス集合
member	owner	T(M/A, F/A)	1
		P(O/M)	$card(R_j') / card(R_i)$ $R_i' \subseteq R_j$は，子レコードオカーランスをもつ親オカーランスの集合

＊　α/β = membership class
　　α:M = MANDATORY　　β:A = AUTOMATIC
　　O = OPTIONAL　　M = MANUAL
　　F = FIXED

c．選択度　(selectivity)　st_i

アクセスシーケンス内のノードd_iを考えてみる。このノードの制限リンクrl_iは，制限論理式$rf(d_i)$を表わしている。$rf(d_i)$の選択度st_iは，$rf(d_i)$を満足するd_iオカーランス数が，d_i内にどの位存在するかを示している。

$$st_i \triangleq card(\{ r_i \mid r_i \in R_i \wedge rf_i(r_i) = true \}) / card(R_i) \quad \cdots\cdots (13)$$

ここで

$$rf_i(r_i) = \begin{cases} true \ \underline{if} & オカーランス\ r_i\ が制限論理式\ rf(d_i)\ を満足する。 \\ false \ \underline{otherwise} & \end{cases}$$

即ち，　$st_i \cdot card(R_i)$は，R_i内の条件$rf(d_i)$を満足するオカーランス数となる。

ここで，次の様な仮定を設ける〔HEVNA78b〕。

1）オカーランス内の各属性値は，均一に分散している(uniform distribution)。

2）属性間の値の依存関係はない。

この仮定のもとで，論理式$rf(d_i)$に対する選択度を統計的に次の様に定義する。

論理式　$d_i.a = v$　をpとする。

ここでaは$R_i(rec(d_i))$の属性で，vはある定数($v \in dom(a)$)とする。

$$st_i \triangleq \sum_{\forall v \in dom(a)} [card(\{ r_i \mid r_i \in R_i \wedge r_i[a]=v\}) / card(dom(a))] / card(R_i) \quad \cdots\cdots (14)$$

表 7.2　制限論理式と選択度*)

論　　理　　式	選　　択　　度
1) $d_i.a = v$	st_a
2) $d_i.a \neq v$	$1 - st_a$
3) $d_i.a > v \vee d_i.a < v$	$(1-st_a)/2$
4) $d_i.a \geq v \vee d_i.a \leq v$	$(1-st_a)/2 + st_a = (1+st_a)/2$
5) $d_i.a = v_1 \vee d_i.a = v_2 \vee \cdots\cdots \vee d_i.a = v_n$	$\sum_{i=1}^{n} st_a = n \cdot st_a$
6) $d_i.a_1 = v_1 \vee d_i.a_2 = v_2 \vee \cdots\cdots \vee d_i.a_n = v_n$	$1 - \prod_{i=1}^{n} (1 - st_{a_i})$
7) $rf_1 \wedge rf_2 \wedge \cdots \wedge rf_n$ ここで rf_iは，制限論理式	$\prod_{i=1}^{n} st_{rf_i}$
8) $rf_1 \vee rf_2 \vee \cdots \vee rf_n$	$1 - \prod_{i=1}^{n} (1 - st_{rf_i})$

(14)式を基本にした時の，論理に対する選択度を，表7.2にまとめる。3)と4)とは，論理的な根拠はない。$d_i.a > v$ を満足するオカーランス数は，$d_i.a \neq v$ よりは少ないことを表わしている。この選択度の妥当性は，今後，実験を通して明らかにしていきたい。

B．DBTGモデルのアクセスパス

DBTGモデルで許されるアクセスパスとしては，次のものがある。

1) CALC機構による直接アクセス

2) レルム内のシステムが与えた順序に基づいたシーケンシャルアクセス

3) セットオカーランス内の順序に基づいたシーケンシャルアクセス
（singular set-type）

CALC機構としては，主インデックスに対応するDNA　CALCと，2次インデックスに対応するDA　CALCとがある。

セットオカーランスの順序としては，ある属性値の昇順，降順の2つがある。又，順序内に，この属性値がユニーク（DNA）が，冗長を許すか（DA）がある。

*) diが第2番目以降の合法ノード（7.4.4）の時，st＝1/card (Ri)

ノードdに対する上記3種のアクセスパスを用いたアクセスコストacc(d)は，次の様になる。ここでst'_dは，ノードについてのCALC equi-restrictionの選択度である。

1) DNA CALC　　acc(d) = card(range(d))・st'_d
   DA CALC　　acc(d) = card(range(d))・st'_d
2)　　acc(d) = card(range(d))
3) singular set
   DNA　　acc(d) = card(range(d))／2

表7.3　primary access restriction formula (prf)

access path	access cost	prf
DNA CALC DA CALC	card(R)・st_d	$P_a(v) = d.a = v$ $P_a(v_1) \vee P_a(v_2) \vee \cdots\cdots \vee P_a(v_m)$
DNA next	card(R)／2	$P_a(v)$ $P_a(v_1) \vee P_a(v_2) \vee \cdots\cdots \vee P_a(v_m)$
ascending next prior	card(R)／2	$P'_a(v) = n.a \leq v$ $n.a < v$ $P'_a(v) = n.a \geq v$ $n.a > v$
descending next prior	card(R)／2	$P'_a(v) = n.a \geq v$ $n.a > v$ $P'_a(v) = n.a \leq v$ $n.a < v$
nothing	card(R)	

C．アクセスコスト関数

Aにおいて求めた結合度(ct_{ij})と選択度(st_i)を用いて，与えられたアクセスパスのアクセスコストを求める。あるアクセスパスTのアクセスコストは，アクセスコスト関数ac(T)によって求められる。先に述べた様にアクセスコストは，パス内のアクセスされるオカーランス数の期待値である。

アクセスパスは7.4.3で述べる様に，木構造をしている。ノードは，DBTG問合せグラフ(DQG)ノードに対応し，枝は，セット型を介したアクセス関数に対応している。ここで，以下のものを定義する。

T　≙　アクセスパスを表わす木(アクセス木)
N(T)　≙　アクセス木T内のノードの集合
　　各ノードは，レコード型に対応している。
B(T)　≙　アクセス木T内の枝の集合
　　≙　$\{ (a_i, a_j) \mid a_i \in N(T) \wedge a_j \in N(T) \}$

a ($\in$ N(T)) をアクセス木内のノードとする。

succ(a) $\triangleq$ ノードaの子ノードの集合

$\triangleq$ $\{ a_i \mid (a, a_i) \in B(T) \}$

prec(a) $\triangleq$ ノードaの親ノード

$\triangleq$ a_i s.t. $a \in succ(a_i)$

root(T) $\triangleq$ Tの根ノード

アクセス木は，互いにセットオカーランスによってリンクされたレコードオカーランスから成っている。アクセス木Tに対して，オカーランス木 ot を定義する。

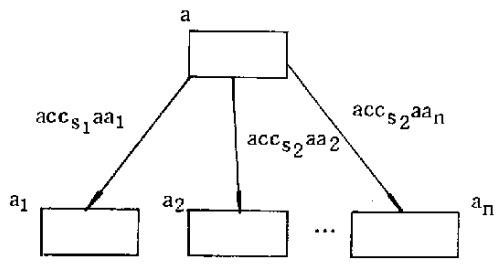


図7.16 アクセス木 T

アクセス木T内のあるノードaを考える。$a_1, a_2, \cdots, a_n$ をaの子ノードとする〔図7.16〕。

i.e. $succ(a) = \{a_1, \cdots\cdots, a_n\}$

又 $arec(a) = R$,

$arec(a_i) = R_i$

$(i = 1, 2, \cdots\cdots, n)$

ここで、$arec : A(T) \rightarrow REC$

$\forall a \in A(T)$ $arec(a) = r$ は，aが表わすレコード型。

図7.16のアクセス木Tから生成されるあるオカーランス木は，次の様になる。

$$N(ot) \triangleq \{ r \mid a \in N(T) \wedge r \in arec(a) \}$$

$$B(ot) \triangleq \{ (r_i\ r_j) \mid a_i \in N(T) \wedge a_j \in N(T) \wedge r_i \in arec(a_i) \wedge r_j \in arec(a_j) \wedge r_i \in N(ot) \wedge r_j \in N(ot) \wedge r_j \in accs_{ij} a_i a_j (r_i) \} \quad \cdots\cdots (15)$$

アクセス木Tの根ノード(R)のオカーランスから構成可能なオカーランス木otの集合をOTとする。OT内のオカーランス木の総数をNOTとする。

$$NOT = \left(\prod_{\substack{\forall a_i, a_j \in N(T) \\ s.t.\ a_j \in succ(a_i)}} ct_{a_i a_j} \right) \quad \cdots\cdots (16)$$

Tに対するオカーランス木の集合OTの中で，条件を満足するオカーランス木(satisfiable occurrence tree or sot) を次の様に定義する。

$$sot \in OT$$
$$\forall r \in N(sot) \qquad rf_a(r) = true \quad \cdots\cdots (17)$$

T内のsot数NSOTは，(18)式の様になる。

$$NSOT = NOT \cdot \prod_{a \in N(T)} st_a \qquad \cdots\cdots (18)$$

アクセス木Tの選択度 st_T は，Tの根ノードのあるオカーランスが，条件を満足するオカーランス木（sot）を持つ確率を表わしている。

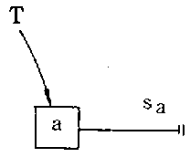
$$st_T \triangleq NSOT / NOT \triangleq \prod_{a \in N(T)} st_a \qquad \cdots\cdots (19)$$

以上を用いて，アクセス木Tのコスト関数 ac(T) の定義を行なう。

ここで

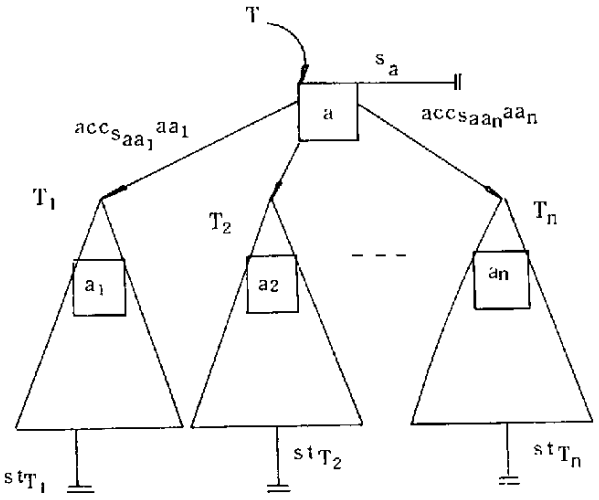
oc(T) ≙ Tの根ノードの1つのオカーランスからアクセスされるオカーランス数

i) Tが1ノードの時



$$\left.\begin{aligned} N(T) &= \{a\} \\ B(T) &= \phi \\ oc(T) &= 1 \\ st_T(T) &= s_a \end{aligned}\right\} \qquad \cdots\cdots (20)$$

ii) Tの根ノードがnコの部分木をもっている時



$a = root(T)$

$succ(a) = \{a_1, a_2, \cdots\cdots, a_n\}$

ここで

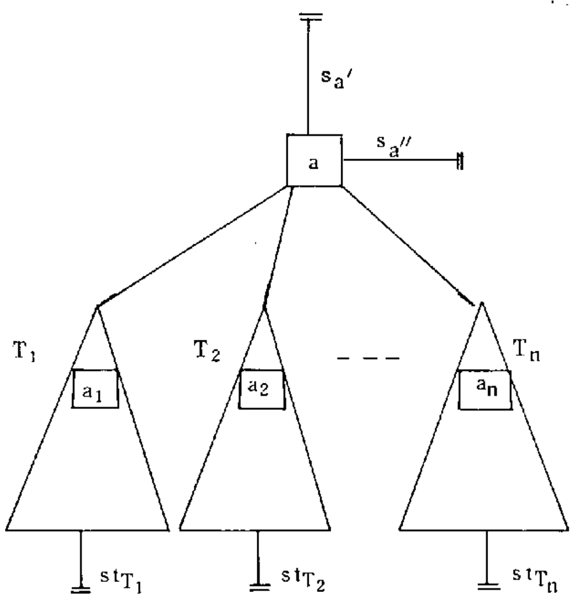
$a_i = root(T_i)$

$$\left.\begin{aligned} st_T &\triangleq st_a \cdot st_{T1} \cdot st_{T2} \cdot \cdots\cdots \cdot st_{Tn} \\ oc(T) &\triangleq 1 + st_a \cdot ct_{aa_1} \cdot oc(T_1) \\ &\quad + st_a \cdot st_{T1} \cdot ct_{aa_2} \cdot oc(T_2) \\ &\quad \vdots \\ &\quad + st_a \cdot st_{T1} \cdot st_{T2} \cdot \cdots\cdots \cdot st_{T_{n-1}} \cdot ct_{aa_n} \cdot oc(T_n) \end{aligned}\right\} \cdots (21)$$

$$\triangleq 1 + st_a(ct_{aa_1} \cdot oc(T_1) + st_{T_1} \cdot (ct_{aa_2} \cdot oc(T_2) + \cdots + st_{T_{n-2}}(ct_{aa_{n-1}} \cdot oc(T_{n-1}) + st_{T_{n-1}} \cdot ct_{aa_n} \cdot oc(T_n)) \cdots))$$

アクセス木は，pre-orderにアクセスされていくとする。ノードaのあるオカーランスが部分木$T_1, T_2, \cdots\cdots, T_i$ の中に条件を満足するオカーランス木（sot）を持つ確率は，$st_a \cdot st_{T_1} \cdot \cdots\cdots \cdot st_{T_i}$ となる。これは又，T_{i+1} 番の木が，アクセスされる確率でもある。従がって，より多くのオカーランス木を持つ部分木を，より後にアクセスするならば，これ以前の部分木によって，より多くselect されることになり，アクセス空間を大幅に縮小できることになる。

iii）Tの根ノードのアクセス



$st_a' \triangleq$ aに関するprimary restriction formula(prf_a)の選択度

$st_a'' \triangleq$ aに関するprf_a以外のrestriction(rf_a)の選択度

$st_a \triangleq$ aのrestriction(rf_a)（$rf_a = prf_a \wedge nrf_a$）の選択度

i.e. $st_a = st_a' \cdot st_a''$

$acc(T) = card\ (range(a))\ st_a' \cdot oc(T)$ ……(22)

ここで

$$OC(T) = 1 + st_a' \cdot (st_{aa_1} \cdot oc(T_1) + st_{T_1} \cdot (st_{aa_2} \cdot oc(T_2) + st_{T_2} \cdot (st_{aa_3} \cdot oc(T_3) + \vdots\ st_{T_{n-2}}(ct_{aa_{n-1}} \cdot oc(T_{n-1}) + st_{T_{n-1}} \cdot ct_{aa_n} \cdot oc(T_n))\cdots\cdots))) \quad \cdots\cdots (23)$$

$$st_T = st_a' \cdot st_a'' \cdot st_{T_1} \cdot st_{T_2} \cdot \cdots\cdots \cdot st_{T_n} = \prod_{\forall a \in N(T)} st_a \quad \cdots\cdots (24)$$

Tのアクセスコストacc(T)は，(22)式で定義される。

7.4.3　DQG分割アルゴリズム

非主結合に対応した非セットリンクに対しては，CODASYL DBTG モデルは，対応したアクセスパスを有しない。このことは，一般に，この非セットリンクを処理するためには，非主結合リンクに対応した集合演算が必要になる。

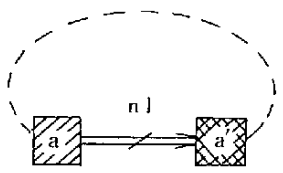


図7.17　非セットリンク

しかし，ある非セットリンク n1 の正ノードaと反ノードa'がある時，aとa'との間にセットリンク(s1)から成る他のパスが存在する時〔図7.17〕，aとa'との間に集合演算は必要ない。

即ち，一方がこのアクセスパスの中で先にアクセスされたならば，他方でアクセスされたオカーランスが異なっていることを確めればよいからである。このため，DBTG問合せグラフ(DQG)からアクセスパスを生成するためには，まず，次の2種の非セットリンクを分ける必要がある。

1) 集合演算の必要な非セットリンク
2) 集合演算の不要な非セットリンク

セットリンクによって連結なノード集合に対しては，集合演算は不要である〔TAKIM80a〕ことから，DQGをサーチして，この連結なノード集合をみつけることが必要になる。これは図7.18に示す様なDQG分割アルゴリズム(DQGD)によって行なえる。

1) DQG内の非セットリンクを一時的に切断する。
2) DQG内のセットリンクを，depth-first にサーチして，連結なDQGノードの集合のグループ(DQG_i)をつくる。
3) 各DQGノードグループ(DQG_i)に対して，〔TAKIM80a〕〔7.4.4〕によって，アクセス木を生成する〔DFA〕。
4) 各アクセス木からCOBOL DML プログラムを生成し，CODASYL DBS 上で実行させる〔DMLG〕。
5) 各 DQG_i ごとに生成された結果リレーションに対して，DQGの非セットリンクに対応した結果処理(非主結合処理もふくむ)を行なう〔JQGP〕。
6) 結果を編集して，ユーザに出力する。

図7.18　DQG分割アルゴリズム(DQGD)

同一部分グラフ内に，あるERQGノードに対する正ノードと反ノードとが存在する時，これらのノードはカップル化(coupled) されていると言う。カップル化された正ノードと反ノードとは，部分グラフ内のセットリンクを介して連結である。従って，DBTGモデルのアクセスパ

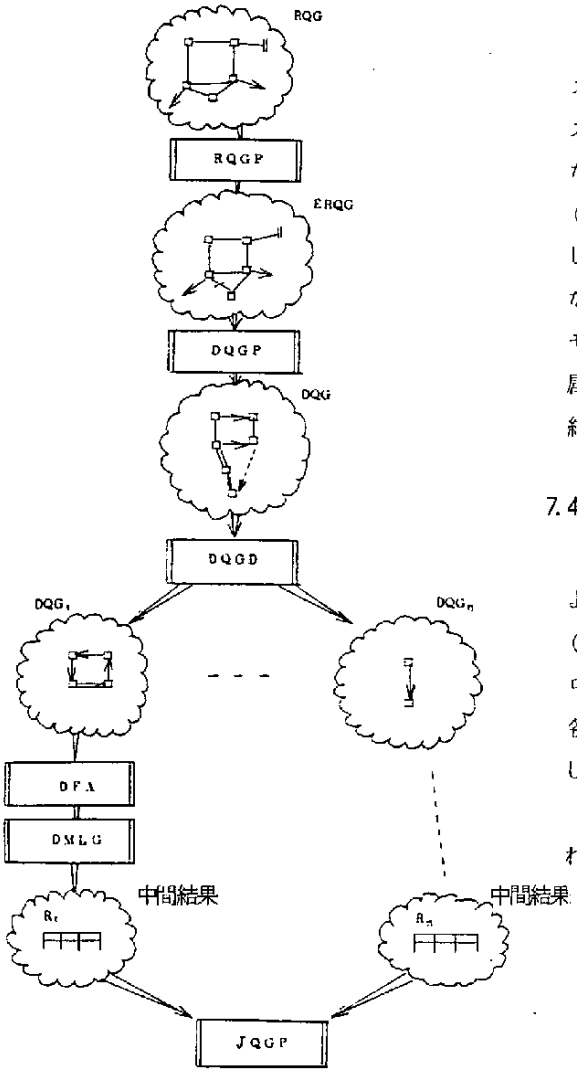


図 7.19　問合せ変換の概要

スとしてのセット型を通して両者をアクセスできる一方，同一部分グラフ内に存在しない正ノードと反ノードとは，孤立化（isolated）されていると言う。孤立化した正ノードと反ノードとの間には，連結なセットリンクは存在しないので，DBTGモデルのアクセスパスは独立して，各々が属する2つの部分グラフで生成される中間結果の集合演算が必要になる。

7.4.4　DFAアルゴリズム

DQG分割アルゴリズム（DQGD）によって分割された連結な部分グラフの各々（DQG$_i$）に対して，COBOL DMLプログラムを生成し，必要な中間結果を得る。各部分グラフから生成された中間結果に対して，必要な結果処理を行なう。

各部分グラフから中間結果として出力されるべき属性としては次のものがある。

1）LCS問合せの目標属性

2）孤立正ノード又は孤立反ノードの主キー属性値

3）部分グラフ間の非主結合属性値

DBTGモデルにおけるアクセスパスは，セット型とレコード型をナビゲーショナルにたどっていくものである。DBTG問合せグラフ（DQG）はレコード型をノードとし，セット型をノード間のリンクとするグラフ構造をしている。このグラフ表現から，アクセスパスを生成するためには，グラフ内の全てのセット型をユニークにふくむ木（これをアクセス木と呼ぶ）を生成せねばならない。DQGからアクセス木（AT）を生成するアルゴリズムは，DFA〔TAKIM80a〕と呼ばれるものである。アクセス木は，ATノードはDQGノードに，AT枝はDQGアークに対応している。アクセス木生成の目標としては，次のものがある。

1）中間結果の数を最少化する。

2）アクセスされるオカーランス数をなるべく少なくする。

DFAは，第1の目標を達成し，何等の中間結果ファイルを必要としない。第2の目標に対して

は，よりオカーランス数の少ないものを，より早くアクセスするという簡単なヒューリスティックスを用いることによって，アクセス空間を縮小できる。

ＤＦＡは，ＤＱＧのアークを縦型にサーチしながら，アクセス木（ＡＴ）をつくるものである。ＤＦＡを図7.20に示す。

0) DQG_i に対して

DN ≙ { d | d = DQG_i 内のノード }

SLNK ≙ { sl | sl = DQG_i 内のセットリンクの集合 }

pushdown (Λ) ； slo ← NIL ；

1) ＤＮ内の全てのノードに対して，

d s.t. ∃ d∈DN ∀d′ ∈ DN(d′ ≠ d)

acc(d) ≦ acc(d′) をもとめる。

これを，根ノードとする。

2) ＤＱＧノードｄのマーク。 ｄが既にマークされていれば，ｄは合流ノード（ＣＮ）になる。

3) ＳＬＮＫをサーチして，sl s.t. sl = (d, d_i) 又は sl = (d_i, d)をみつける。

もし，みつかれば pushdown (d,sl) ； sl と slo の時，ｄをＳＮとマーク。

SLNKから sl を delete SLNK ← SLNK − {sl} ；

d ← d_i ； slo ← sl ； go to 2) ；

4) みつからない時 ppup (d, sl) ；

d ≠ Λの時 go to 3) ；

d = Λの時 terminate ；

図7.20 ＤＦＡアルゴリズム〔TAKIM80a〕

ＤＦＡは，ＤＱＧ内のアーク（セット型）をユニークにたどるものであることから，ＤＱＧ内にループが存在する時，ループ内のあるＤＱＧノードは２回サーチされることになる。このことは，あるＤＱＧノードに対応したＡＴノードを，ＤＱＧノードに対する合流ノード（confluent node）と呼ぶ。

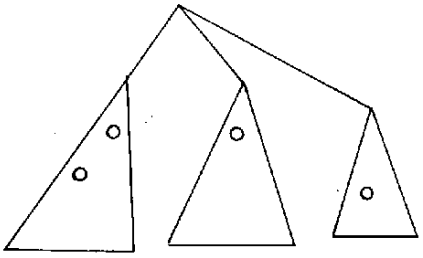
∀d ∈ DN D(d) ≙ { a | a∈ATN ∧ d = d_n(a) }

∃d∈DN card(D(d)) > 1 なる時

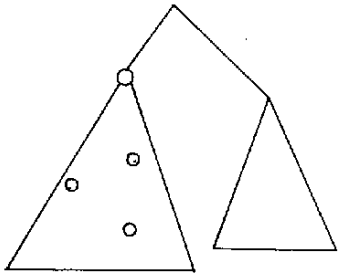
a∈ D(d) を，ＤＱＧノードｄの合流ノードと呼ぶ。

合流ノードとなるレコード型（range(d)）は，アクセス木に対する１つのオカーランス木を得るために，異なったセット型（ＡＴ枝）を介して複数回アクセスされることを表わしている。このことは又，条件を満足するオカーランス木内のあるＤＱＧノードｄに対する合流ＡＴノードのオカーランスは，同一でなければならないことでもある。

アクセス木の1つの枝は，ＤＱＧのセットリンクに対応しているために，枝の親と子とは，一般に１：ｎ（ｎ≧１）の関係性がある。従って，ある２つの合流ノードが異なった部分木内に存在するならば，互いに複数個のオカーランスを求めてしまう。ふつうならば，このオカーランス集合の積を取る集合演算と，オカーランス集合を格納するための中間結果ファイルとが必要になってしまう。もし，あるＤＱＧノードのある合流ノードを根ノードとする部分木内に，他の全ての合流ノードをふくめることができるならば〔図7.21〕，上記の問題は解決する。即ち，全ての合流ノードは，この根の合流ノードのオカーランスと同一であればよいので，合流ノードのアクセスにおいて，ただ根の合流ノードのＤＢキーとの比較を行ない同一であることをチェックするだけでよくなる。又，ＤＱＧノードの制限式に対するチェックもただ根のオカーランスに対してのみ行なえばよくなる。更に，目標属性値の出力も根の合流ノードだけでよくなる。



a) アクセス木(not DFA)



b) DFAによるアクセス木

○：合流ノード

図 7.21

我々のＤＦＡアルゴリズムは，上記の目標を達成している。即ちグラフを縦型にサーチしているために，複数のセットリンクをもつノードは，はじめにサーチされたリンク以外は，必ずこのノードを親にしてサーチされるか又は，このサーチの最後につかわれるかである。

ＤＦＡでは，アクセスされるオカーランス数を減少するために，次の様なヒューリスティックスを用いている。

1) ＡＴの根ノードとしては，ＤＱＧノードのなかで，根ノードとした時のアクセスされるオカーランス数の期待値の最少のもので選ぶ。同一のものがあれば，条件を満足するオカーランス数の最少のものを選ぶ。

$$d \in DN \quad s.t. \quad \forall d' \in DN$$
$$d' \neq d \ \wedge$$
$$acr(d) \leq acr(d')$$

ここで $\forall \alpha \in DN \quad acr(d) = card(range(d)) \cdot st_{d'}$

2) 次の子ノードを見つける時，可能な複数のＤＱＧノードのなかでアクセスされるオカーランス数の最少のものを選ぶ。即ち，結合度の最少のものを選ぶ。

もし，結合度の同一のものがあれば，選択度が少ないものを選ぶ。

1)は，まず始めに，最も速く条件を満足するものをみつけるようにする。CALC 等の直接アクセスパスをもったノードが根ノードとして選ばれる。2) では，結合度の高いものを，後でアクセスすることによって，これに対する選択度を高め，アクセス空間で縮小する。

7.4.5 アクセス木

DFAによって，DQGから生成されたアクセスパスは，アクセス木（AT）と呼ばれる木構造によって表わされる。アクセス木のノード（ATノード）は，DQGノードに対応して，レコード型を表わし，枝（AT枝）は，DQGのセットリンクに対応してセット型を表わしている。このアクセス木（AT）は，次の様に定義される。

AT ≜ (ATN, ABR, RLNK, TLNK)

ATN ≜ {a} ≜ ATノードの集合

a ≜ (d, arec, adn, atype, par)

arec : ATN → REC

$\forall a \in ATN\ \exists r \in REC\ \ arec(a) = r$

rは，ATノードaに対応したレコード型

adn : ATN → DN

$\forall a \in ATN\ \exists d \in DN\ \ adn(a) = d$

dは，ATノードaに対応したDQGノード

rec (adn(a)) = arec(a)

atype: ATN → {NORM, CN, SN, PRO, ANTI}

par : ATN → ATN

aがCN，PRO，ANTIの時，アクセス木内で最初に現われたCN又は coupled ノードへの pointer

ABR ≜ {b} ≜ AT枝の集合

b ≜ (a_1, a_2, aset, dsl)

ここで $a_1, a_2 \in$ ATN で，a_1 は a_2 の親ノードである。

aset : ABR → SET

$\forall b \in ABR\ \exists s \in SET\ \ aset(b) = s$

sは，AT枝bに対応したセット型

dsl : ABR → SLNK

$\forall b \in ABR\ \ \exists sl \in SLNK\ \ dsl(b) = sl$

slは，bに対応したセットリンク

$arec(a_2)$ = owner of aset(b) の時

a_2 を owner ノードと呼ぶ

$arec(a_2)$ = member of aset(b) の時

a_2 をＮＥＸＴノードと呼ぶ

あるＤＱＧノードｄに対する合流ノード（ＣＮ(ｄ)）とは，次の様に定義できる。

$$CN(d) \triangleq \{ a \mid adn(a) = d \}$$

アクセス木をpre-orderにサーチした時現われるＣＮ(ｄ)内ノードを，現われる順に並べた集合を

$$CN'(d) \triangleq \{ a_1, a_2, \cdots\cdots, a_n \}$$ とする。

この時，ｄの全てのＤＱＧノード a_1，a_2，……，a_n は，a_1 を根ノードとする部分木内に全てふくまれる。

7.5 ＤＭＬ 生 成

アクセス木（ＡＴ）は，ノードとしてレコード型，枝としてセット型を表わし，木内の枝とノードとの対は，ＤＢＴＧモデルにおけるアクセス単位を表わしている。アクセスパスは，アクセス木ＡＴ内のオカーランス木を縦型にサーチすることに対応している。

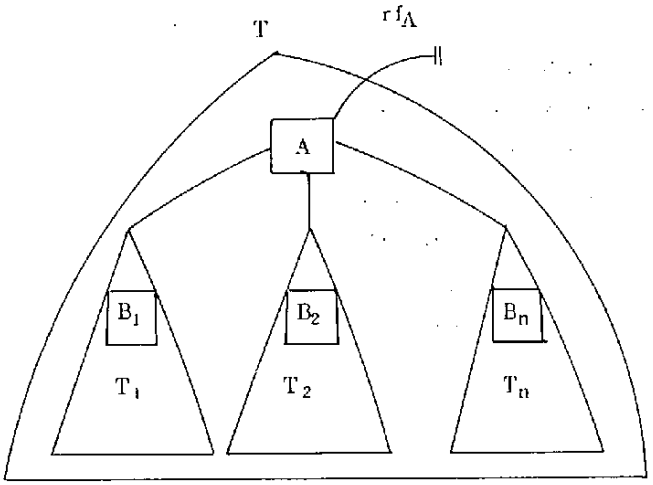


図 7.22　アクセス木 T

7.5.1 オカーランスのアクセス

ここでＸをあるＡＴノードとする時，このノード内のオカーランス x(∈ X)に対して，

ＯＴ(x)≜xを根ノードとするオカーランス木の集合

ＳＯＴ(x)≜ＯＴ(x) 内の条件を満足するオカーランス木の集合

（ＳＯＴ(x)⊆ＯＴ(x)）

図7.22の様なアクセス木（Ｔ）を考えてみる。ここでＡはＴのルートノードであり，B_1，……，B_n は，各々Ａの部分木 T_1，……，T_i の根ノードである（i.e. $succ(A) = \{B_1, B_2, \cdots\cdots, B_n\}$）。この時，a（∈Ａ）オカーランスが，条件を満足するノードであるとは，次の必要十分条件を満足せねばならない。

1. $rf_A(a) = true$　でかつ
2. $B_i \in SUCC(A)$ なる B_i が存在すれば，

ある $b_i \in acc_{AB_i}(a)$ に対して，$SOT(b_i) \neq \phi$　である。

ＯＴ(a) と ＳＯＴ(a) とは，次の様なオカーランス要素とするリスト構造として定義される。

$$OT(x) \triangleq (a\ BOT(a,B_1)\ BOT(a,B_2)\ \cdots\cdots BOT(a,B_n)) \quad \cdots(1)$$

$$SOT(x)\begin{cases} \triangleq (a\ BSOT(a,B_1)\ BSOT(a,B_2)\cdots\cdots BSOT(a,B_n)) \cdots(2) \\ \qquad \underline{if}\ \ rf_A(a) = true \\ \triangleq \phi \qquad \underline{otherwise} \end{cases}$$

ここで $\{ b_{i1}, \cdots\cdots, b_{il_i} \} = acc_{AB_i}(a)$ とすると,

$$BOT(a, B_i) \triangleq (OT(b_{i1})\ OT(b_{i2})\cdots\cdots OT(b_{il_i})) \quad \cdots\cdots (3)$$

$$SOT(a, B_i) \triangleq (SOT(b_{i1})\ SOT(b_{i2})\cdots\cdots SOT(b_{il_i})) \quad \cdots\cdots (4)$$

あるリスト構造をL，この中のある要素を l_i とすると，l_i の右隣りの要素を l_{i+1} とすると，

$next(L, l_i) = l_{i+1}$ なる関数 next を導入する。

lnext は，l_i と同 level の要素の次のものである。

$lnext(L, l_i) = l_j$

アクセス木内の B_i 内のあるオカーランス b_{ij} ($j=1, 2, \cdots\cdots, l_i$) の成功復帰オカーランスと，失敗復帰オカーランスを，もとめる関数を各々 SRTRN, FRTRN とする。

$1 \leq j < l_i$

$$SRTRN(b_{ij}) \triangleq \begin{cases} b_{ij+1} & \underline{if}\ \ SOT(b_{ij}) \neq \phi \\ \Lambda & \underline{otherwise} \end{cases}$$

$$FRTRN(b_{il_i}) \triangleq \begin{cases} b_{ij+1} & \underline{if}\ \ SOT(b_{ij}) = \phi \\ \Lambda & \underline{otherwise} \end{cases}$$

$j = l$

$$SRTRN(b_{ij}) \triangleq \begin{cases} next(b_{ij})\ \underline{if}\ SOT(b_{i1}) \neq \phi \ \vee \\ \qquad \cdots\cdots \vee\ SOT(b_{il_i}) \neq \phi \\ \Lambda \qquad \underline{otherwise} \end{cases}$$

$$FRTRN(b_{ij}) \triangleq \begin{cases} lnext(b_{ij})\ \underline{if}\ SOT(b_{ij}) = \phi \ \wedge \\ \qquad \cdots\cdots \wedge\ SOT(b_{il_i}) = \phi \\ \Lambda \qquad \underline{otherwise} \end{cases}$$

アクセス木内で，あるATノードaの上方の枝をbとする。

即ち $b = (a', a, aset, dsl)$

この時

$$a = \begin{cases} \text{NEXTノード} & \underline{if}\ \ arec(a) = member\ (aset(b)) \\ \text{OWNERノード} & \underline{if}\ \ arec(a) = owner\ \ (aset(b)) \end{cases}$$

$c = next(b_{ij})\ (j = l_i)$ の時

$c \in C$ なるノードCは

$$C = \begin{cases} sister(B_i) & \underline{if}\ \ sister(B_i) \neq \phi \\ last\text{-}next(B_i) & \underline{otherwise} \end{cases}$$

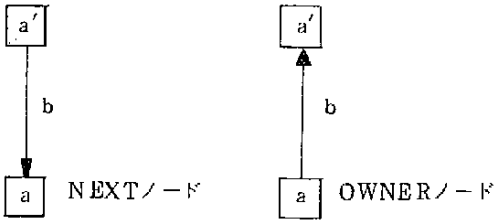


図 7.23

last-next(B_i) は，B_i 上方で，最後に現われたNEXTノードである。アクセス木のあるノードAから，Bまでのパスを考える。このパス内のノードよりも

（A，$C_1, C_2, \cdots\cdots, C_n$，B）とする。

ここで

$C_1 \in succ(A)$，$C_{i+1} \in succ(C_i)$，

$B \in succ(C_n)$，この時，

AがNEXTノードで，$C_1, C_2, \cdots\cdots, C_n$ がどれもNEXTノードでない時，AをBの last next ノードと呼ぶ。

即ち last-next(B) = A

図7.24 のノードBは，ノードC，D，F，Gのlast-nextノードである。

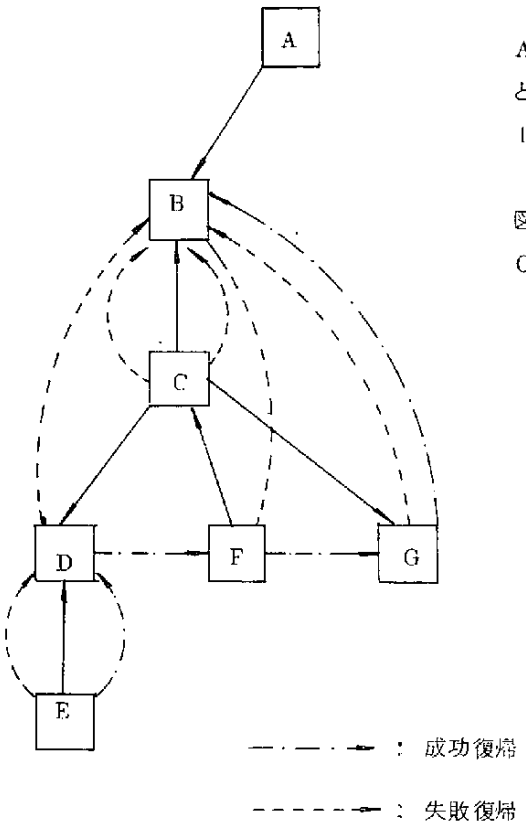


図7.24　成功及び失敗復帰

ここで，ノードB_i 内のオカーランス(b_i) の成功復帰オカーランス（SRO）と失敗復帰オカーランス（FRO）を定義する。

図7.22 で $acc_{AB_i}(a) = \{ b_{i1}, b_{i2}, \cdots\cdots, b_{il_i} \}$ ($l_i > 1$)の時

$1 \leq j < l_i$ なる b_{ij} のSROとFROは b_{ij+1} である。

SRTRN(b_{ij}) = if SOT(b_{ij}) $\neq \phi$ then b_{ij+1}

else ϕ ;

$$FRTRN(b_{ij}) = \underline{if}\ \ SOT(b_{ij}) = \phi \quad \underline{then}\ \ b_{ij+1}$$
$$\underline{else}\ \ \phi \quad ;$$

$j = l_i$ の時

$$SRTRN(b_{il_i}) = \underline{if}\ \ SOT(b_{i1}) \neq \phi \vee SOT(b_{i2}) \neq \phi \vee \cdots\cdots \vee SOT(b_{il_i}) \neq \phi$$
$$\underline{then}\ \ next(b_{il_i})$$
$$\underline{else}\ \ \phi \quad ;$$

$$FRTRN(b_{il_i}) = \underline{if}\ \ SOT(b_{i1}) = \phi \wedge SOT(b_{i2}) = \phi \wedge \cdots\cdots \wedge SOT(b_{il_i}) = \phi$$
$$\underline{then}\ \ next(b_{il_i})$$
$$\underline{else}\ \ \phi \quad ;$$

7.5.2 指示子 (currency)

CODASYL DBTG モデルのアクセスでは，指示子(currency)の概念が重要である。あるレコード型に対応したATノードが，アクセス木(AT)内に複数個現われる時，各ノードごとに，異なった指示子の値が保持されなければならない。この様な場合としては，次下のものがある。

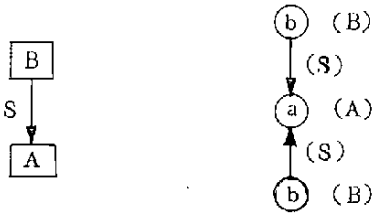
1) Rに対するDQGノードdに対する合流ノード $\{a_1, a_2, \cdots\cdots, a_n\}$ が存在する時。

2) あるレコード型(R)に対して異なったDQGノード $d_1, \cdots\cdots, d_n$ が存在する時，これは，あるLCSリレーションに対して，複数の組変数を定義した場合である。

さらに，セット型Sについて，次の場合，指示子が問題になる。

3) アクセス木内のあるATノード(a)の上方の枝(b_1)と下位の枝(b_2)とが同一のセット型に対応して，aは上方の枝のセット型の子レコード型である時〔図7.25〕。

この場合には，対応するATノードaにおいて，指示子の情報を保持する必要がある。



DBTG 図式　　AT

図 7. 25

7.5.3 アクセスパススキーマ

アクセスパスのスキーマは，ノードのtその上方の枝bとの対から成るアクセス単位(access unit)のシーケンスとして表わせる。アクセス単位は，アクセス木(T)のノードを縦型にサーチすることによって得られる。

$$access\ path\ (T) = (\ v_1, v_2, \cdots\cdots, v_m\ )$$
$$v_i = (a, b, stype, ntype, apt, rf_a, tl_a, srtrn, frtrn\ )$$

ここで　a　≜　ATノード

b　≜　AT枝

stype≜ { MEM　if　arec(a) = member (aset(b))
OWNER　if　arec(a) = owner　(aset(b))

ntype≜ { NORM　通常のノード
CN　合流ノード
SN　synonym ノード
PRO　正ノード
ANTI　反ノード

apt　≜ { v_i　if (ntype = NORM ∨ SN)又は，あるDQGノードの最初の合流ノード又は最初のPRO又はANTIノード。
v_{jj} (j < i)　if　DQGノードの2番目以降の合流ノード又はPRO又はANTIノード。

rf_a　≜　ノードaの restriction formula

tl_a　≜　ノードaの target list

srtrn ≜　aの成功復帰アクセス単位

frtrn ≜　aの失敗復帰アクセス単位

7.5.4　アクセスパターン

COBOL DML は，アクセス木から生成されたアクセスパスをもとにして，次の様にして生成される。

1) 共通部分の出力　Identification division
Data division
サブルーチィン

2) 各アクセス単位ごとに
アクセスパターンの照合を行ない，対応するDMLブロックを出力する
DMLブロック内の変数を data division に出力
出力属性をデータ部と出力用サブルーチィンに出力

必要なDMLブロックは，異種性情報（HI）内のDMLI（DMLリレーション）に格納されている。この詳細は，付記ⅡのDMLIを参照されたい。

7.6　結　合　処　理

COBOL DML プログラムは，7.4のDQG分割で分割されたDQG部分グラフごとに生成される。これらの各プログラムは，CODASYL DBTG DBS 上で実行され，各々結果リレーション間では，DQG分割で分割した非セットリンクに対応した主キー属性についての非主結合（≠）

と，部分グラフ間の非主結合（$<$，$\leq$，$=$，$\geq$，$>$，$\neq$）とを行なう必要がある。
ここで

DQG_i ≙ DQGから分割されたi番目のDQG部分グラフ
DML_i ≙ DQG_i から生成されたCOBOL DML プログラム
R_i ≙ DML_i の実行によって生成された結果リレーション
A_i ≙ R_i の属性集合
= $tl(R_i) \cup pjl(R_i) \cup njl(R_i)$
$tl(R_i)$ ≙ 問合せの結果属性の集合
$pjl(R_i)$ ≙ 他の結果リレーションとの結合のための主キー属性集合
$njl(R_i)$ ≙ 非主結合属性の集合

結果リレーションR_iに対する問合せは，次の様に表わせる。

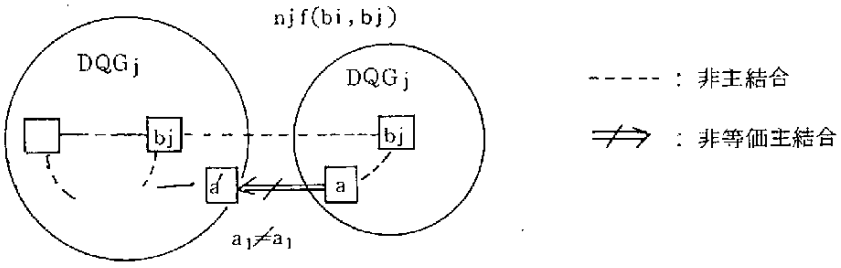


図 7.26　DBTG 部分グラフと結合

JQG ≙ $(JN, JLNK, TLNK)$
JN ≙ $\{ R_i \mid i=1, 2, \cdots\cdots, n \}$
R_i ≙ DQG_i の結果リレーション
$JLNK$ ≙ $(PJLNK, NJLNK)$
$PJLNK$ ≙ $\{ pjl_{ij} \mid i, j=1, 2, \cdots\cdots, n, i \neq j \}$
pjl_{ij} ≙ $\begin{cases} (pnjp(R_i, R_j)) & \underline{if}\ pnjp(R_i, R_j)\ exists \\ \phi & \underline{otherwise} \end{cases}$
$NJLNK$ ≙ $\{ njl_{ij} \mid i, j=1, 2, \cdots\cdots, n, i \neq j \}$
njl_{ij} ≙ $\begin{cases} (njf(R_i, R_j)) & \underline{if}\ njf(R_i, R_j)\ exists \\ \phi & \underline{otherwise} \end{cases}$
$TLNK$ ≙ $\{ tl_i \mid i=1, 2, \cdots\cdots, n \}$
tl_j ≙ R_i の結果属性の集合
≙ $(R_i, tl(R_j))$

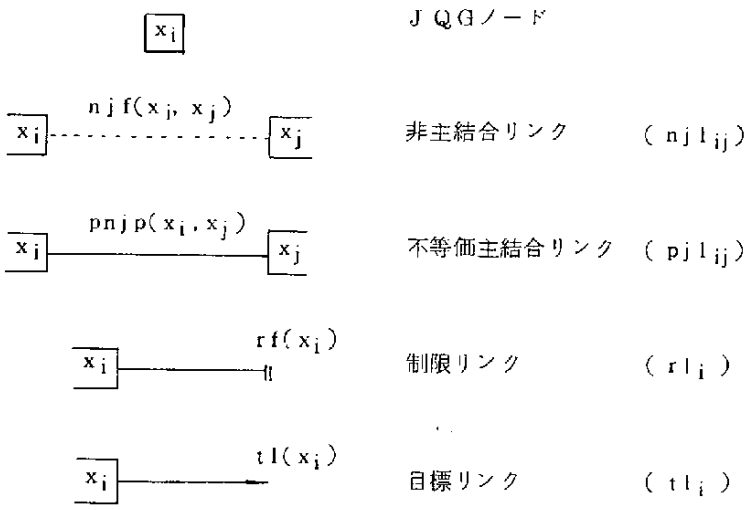


図 7.27　JQG の記述

これは，次の問合せを表わしている。

$$\underline{\text{range}} \quad (x_1, R_1)\cdots\cdots, (x_n, R_n) \ ;$$

$$\underline{\text{retrieve into}} \quad R \quad (\{tl(x_1), \cdots\cdots, tl(x_n)\})$$

$$\underline{\text{where}} \quad \bigwedge_{i=1,\cdots,n-1} \ \bigwedge_{j=i+1,\cdots,n} (pnjp(x_i, x_j) \wedge njf(x_i, x_j))$$

$$\bigwedge_{i=1,\cdots,n} rf(x_i) \ ;$$

このJQGの処理は，CODASYL DBTG モデルが，非等価主結合に対してアクセスパスを備えていないので，DBTG DBS とは独立な処理等によって処理される必要がある。問合せ変換において，生成されるDQG$_i$の数は，多くても問合せ内の不等価主結合の数(＋1)であることから，JQG処理に対しては，次の様な簡単な方法を用いている。

1) カーディナリティの一番小さいリレーション(R)を選ぶ。
2) Rと主結合リンクで結ばれたリレーションの中で，カーディナリティの最少のもの(S)を選ぶ。
3) RとSの結合処理を行なう。

7.7 ビュー定義とアクセス

ローカルデータベースプロセッサ(LDP)は，分散型データベースシステム(DDBS)における異種DBS(特にCODASYL DBS)の共通リレーショナルインタフェースであるとともに，エンドユーザ向けの非手続的リレーショナルインタフェースでもある。このため，CODASYL ス

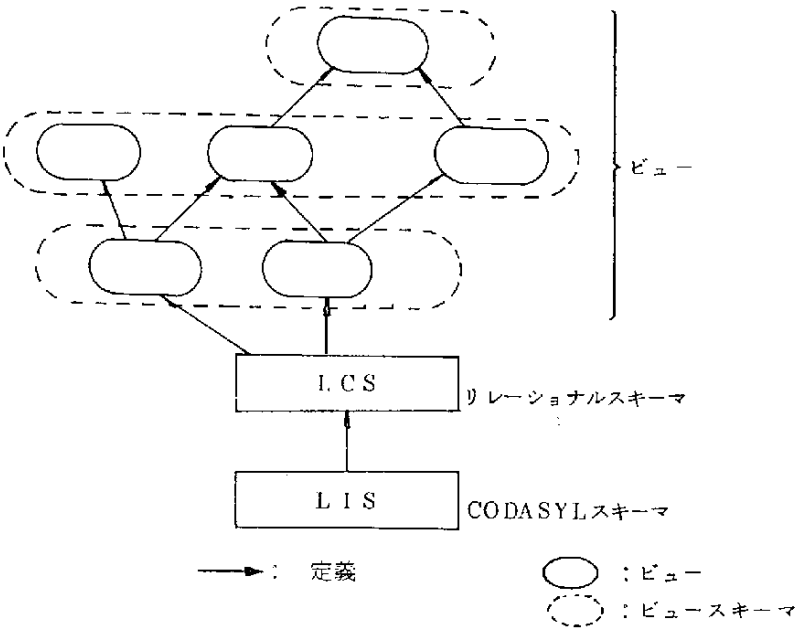


図 7.27　ビュー と LCS

キーマ(i.e. LIS)から変換されたリレーショナルスキーマ(i.e. LCS)上に，ユーザアプリケーションに適したリレーションの記述(ビュー)の定義を許し，これに対する問合せ(ビュー問合せ)を，LCSリレーションに対する問合せに変換する機能が必要になる。ビューは，上述したユーザの使いよさに加えて，視せたくないデータを隠すというセキュリティ面でも利用できる。

LDP(－V1.5)では，ビューの階層的な定義機能と，この定義されたビューに対するアクセス(検索)を許している。

7.7.1 ビュー定義

ビュースキーマは，ビュー定義の集合であり，各ビュー利用アプリケーションの管理者によって定義される〔図7.28〕。

```
〈 view schema 〉 ::= defschema 〈 view-schema 〉 ;  〈 range-statement 〉
                     〈 define-statement 〉   endschena ;
〈 range-statement 〉 ::= range (〈 var 〉, 〈 rel-name 〉 [:〈 view schema 〉] )
                          {(〈 var 〉, 〈 rel-name 〉 [:〈 view schema 〉])} ;
〈 define-statement 〉 ::= define 〈 view 〉 (〈 target-list 〉)
                          [where 〈 qual 〉] ;
```

図 7.28　ビュースキーマの定義

range 文は，ビュースキーマ内の定義文(define 文)が用いる組変数を定義する。組変数は，LCSリレーション又は他のビュースキーマ上のビューに対して定義される。ビュースキーマ内のビューは，互いに結合(join)を行なえることが特徴である。ビューにおける結合がLCS

リレーションの結合〔7.1〕に変換できるために，次の制約が必要である。

1) ビューにおいても，ＬＣＳリレーションのキー属性は保存される。

即ち，ビューにおけるキー属性は，又，ＬＣＳリレーションのキー属性でなければならない。

2) ビューに対する検索において，ビュー間の結合としては，次のものが許される。

a) 同一ビュースキーマ内のビュー間の主キー属性についての＝又は≠の結合（主結合）。

b) a)の結合が存在する時，主キー以外の属性間の任意の結合（非主結合）。

3) 1つのLCSリレーションのキー属性は，ビュースキーマ内の複数のビュー内に現われてはならない。

これらの制約は，ビューにおける主結合を，ＬＣＳリレーションにおける主結合に直接に対応づけるためである。我々は，ＬＣＳにおける意味構造（ＥＳＲとＲＳＲとのリンク）は，ビューにおいても保存されねばならないと考えている〔付記Ⅰのオブジェクトモデルを参照されたい〕。このことによって，ビュー間における結合が，ＬＣＳリレーション間の結合に変換できるものと，できないものとに分けることができる。更に，ビューを通しての更新においても，この考え方は有効であると考えている。

ビュースキーマは range 文の集合と対応したdefine文とから成っている。これは define 文の用いる変数とリレーションとの対応のユニークさを保つためである。即ち，あるdefine文（D_1）が用いる変数ｘと，他のdefine文（D_2）が用いる変数ｘを同一性は，この２つの文が同一のrange 文を用いることを明らかにすることによって保障される。ビュー D_1 とD_2との結合をとる時に，D_1のｘとD_2のｘとが同一であるかないかは，問合せの意味に大きく影響してくる。以上のために，我々は，ビュースキーマの概念を設けた。

ビューＶは，ｎ個のビュー又はＬＣＳリレーション V_1，……，V_n から定義される。

$$V_1 \times V_2 \times \cdots\cdots \times V_n \rightarrow V$$

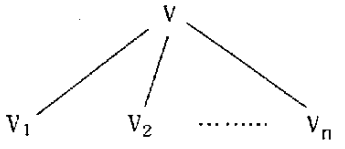


図7.29　ビュー定義木

これは，図7.29の様に，ビュー定義木によって表わされる。現在は，V_1，……，V_n は，同一のＬＣＳ又はビュースキーマに属していると仮定する。異なったスキーマに属している時は，各ビュースキーマ内の定義文の組変数の対応を示す必要がある。ビュー定義は，スキーマ階層に対応して図7.30の示す様な，定義木として表わされる。

ビューの消去は，destroy コマンドによってなされる。

destroy　{〔<view>〕〔：<view schema>〕};

上位層のビューの消去は，下位層に影響を与えないが，下位層の消去は，上位層のビューの消去に伝搬していく。

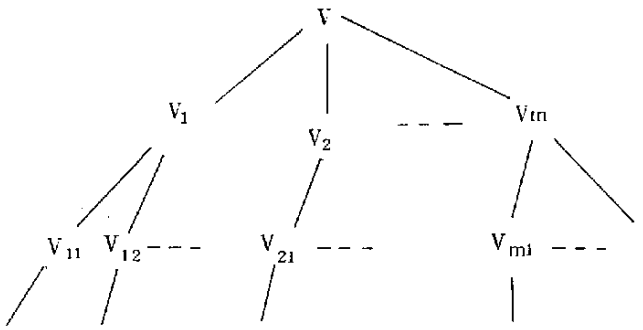


図 7.30　ビュー定義木

7.7.2　ビューアクセス

ビューに対するアクセスは，問合せ変形手法〔STONM76〕によって，LCS問合せ表現に変換される。問合せをRとし，Rの参照するビューは，図7.30 の様なビュー定義木で表わされる。これを示すと図7.31の様な木になる。これをRRT（reference relation tree）と呼ぶ。RRTの各ノードは，問合せRの参照するビューを表わす変数を示し，葉は（$V^{(n)}$）は，LCSリレーションを示している。ただし，同一変数は1つのノードで表わすことにする。即ち，一番下位のノードを上位のビューノードがポイントすることになる。

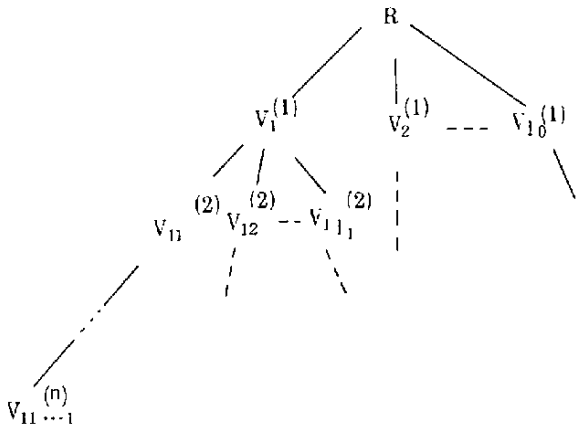


図 7.31　RRT

ここで，i番のビュー $V^{(i)}_{j1\cdots ji}$ が，図7.3 の様な参照関係を持っているとする。

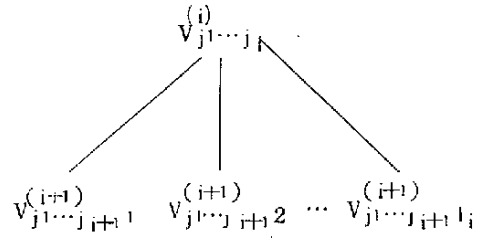


図 7.32

又，ここで

$V^{(k)}_{\alpha}$ の目標リストを

$\{ ga^{(k)}_{\alpha h} = acxp^{(k)}_{oh} \mid h = 1, 2, \cdots, p_\alpha \}$

条件式を

$qual^{(k)}_{\alpha}$ とする。

$$\underline{\text{range}} \quad (l_1, L_1) \cdots\cdots (l_m, L_m) \ ;$$

$$\underline{\text{define}} \quad V^{(i)}_{j_1 \cdots j_i}(\{ ga^{(i)}_{j_1 \cdots j_i h} = \sigma^{(i-1)}_{j_1 \cdots j_i h} \ aexp^{(i)}_{j_1 \cdots j_i h} \mid h = 1, 2, \cdots, p_{j_1 \cdots j_i} \}) \qquad \cdots (1)$$

$$\underline{\text{where}} \quad \sigma^{(i-1)}_{j_1 \cdots j_i h} \quad qual^{(i)}_{j_1 \cdots j_i} \quad \bigwedge_{h'=1, \cdots, l_i} \delta_{h'}^{(i-1)} \ ;$$

ここで

$$\delta_{h'} = \begin{cases} qual^{(i-1)}_{j_1 \cdots j_i h'} & \underline{\text{if}} \quad V^{(i-1)}_{j_1 \cdots j_i h'} \quad \text{は，木の下位のノードに表われない。} \\ true & \underline{\text{otherwise}} \end{cases}$$

$$\sigma^{(i-1)}_{j_1 \cdots j_i h} = \{ aexp^{(i-1)}_{j_1 \cdots j_i h' h} / V^{(i-1)}_{j_1 \cdots j_i h'} \cdot ga^{(i-1)}_{j_1 \cdots j_i h' h} \mid h' = 1, 2, \cdots, l_i \}$$

l_i = LCSリレーション L_i の組変数

l_i は，RRTの葉ノード

(1)を，順に上位のノードに対して行なうことによって，最終的に，ビュー問合せをLCS問合せに変形できる。

$$qual^{(i-1)} = \bigwedge_{h'=1, 2, \cdots, l_i} \delta_{h'}^{(i-1)}$$

$$\sigma^{(i-1)} = \{ \sigma_{j_1 \cdots j_i h'} \mid h' = 1, 2, \cdots\cdots, p_{j_1 \cdots j_i} \} \quad \text{とする。}$$

最終的に問合せRは，次の様になる。

$$\underline{\text{range}} \quad (l_1, L_1) \cdots\cdots (l_m, L_m) \ ;$$

$$\underline{\text{retrieve into}} \quad R(\{ r_q = \sigma^{(n-1)} \sigma^{(n-2)} \cdots\cdots \sigma^{(1)} aexp_q \}) \qquad \cdots (2)$$

$$\underline{\text{where}}$$

$$\sigma^{(n-1)} \ (\sigma^{(n-2)} \ (\cdots (\sigma^{(1)} \ qual \wedge qual^{(1)}) \cdots\cdots) \wedge qual^{(n-2)}) \wedge qual^{(n-1)} \ ;$$

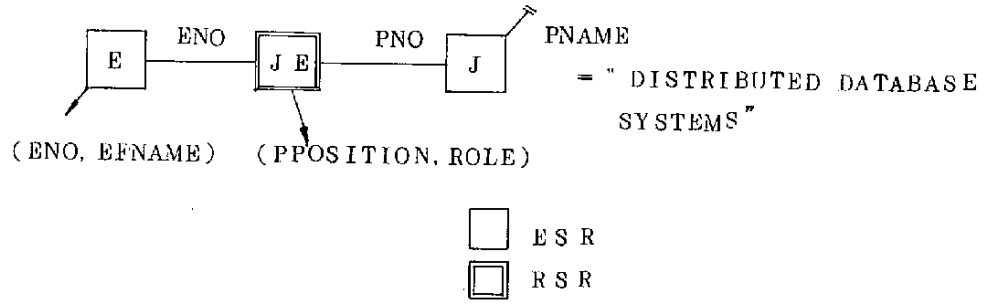
7.7.3 例

1) DDBS－EMP(eno, name, pposition, role)

このviewは，プロジェクト名が"DISTRIBUTED DATABASE SYSTEMS"である従業員のリストを表わしている。

このview定義をQUELで書くと次の様になる。

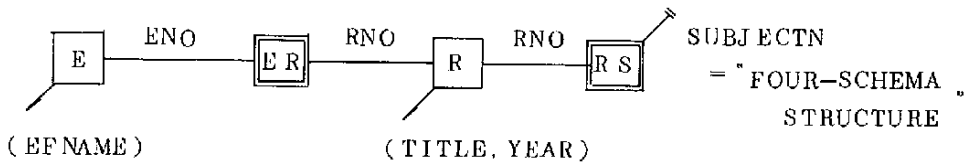
```
ANGE    ( E, EMPLOYEE ) ( J, PROJECT ) ;
ANGE    ( JE, PROJ-EMP-LNK ) ;
EFINE   DDBS-EMP ( E.ENO, NAME=E.EFNAME, JE.PPOSITION, JE.ROLE )
HERE
    E.ENO          = JE.ENO                AND  JE.PNO          = J.PNO    AND
    J.PNAME        = "DISTRIBUTED DATABASE SYSTEMS" ;
```



2) D B S－R E P (name, title, year)

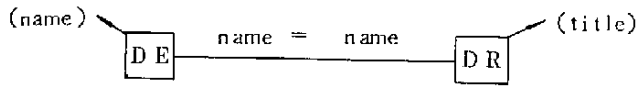
SUBJECT 名が「FOUR－SCHEMA STRUCTURE」であるレポートに関するリレーションをview定義をQUEL で書くと次のようになる。

```
RANGE    ( E, EMPLOYEE ) ( R, REPORTR ) ( S, SUBJECT ) ;
RANGE    ( ER, EMP-REP-LNK ) ( RS, REP-SUBJ ) ;
DEFINE   DBS-REP ( NAME=E.EFNAME, R.TITLE, R.YEAR )
WHERE
    E.ENO          = ER.ENO          AND  ER.RNO          = R.RNO          AND
    R.RNO          = RS.RNO          AND
    RS.SUBJECTN    = "FOUR-SCHEMA STRUCTURE";
```

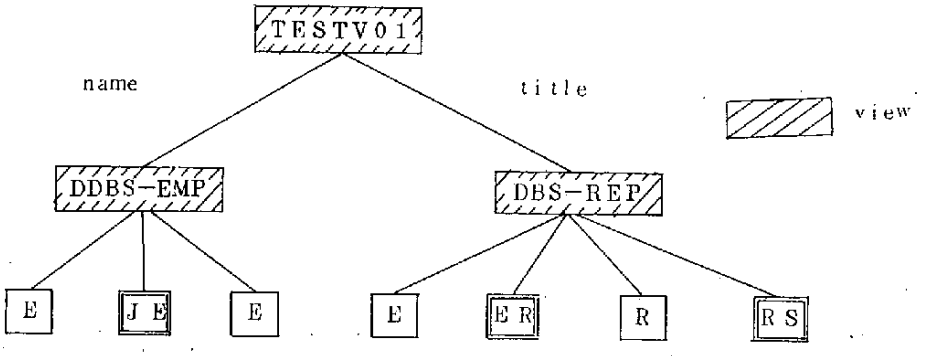


3) 「DDBS project内で,DBSの論文を書いたものをリストせよ。」という問合せは次のようにQUELで書くことができる。

```
RANGE    ( DE, DDBS-EMP ) ( DR, DBS-REP ) ;
RETRIEVE INTO TESTV001 ( DE.NAME, DR.TITLE )
WHERE
    DE.NAME        = DR.NAME ;
```

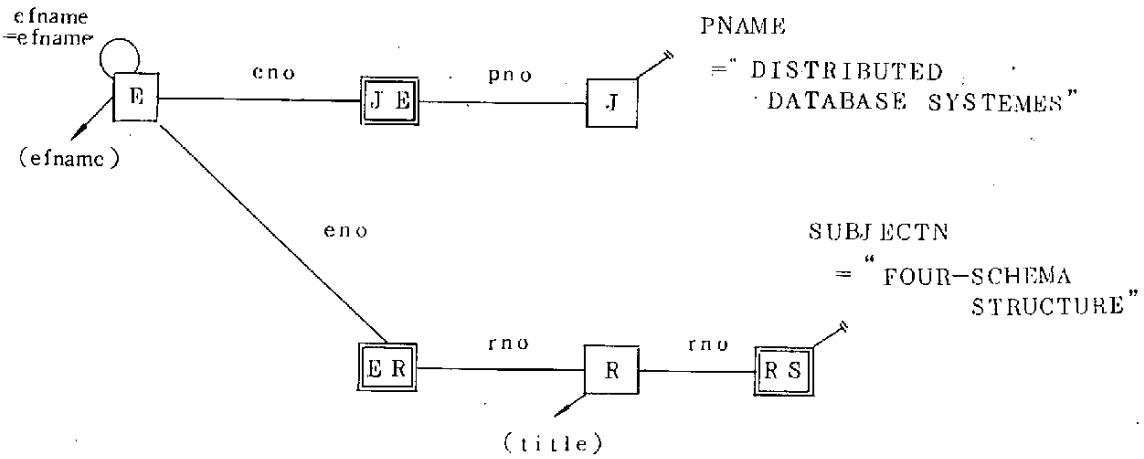


4) view定義の階層構造（view定義木）



5) query modification されてＬＣＳ問合せに変換されたquery

```
RETRIEVE INTO TESTV001 ( NAME=E.EFNAME , TITLE=R.TITLE ) WHERE E.
      EFNAME = E.EFNAME AND E.ENO = ER.ENO AND ER.RNO = R.RNO
      AND R.RNO = RS.RNO AND RS.SUBJECTN = "FOUR-SCHEMA STRU
      CTURE" AND E.ENO = JE.ENO AND JE.PNO = J.PNO AND J.PNAME
      = "DISTRIBUTED DATABASE SYSTEMS" ;
```



この例の結果は，付記Ⅱに示してある。

7.8 評　　価

本節では，我々の開発したローカルデータベースプロセッサ（ＬＤＰ－Ｖ1.5）の評価について述べる。ＬＤＰ－Ｖ1.5は，次の機能を有している。

ＬＤＰ－Ｖ1.5　　1）等価主結合
2）ＡＩＭ（Ｍ－１６０）ＤＢＳの起動
3）ビュー定義とアクセス
4）検　　索

ＬＤＰ－Ｖ1.5は，現在当協会のFACOM M－160上に，PL／Ｉを用いて実現されている。プログラムは，PL／Ｉ文約5,000である。ＬＤＰ－Ｖ1.5は〔TAKIM80a〕に基づいたＬＤＰ－Ｖ１〔JDDBS80〕を出発点としている。このため設計に３人月，実現（プログラム，コーディング，デバック）に５人月を要した。

評価は，主に下記の点について行なった。

1）アクセスコスト関数の正しさ

選択度，結合度に基づいたアクセスコスト関数〔7.4.2〕の正しさを検証するために，この関数に基づいて計算されたアクセスオカーランス数の期待値と，実際のアクセスオカーランス数との比較を行なった。

2）オカーランス数と，応答時間の関連

我々の問合せ変換アルゴリズムの一つの目的は，応答時間を減少することである。このために，7.1節で，応答時間は，アクセスされるオカーランス数に比例するとした。この仮定の正しさの検証した。

3）ソフトウェアの生産性

ユーザにＱＵＥＬの様な，高水準非手続き的言語を提供することによるソフトウェアの生産性の向上を示す。

4）3)と関連して，ＡＩＭ　ＤＢＳ　上に，インタフェースを設けることによるオーバヘッドの増加を調べる。

評価のための目標ＤＢＳとして，付記Ⅱに示したプロジェクトデータベースシステム（PRDBS）を用いた。レコード型のオカーランスは，数１０のオーダーで，全体で数100KBである。これに対する問合せとしては，付記Ⅲの２に示す２５個の問合せを考えた。

図7.3.2には，ＦＩＮＤ文によって位置決め（アクセス）されたオカーランス数（……）と，7.4.2のコスト関数によって求められたオカーランス数の期待値（ ― ）がプロットされている。この図から解かる様に，テスト井20～22，30～32，40～41及び50～51，70のテスト問合せに対して，見積り値と，実際アクセスされた値とが異なっている。これらの特徴は，SUBJECTレコード型に制限式をもっている点である。SUBJECTレコード型は，８８のオカーランスから成っているが，この中で，子レコードを持つものは，高々数割である。問合せの制限式は子レコードを持つものに対して出されているので，子レコードを持たない確率によってunder-estimate　さ

れている。他に，ＰＲＤＢＳ のメンテナンスの不備によって，レコードオカーランス間のリンクの切れているところがあることも，予測と実際との差になっている点もある。ＰＲＤＢＳ 自体のオカーランス数が，絶対的に少ない点が，評価を不十分にしており，今後より大規模ＤＢＳにおいて，選択度の仮定の正しさを検証していきたいと考えている。

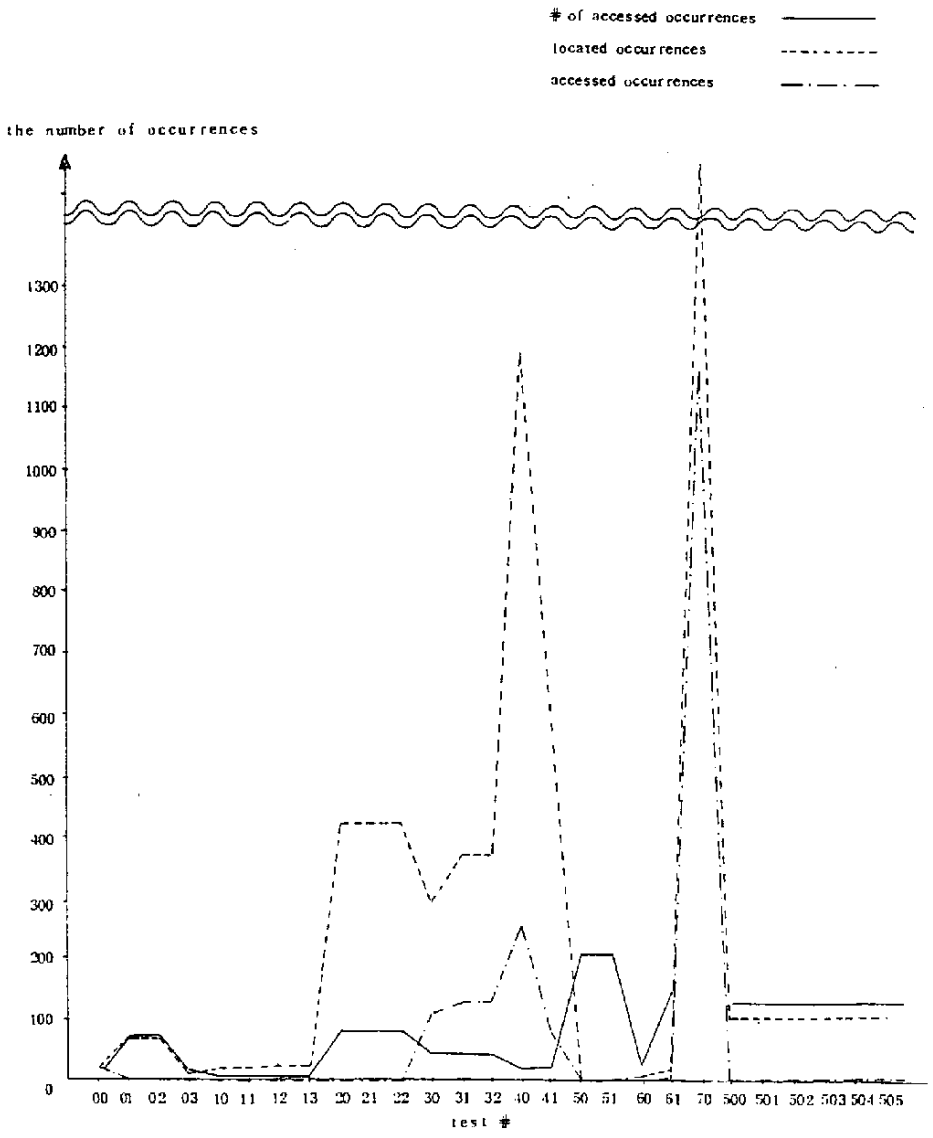


図 7.32　オカーランス数の見つもりと実際

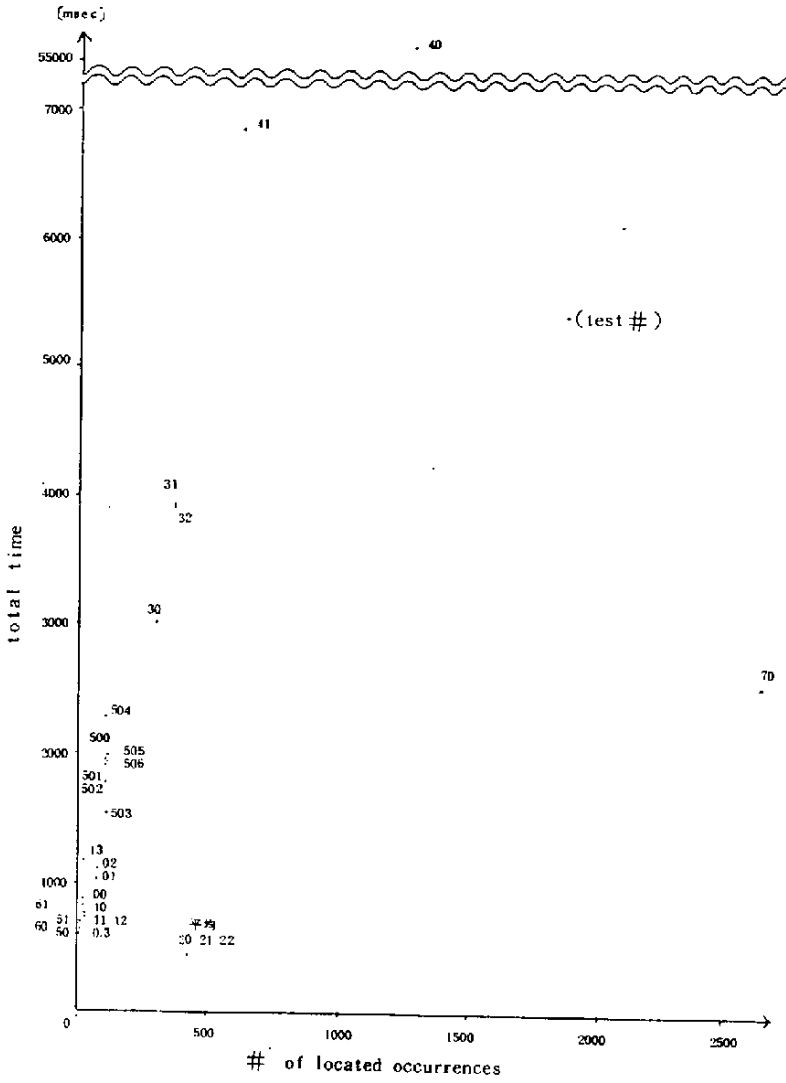


図 7.33　オカーランス数と応答時間

図7.33は，位置決めされたオカーランス数と全処理時間との関連を示している。test # 70と20〜22とが，はずれているが，応答時間はアクセスオカーランス数に依存していることが解かる。このことより，アクセス（位置決め）されるオカーランス数をなるべく減少させることが必要となる。

QUELによる問合せの生成と，同一の問題をCOBOLでつくる場合を比較してみると前者では，平均10〜15分で，問合せの入力を行ない結果を得られる。一方後者ではCOBOL経験者でもコーディング，デバックをいれると4〜5時間（複雑なもの(例 test 70)では数日）を必要とする。

この様に，高水準非手続き言語によるソフトウェアの生産性は，数１０倍から数１００倍高いことが解かる。更に，複雑なプログラム言語の詳細を知らない初心者も，容易に必要とする結果を得ることができる。

リレーショナルインタフェースを設けることによる問題は，そのオーバヘッドにあるといわれている。しかし，我々のテスト例では，簡単な問合せ（ test 0〜32)で4 〜５秒*，複雑な問合せ（ 60〜70 ）で１３〜１５秒*のＣＰＵ時間で処理される。この様にＣＰＵ時間に関する限り，さしたるオーバヘッドとはならないと確信している。前のパラグラフで述べた，ソフトウェアの高生産性によって，このオーバヘッドは十分に償なわれると考えている。

この様に，我々のＬＤＰ（－Ｖ1.５）は非定型的に業務に対して，十分に適用可能である。

7.9 まとめと今後の課題

本章では，分散型データベースシステム（ＤＤＢＳ）における異種ＤＢＳ（ i.e. CODASYL ＤＢＳ）に対する共通リレーショナルインタフェースシステム（これをＬＤＰと呼んでいる）における問合せ変換システムについて論じた。このシステムは，リレーショナル問合せＱＵＥＬを入力として，COBOL DML プログラムを生成し，CODASYL DBS（具体的にはＡＩＭ ＤＢＳ）でこれを実行させ結果を得るものである。現在，実現を終了したＬＤＰ－Ｖ1.５は，次の機能を有している。

1) 検索機能
2) 結合として，主キー属性間の等価結合（主結合）のみを許す。
3) ビュースキーマの階層的定義とアクセス機能
4) ＡＩＭ ＤＢＳの実動
5) CODASYL スキーマからリレーショナルスキーマ自動生成機能

本章ではこのＬＤＰ—Ｖ1.５の機能とともに，現在検討を進めているＬＤＰ－Ｖ２について述べた。ＬＤＰ－Ｖ２は，次の機能を備えている。

1) 結合として，主キー属性間の等価（＝），非等価（≠）結合（主結合）と，非キー属性間の非主結合（＜，≦，＝，≧，＞，≠）
2) 検索に加えて，リレーショナルスキーマ（ＬＣＳ）を通しての更新機能
3) オブジェクトモデル〔付記Ｉ，TANAY80〕に基づいたスキーマ設計
4) 出力結果の処理（フォームの処理）
5) ＡＩＭに加えて，ＡＤＢＳ（ＡＣＯＳ－７００）の実動
6) 統計処理機能（ computational facilities)
7) 最適アクセスパス生成

本報告書ではこの中の1)について特に論じた。現在，２）の更新機能について検討を進めている。更新では，CODASYL モデルの備えているインテグリティ保持機能としてのメンバシップクラス

---

*) Ｍ－１６０ によるデータである。

に対応したＤＭＬの生成がポイントになる。

Zaniolo 〔ZANIC79〕のスキーマ設計法では，null値の問題の発生と，あるレコード型が複数の親レコード型を持っている時のキーの概念のあいまいさが指摘できる。しかしMANDATORY／AUTOMATIC（M／A）に対するメンバレコードオカーランスの挿入を，メンバレコード型に対応するリレーションに対する１つの挿入演算によって処理できるので，CODASYL スキーマの基本演算とリレーショナルモデルの基本演算とを１対１に対応できる利点を有している。これに対して我々のスキーマ設計法では，Zaniolo の前者の問題はなくなるが，後者が問題になる。我々のリレーショナルスキーマでは，セット型は，１つのＲＳリレーションとして独立に存在している。このためM／Aのメンバシップクラスを持ったセット型の子レコードへの挿入を１つの基本演算では行なえない。しかし，我々はこの問題に対して，リレーショナルスキーマ（ＬＣＳ）を，CODASYL スキーマ（ＬＩＳ）に対する抽象データ型としてとらえることによって，ＴタイプのＲＳリレーション挿入に対しては，このＲＳと子レコード型に対応するＥＳリレーションとの２つに対して同時に挿入を行なうオペレーションを定義しておけばよい。現在，この様な方法で更新処理についての検討を進め，来年度，ＬＤＰ－Ｖ２の実現を行ないたいと考えている。

問合せの結果属性間には，CODASYL スキーマの構造に対応して，一般に１：Ｎの関係性が存在している。このため，この結果をユーザに解かりやすく出力する必要がある。いわゆるフォームの処理である。ユーザのインタフェースとしては必須の機能である。

我々のアクセスパス生成法は，１レベルのサーチによって，このなかで最もらしいパスを選択している。これに対して，現在２レベルまでサーチする方法をＤＦＡ２として検討している。又，現在のＤＦＡに対するＣＰＵ時間は，かなり複雑な問合せでも５０～６０ｍ秒であることから，ノード数の少ない問合せ（４～５個）に対しては，可能なアクセスパスを全てサーチすることも有効であると考えている。アクセスコスト関数は，現在アクセスされるオカーランス数の平均値の見積もりを行なっている。

しかし，〔GOODN79, SELIP79〕では，最悪ケースの見積もりを行なっている。この２つの方法のどちらが有効かは，今後実験を通して明らかにしていきたい。

8. まとめと今後の課題

本報告書（Ⅱ部）では，今後の情報システムの核となっていくと思われる分散型データベースシステム（ＤＤＢＳ）のモデル化を試みた。このなかで我々は，次の成果を得られた。

1) 第Ⅱ期ＤＤＢＳ（ローカルネットワーク，異種データベースシステム（ＤＢＳ），小型データベースシステム）の全体アーキテクチャ（これをＪDDBS－Ⅱと呼ぶ）の確立〔第4.5章〕。

2) ＪDDBS－Ⅱにおける問合せの通信処理モデルと，アルゴリズムの提案〔第6章〕。

3) 異種ＤＢＳ（CODASYL DBTG DBS）に対するリレーショナルインタフェースシステムのモデル化（ＬＤＰ－Ｖ２）と実現（ＬＤＰ－Ｖ1.5）— ローカル情報システム〔７章，２章，付記Ⅱ〕。

4) オフィス情報システム（ＯＩＳ）におけるデータベースシステムモデルとしてのフォームフローモデルの提案〔第３章〕。

5) データベースのビュー更新管理のためのモデルとしてのオブジェクトモデルの提案と，データ抽象化技法の確立〔付記Ⅰ〕。

これらの成果によって，今後のデータベースシステム技術の核となる部分のモデル化を行なえたと確信している。

ＪDDBS－Ⅱは，これまでの分散型データベースシステム（e.g. ＳDD－1，POLYHEME）に対して，次の点を特徴としている。

ⅰ）物理的な１：Ｎ通信機能を備えたローカルネットワークの利用

ⅱ）小型データベースシステムの組込み

ⅲ）異種ＤＢＳの組み込み

ⅳ）オフィス情報処理への適用

ＪDDBS－Ⅱは，将来の情報システムの基本的要素となるもので，このモデル化は重要な意味を持っている。これは，将来の情報システム利用が，各サイト（個人もふくむ）で分散して生成された情報を，通信ネットワークを介して結合利用することが中心となっていくためである〔図8.1〕。このために，本年度はＪDDBS－Ⅱのモデル化とともに，この一部の実現による実験を行なった。モデル化では，特に，ユーザの問合せのうち，検索に対する処理技術の確立を目指し，2)及び3)の成果を得た。2)では，従来の１：１通信に基づいたアルゴリズムに対して，新たに１：Ｎ通信を用いたアルゴリズムの提案と，そのシミュレーションによって従来のアルゴリズムとの比較を行なった。この比較評価によって，我々のアルゴリズムはその簡単さと，パフォーマンス的有効さとの２点を備えていることを明らかに出来た。

3)では，ＤＤＢＳの共通リレーショナルインタフェースとしてのローカルデータベースプロセッサ（ＬＤＰ）（又は，ローカル情報システム）のモデル化と，その一部の実現を行なった。実現したシステムＬＤＰ－Ｖ1.5は次の制約のもとで，CODASYL 型のＡＩＭ DBS をリレーショナルモデルのスキーマ（ＬＣＳ）と言語ＱＵＥＬとによって，実際にアクセスすることが出来るもの

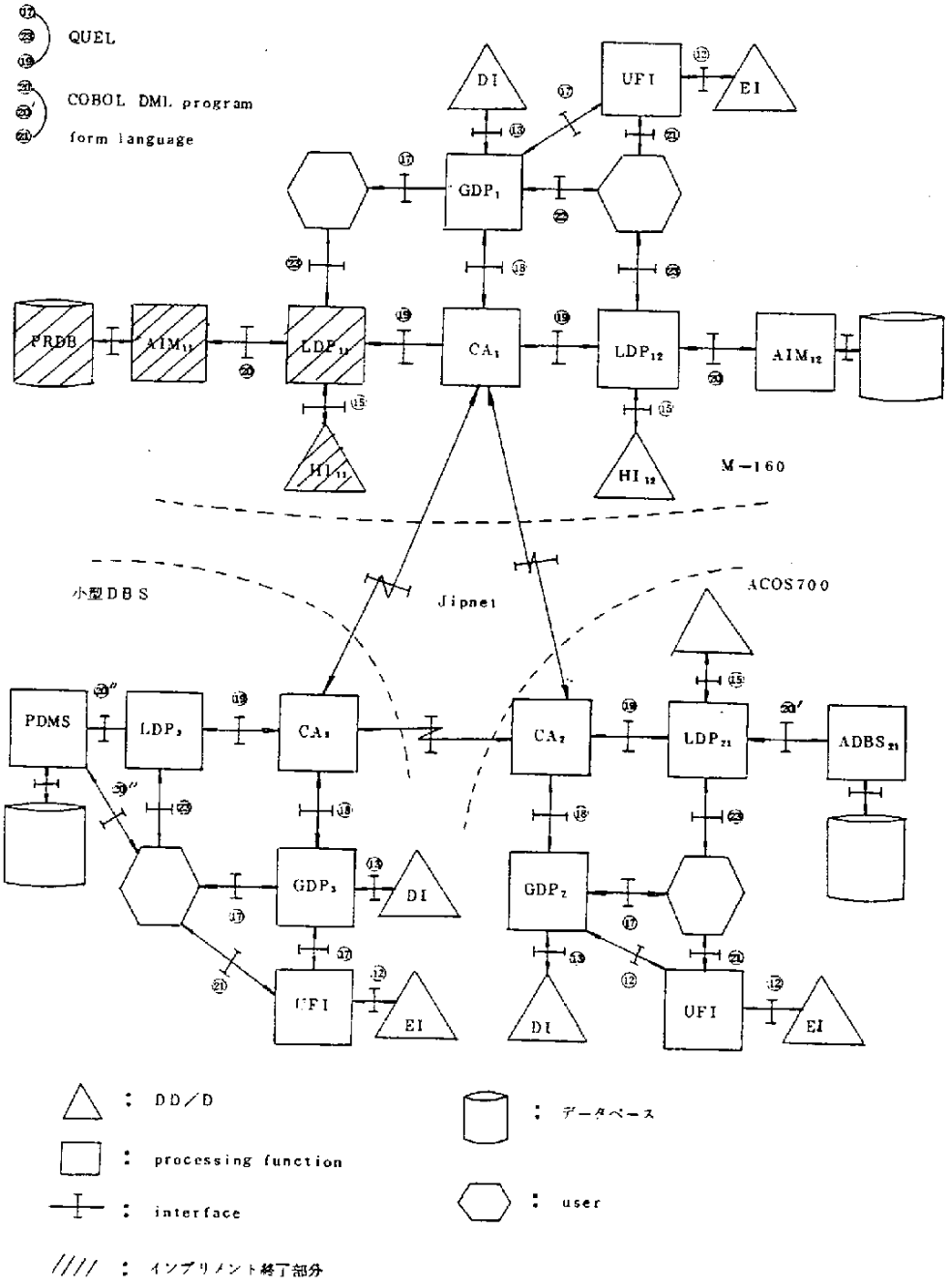


図8.1　JDDBS＝Ⅱ アーキテクチャ〔図4.6及びJDDBS 80〕

である。

i）結合としては主キーの等価結合のみが許される。

ii） aggregate 関数をふくまない。

iii）条件式は 7.1 に示した様な積正現形をしている。

このシステムは更にＬＣＳに対してビューを階層的に定義でき，これに対するアクセスをＬＣＳ問合せに変換して，実行できる。この様にＬＤＰは，ＤＤＢＳの共通インタフェースであるとともにエンドユーザに対する高水準インタフェースでもある。CODASYL スキーマから，リレーショナルスキーマの生成は，ＬＣＳ生成（同種化）において重要である。このため，第２章では，リレーショナルモデルと，CODASYL モデルの整理を行なった。

今後のデータベース利用において，特に小型データベースシステムの普及によって，従来のＤＢＳの様に，情報システム全体を一つの概念スキーマのもとで全ての統合化することによっては管理できなくなる。例えば，データベースから算出されたスナップショットが，他のアプリケーションによって共有され，データベースとなっていく時，このスナップショットと，もとのデータとの間の矛盾性をある程度認めていくことが必要になる。我々は，このためのモデルとして，フォームフローモデルを提案した。このモデルは，次の３つの基本概念に基づいている。

1） federated DBS

2） フォームの共有と非共有

3） フォームの転送

これによって，従来，不明確であったＯＩＳ等の新たな計算機アプリケーションとデータベースシステムとの関連を明確にし得たと考えている。

今後の課題としては以下の点がある。

1） フォームフローモデルの確立

2） フォームフローモデルと，ＪDDBS－Ⅱとの関連づけ

3） 更新処理アルゴリズムの確立と実現

－ ＤＭ間通信処理－ビュー更新，同時実行制御

ＤＭ内通信処理－モデル／言語変換

－ オブジェクトモデルとデータ抽象化技法の適用

参照文献

〔ABRIJ74〕

Abrial, J. R., "Data Semantics." Proc. IFIP TC-2 Working Conf. on DBMS, Cargese, Corsia, France, April 1974, pp.1-60.

〔ADIBM78〕

Adiba, M. and Euzet, C., "A Distributed Data Base System Using Logical Relational Machines," Proc. of 4 th VLDB, Berlin, Sept.1978, pp.450-461.

〔ADIBM80〕

Adiba, M., Andrade, J. M., Fernandez, F. and Toan, N. G., "An Overview of the Polypheme Distributed. Database Management System," Proc. of the IFIP'80, Tokyo-Melbourne, Oct.1980, pp.475-479.

〔ANSIX75〕

"Interim Report of the Study Group on Data Management Systems," ANSI/X3/SPARC DBMS Study Group, Report 75-02-08, Feb.1975.

〔BAUMS80〕

Bauman, L.S. and Coop, R.D., "Automated Workflow Control : A Key to Office Productivity." AFIPS Conf. Proc., 1980, pp.549-554.

〔BERNP79b〕

Bernstein, P.A., and Goodman, N., "Full Reducers for Relational Queries, Using Multi-Attribute Semi-Joins," Proc. of the IEEE Computer Networking Symposium, Gaithersburg, Maryland, Dec.1979, pp.206-215.

〔CARDA79〕

Cardenas, A.F., "Database Management Systems," Allyn and Bacon, Inc., Boston, MA., 1979.

〔CHANC73〕

Chang, C.L., Lee, R.C.T., "Symbolic Logic and Mechanical Theorem Proving," Acadmic Press Inc. 1973.

〔CODAS73〕

CODASYL, "CODASYL Data Description Language Journal of Development," June 1973, NBS Handbook 113(1974).

〔CODAS77〕

"STORED-DATA DESCRIPTION AND DATA TRANSLATION: A MODEL AND LANGUAGE" Information Systems, Vol.2, No.3, 1977.

〔CODAS78b〕

CODASYL DDL Committee, "Report of the CODASYL Data Description Language Committee," Information Systems, Vol.3, No.4, 1978, pp. 247−320.

〔CODDE70〕

Codd, E.F., "A Relational Model of Data for Large Shared Data Bank," Communications of the ACM, Vol.13, No.6, June 1970, pp.337−387.

〔COOKC80〕

Cook, C.L., "Streamlining Office Procedures-An Analysis Using the Information Control Net," AFIPS Conf. Proc., 1980, pp. 555 − 565.

〔DATEC77〕

Date, C.J., "An Introduction to Data Base System," (Second Edition), Addison-Wesley, June 1977.

〔ELLIC80〕

Ellice, C.A., and Nutt, G.I., "Office Information Systems and Computer Service," ACM Computing Survey, Vol.12, No.1, Nov.1980, pp.27−60.

〔HAMMM80〕

Hammar, M. and Mcleod. D., "On Database Management System Architecture," Infotech State of the Art, 1980, pp.177−201.

〔HEVNA78a〕

Hevner, A. and Yao, S.B., "Query Processing on a Distibuted Database" Proc. of the 3rd Berkeley Workshop, San Francisco, Aug.1978, pp.91−107.

〔HEVNA78b〕

Hevner, A. and Yao, S.B., "Optimization of Data Access in Distributed Systems," TR281, Computer Seience Dept., Purdue Univ. July 1978.

〔LADDI80〕

Ladd, I. and Tsichritzis, D., "An Office Form Flow Model," AFIPS Conf. Proc., 1980, pp.535−535.

〔LEBIJ80〕

Le Bihan, J., et al., "SIRIUS : A French Nationwide Project on Distributed Data Bases." Proc.of the 6 th VLDB, Oct.1980.

〔MOORR79〕

Moore, R.C., "Handling Complex Queries in a Distriduted Database," Technical Note 70, SRI International, Menlo Park, Oct.1979.

〔NAFFN80〕
Naffah, N., "KAYAK: A National Project for Office Automation," Proc. of the IKD, Berlin, Oct.1980.

〔ROTHJ77〕
Rothnie, J.B. and Goodman, N., "An Overview of Preliminary Design of SDD-1," Proc. of the 2nd Berkely Workshop, May 1977, pp. 39-57.

〔ROTHJ80〕
Rothnie, J.B, et al., "Introduction to a System for Distributed Databases(SDD-1)," ACM TODS, Vol.5, No.1, Mar.1980, pp.1-17.

〔SAGAD77〕
Sagalowicz, D., "IDA: An Intelligent Data Access Program," Proc. of the 3rd VLDB, Tokyo, Oct.1977, pp 293-302.

〔SENKM73〕
Senko, M.E., Altman, E.B., Astrahan, M.M. and Fehedr, P.L., "Data Structures and Accessing in Data Base Systems," IBM Systems Journal, Vol.12, No.1, 1973, pp.30-93.

〔STONM76〕
Stonebraker, M., et al., "The Design and Implementetion of INGRES," ACM TODS, Vol.1, No.3, Sept.1976, pp.189-222.

〔STONM79〕
Stonebraker, M., "MUFFIN: A Distributed Data Base Machine," Proc. of the 1st Distributed Computing System, Hantsville, Oct.1979, pp.459-469.

〔TABAK80〕
Tabata, K., "通信処理モデル," 京大情報処理教育センタ 1980.

〔TAKIM78〕
Takizawa, M., et al., "Resource Integration and Data Sharing or Hetengeneous Resource Sharing," Proc. ICCC '78, Kyoto, Sept.1978, pp.253-258.

〔TAKIM79〕
Takizawa, M. and Hamanaka, E., "The Four-Scheme Structure Concept as the Gross Architecture of Distibuted Database and Heterogeneity Problems," JIP of IPSJ, Vol.2, No.3, Nov.1979, pp.134-142.

〔TAKIM80a〕

Takizawa, M. and Hamanaka, E., "Qucry Translation in Distributed Databases," Proc. of the IFIP'80, Tokyo－Melbourne,. Oct.1980. pp.451－456.

〔TAKIM80b〕

Takizawa. M., "Distributed Problems in Distiibuted － Integration and Query Deccmpostition, " in Prepareton .

〔TANAY79A〕

Tanaka, Y., "Logical Design of Relational Schema and Integrity of a Data Base," Proc.of the IFIP TC－2 Working Conf. on Data Base Architecture, Venice, June 1979, pp.1－20.

〔TANAY79C〕

田中, 津田 "データベースシステム概論" 北大大型計算センタ, 1979.

〔TANAY80〕

Tanaka, Y. and Takizawa, M., "Object Model and Data Abstraction," in Preparetion.

〔TANAY81〕

Tanaka, Y. and Takizawa, M., "オフィス情報処理のためのフォームフローシステム" 京大数理解析研, Feb 1981.

〔WAHB80〕

Wah, B.W. and Yao, S.B., "DIALOG－A Distributed Processor Organization for Database Machine," AFIPS Conf.Proc., Vol.49, Anaheim, May 1980, pp.243－253.

〔WONGE79〕

Wong, E.,Katz, R.H., "An Access Path Model for Physical Database Design," Proc. of ACM－SIGMOD 1980 International Conf. on Management of Data, Santa Monica, CA., May 1980.

〔YAOS77B〕

Yao, S.B., "An Attribute Based Model for Database Access Cost Analysis," ACM TODS, Vol.2, No.1, March 1977, pp.45－67.

〔YAOS80〕

Yao, S.B., "The Design and Implementetion of a High Performance Databa se Machine," Dept. of Computer Sciene, Purdue Univ., 1980.

〔ZLOOM80〕

Zloof, M., "A Language for Office and Business Automation," Proc. of the OA Conf., Atlanta, Geojia, Mar.1980.

〔JDDBS80〕

"分散型リソース処理技術の研究開発 — 分散型データベースシステム," 54−S−001, (財) 日本情報処理開発協会, Mar.1980.

付記 I

データベースとデータ抽象化

— オブジェクトモデルアプローチ

Dec. 1980

田中　譲*　　滝沢　誠**

* 北海道大学工学部

** (財) 日本情報処理開発協会 開発部

目　　次

1. 序　　論

データベースシステムにおいて，ユーザの要求の多様性にどの様に適応していくかが重要な問題となってくる。多様性に対する1つの解は，リレーショナルモデルにおけるビューの定義である。ビューは，多様なユーザアプリケーションに対して，各々に必要かつ適当なデータを記述したものである。

今後のデータベースシステムの利用のもう1つの特徴は，データベースに対する更新の増加である。従来のデータベースシステム利用は，ほとんど静的な検索が中心であったが，今後，オフィス情報システム（ＯＩＳ）を中心に，更新の占める割合は重要となってくるものと思われる。

これらのことより，ビューを通した更新問題が重要となってくる。リレーショナルモデルに対するビュー定義と，その問題が現在種々指摘されている。ビューに対する検索は，問合せ変形（query modification）手法によって，ベースリレーションに対する検索に変換できる。しかし，ビューに対する更新を，自動的にベースリレーションに変換することは，ごく限定された場合においてのみ可能である。

この更新問題は，次の2点に依っているものと思われる。

1）ビューリレーションをベースリレーションと同じレベルのリレーションと考えていて，ベースリレーションに対するオペレーションを，何等の制限することなく，そのまま使わせている。

2）リレーショナルモデルは，実世界のデータの意味とは独立である。このため更新が，

ⅰ）実世界のオブジェクト自体の生成，消滅なのか。

ⅱ）オブジェクトの特徴の変化なのかを明確に区別できない。

この問題解決のための我々のアプローチは，次の2点である。

1）ビューがユーザを定義する時，ユーザは既にこのユーザの使い方を想定している。従って，ビューリレーションに対して，ベースリレーションに対するのと同じオペレーションの無制限な適用は，許されないはずである。

このことより，プログラミング言語におけるデータ抽象化手法を用いることにする。

即ち，

a）ビューリレーションをベースリレーションにより定義する。

b）ビューリレーションに対して許されるオペレーションを定義する。

c）ビューオペレーションを，ベースオペレーションに対するオペレーションによって記述する。

2）リレーショナルモデルに，オブジェクトの概念を導入する。

これによって

オブジェクト自体の生成，消滅と，

オブジェクトの特徴の変化との明確な分離を行なう。

本論文では，我々の提案するオブジェクトモデルについて2章で論じ，このモデルの操作言語について3章で論じる。4章では，オブジェクトモデルに基づいたビュー定義について述べる。

2. オブジェクトモデル

本章では，データベースシステムにおける更新問題（ビュー更新問題も含める）を解決するためのオブジェクトモデルについて論じる。

まず2.1では，オブジェクトモデルの思想と概要を述べる。ついで2.2では，オブジェクトモデルの数学的定義を行なう。

2.1 オブジェクトモデルの思想と概要

オブジェクトモデルは，リレーショナルモデルに対して，オブジェクトの概念を導入したものである。オブジェクトとは，データベースとして我々がモデル化しようとする実世界の実体である。例えば，ある会社においては，個々の従業員各部，各プロジェクトといったものがオブジェクトとなる。

データベースとは，これらのオブジェクトとオブジェクト間の関係性とを，データの値として表現したものである。従って，データベースに対する検索とは，オブジェクトと／又は，オブジェクト間の関係性を表現しているデータの値（オブジェクトの属性の値）を見つけることである。

一方，データベースに対する更新とは，検索に対して，次の２点の特徴がある。

1）オブジェクト自体及び関係性自体の生成（creation）と消滅（annihilation）

2）オブジェクトを表わす値（i.e. 属性値）の変化

リレーショナルモデルでは，この２点を区別することなく，単にリレーションと呼ばれる値の集合に対して，組単位の挿入（append），消去（deletion），属性の値の変化（replace）を行なうだけである。消去する組の意味，書き換える値の意味とは無関係な形式的なオペレーションを定義しているだけである。

我々は，リレーショナルモデルにおけるビューをふくめた更新問題は，この更新の２つの側面を明確に分離出来ないことによっていると考えている。このため我々は，次の様なオブジェクトの概念を，リレーショナルモデルの枠組をこわすことなく導入した。

オブジェクトとは先に述べた様に，モデル化しようとする実世界における実体である。例として，ある１人の女性（例に"山口百恵"としよう）を考えてみよう。この女性は１９５８年に誕生し，高校を出て就職し，１９８０年結婚し，出産し，死亡していくとしよう。彼女のライフサイクルを通してこの女性を特徴づける属性，例えば，姓名，年齢，学籍番号，住所，学校，友達，給料の値は変わっていく。しかしこの女性は誕生し，死亡していくまでは，不変の存在である。この様な，生成されてから消滅するまでに普遍に存在するモデル化する実体をオブジェクト（object）と呼ぶ。オブジェクトは又，他のオブジェクトとは成り得ない。例えば，山口百恵は，三浦百恵となり得ても，桜田淳子とは成り得ない。オブジェクトの特徴を以下に記す。

1）オブジェクトは，ライフサイクル（生成から消滅まで）を持ち，この中で普遍である。

2）オブジェクトは，他のオブジェクト又は，オブジェクトの集合とは成り得ない。

上記したオブジェクトの概念は，リレーショナルモデルを拡張することによって実現される。これをオブジェクトモデルと呼ぶ。

各オブジェクトは，データベースシステム内において，オブジェクト識別子（object－identifier）として代表される。各オブジェクトはこの生成から消滅まで，同一のオブジェクト識別子を持つことになる。

各オブジェクトの特徴（例えば，人間では名前，住所，出身地，勤務先，性別，年齢）と，オブジェクト間の関係性（例えば，ある人がある会社につとめている）とは，データベースシステム内では，リレーション（relation）によって表わされる。このことは，リレーションとして次の2種類があることを示している。

1）オブジェクト・リレーション　（object－relation）

2）アソシエーション・リレーション（association－relation）

オブジェクト・リレーションは，個々のオブジェクトと，これの特徴を表わすものである。一方アソシエーション・リレーションは，オブジェクト間の関連性を示すものである。

例として，ある従業員を表わす，次の様なEMPリレーションを考えてみよう。

EMP（*EMP，emp-name，age，address，salary）

これは，ある会社のオブジェクトである従業員を表わしている。emp-name，age，address，salary は，各々ある従業員オブジェクトの名前，年齢，住所，給料を表わす属性（attribute）である。EMPリレーションの各タプル（tuple）は個々の従業員オブジェクトを表わしている。*EMPは，ロール（role）と呼ばれる。

各タプル内のロールは，値として，このタプルの表わすオブジェクトのオブジェクト識別子を持つ，しかしこのロールの値（i.e. オブジェクト識別子）は，ユーザからは視ることはできない。同様に，ある会社の部オブジェクト（e.g. 開発部，総務部）を表わすDEPTリレーションは，

DEPT（*DEPT，dept－name）

となる。*DEPTは，部を表わすオブジェクト識別子がセットされる，ロールであり，dept-name は，各オブジェクト（即ち各部）の属性である。

次に，従業員と部との関連について考えてみよう。ある従業員オブジェクトは，ある部オブジェクトに属している。この関連性を表わすリレーションがアソシエーション・リレーションである。次に示すDEPT－EMPリレーションはこの例である。

DEPT－EMP（*DEPT，　*EMP）

オブジェクト・リレーションとは，次の様なスキームを持つ。

Ro（*R，　A_1，　A_2，……，A_n）

ここで，Roはリレーション名，*Rはロール，A_1，……，A_n は属性である。オブジェクト・リレーションはただ1つのロールを持ち，このロールによって識別されるオブジェクトの特徴を，属性A_1，……，A_n は属性である。オブジェクト・リレーションは，ただ1つのロールによって識別されるオブジェクトの特徴を，属性A_1，……，A_n によって表わす。

アソシエーション・リレーションは次の様なスキームを持つ。

$$R_A\ (\ {}^*R_1,\ {}^*R_2,\ \cdots\cdots,\ {}^*R_n\ \ [,\ A_1,\ A_2,\ \cdots\cdots,\ A_n\ ]\ )$$

R_A は，リレーション名 ${}^*R_1$，……，${}^*R_n$ はロール，　A_1，……，A_n は属性である。このリレーションは，オブジェクト・リレーションに対して，複数のロールを必ず持つ。このリレーションの各タプルは，ロール ${}^*R_1$，${}^*R_2$，……，${}^*R_n$ の値によって識別されるオブジェクト間の関連性と，その特徴（A_1，……，A_n）を表わしている。

オブジェクト・リレーションとアソシエーション・リレーションとの間には，データ操作に対して，次の様な制限が存在していることが解かれる。

1）消去規則　（deletion rules）

i）オブジェクト・リレーション（以降O－リレーションと記す）内のあるオブジェクトに対応するタプルが消去された時，アソシエーション・リレーション（以降A－リレーション）内のタプルのロールの値として，このオブジェクト識別子を持つものも消去せねばならない。

ii）Aリレーションのあるタプルの消去は，O－リレーションのタプルの消去を起こさない。

iii）あるオブジェクトの消去は，全リレーション内の組のなかで，ロールとしてこのオブジェクト識別子を取るもの全ての消去をもたらす。

2）挿入規則　（insertion rules）

i）A－relation に対して，ある組を挿入するためには，この組のロールの値に対応するオブジェクトが，O－リレーション内に存在せねばならない。

ii）O－リレーションへのオブジェクトタプルの挿入は，必ずどれかのO－リレーションに属さねばならない。

3）置換規則　（replace rules）

i）ロールの値の置換は出来ない。

ii）リレーションの属性の値は置換できる。

各オブジェクトは，このライフサークルを通して，種々のリレーションによって表わされる。例えば，図2.1.1を考えてみよう。山口百恵は誕生と同時にCHILDリレーションによって表わされるが，就学と同時にSTUDENTリレーション及び学校との関連性を示すSTUDENT－SCHOOLリレーションによって表わされる。又，卒業し，就職するとともにSTUDENTリレーションからは消え，EMPリレーションにはいる。成人式をむかえると，CHILDリレーションから，ADULTリレーションに移る。そして彼女の死亡と同時に，このオブジェクトは全リレーションから消滅する。

この様に，オブジェクトのある時点である環境のもとでの様相（facet）は，このオブジェクト識別子を値として持つロールの属するリレーションによって表わされる。又，オブジェクトの，この時の特徴（characteristics）は，オブジェクトの様相を表わすリレーションの対応するタプルの属性値によって表わされる。オブジェクトの様相と特徴は，オブジェクトのライフサイクルを通して変化し得る。

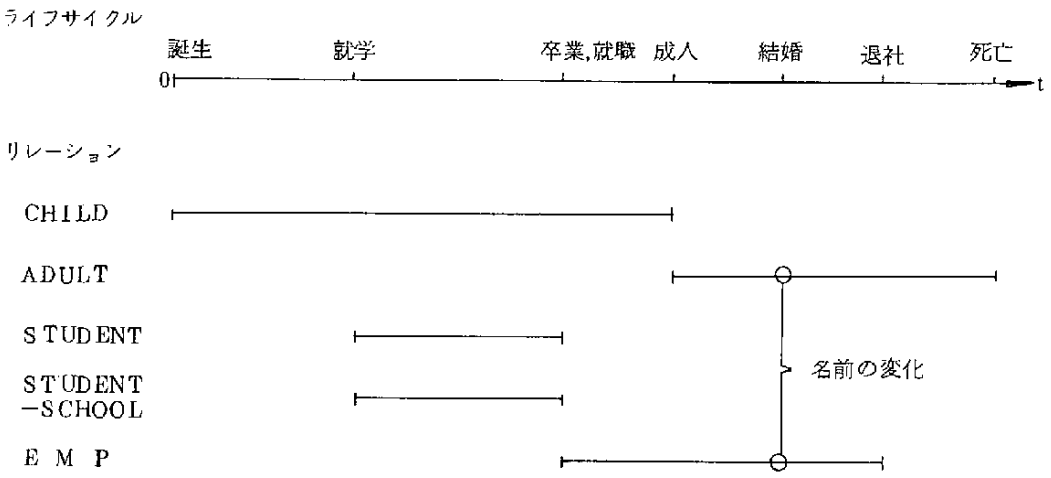


図 2.1.1　オブジェクトのライフサイクル

2.2　オブジェクトモデルの数学的定義

本節ではオブジェクト・モデルを数学的に定義する。

OとDを集合とする。Oはオブジェクトの集合，Dは値の集合と呼ぶ。Ω_0，Ω_1 を互いに排反な有限集合とする。Ω_0 をロール名の集合，Ω_1 を属性名の集合と呼ぶ。$\Omega = \Omega_0 \cup \Omega_1$ をタイプの集合と呼ぶ。各タイプ$A \in \Omega$のとり得る可能な値の集合をAの値域と呼び，Dom(A)で表わす。Ω_0，Ω_1 の各々に含まれる各タイプの値域は，

$$\forall\ A \in \Omega_0 \qquad Dom(A) = O$$
$$\forall\ A \in \Omega_1 \qquad Dom(A) = D$$

とする。

ΩからO∪Dへの関数μで，μの$X \subset \Omega$ への制限を$\mu \mid_X$で表わすとき，

$$\mu \mid_{\Omega_0} : \Omega_0 \rightarrow O \quad (全関数)$$
$$\mu \mid_{\Omega_1} : \Omega_1 \rightarrow D$$

となるものをタイプ集合のΩ上のタプルと呼び，このようなタプルの集合を

$$O^{\Omega_0} \times D^{\Omega_1}$$

で表わす。

Rが，

$$R \subset O^{\Omega_0} \times D^{\Omega_1}$$

のとき，Rを(Ω_0，Ω_1)上の関係と呼ぶことにする。$D' = O \cup D$とするとき，RはD'^{Ω} の部分集合でもあり，値集合D′に対するΩ上の関係とみなすことができ，したがって，オブジェク

ト・モデルの関係をリレーショナル・モデルの論理的枠組の中で取り扱うことができる。

オブジェクト・モデルでは，

$$X \quad \rightarrow \quad Y \qquad (X \subset \Omega_0)$$

なる関数従属関係（ＦＤ）

$$X \quad \twoheadrightarrow \quad Y \qquad (X \subset \Omega_0)$$

なる多値従属関係（ＭＶＤ）

$$X \quad \twoheadrightarrow \quad Y \quad \text{in} \quad \Omega_0$$

なる埋め込み多値従属関係（ＥＭＶＤ）の３つの型の従属関係のみを考察の対象とする。

オブジェクト・モデルにおいて，従属関係に関して正規な形をした関係を以下のように定義する。

定義 2.2.1

$R(\Omega_0, \Omega_1)$ が正規形であるというのは，以下の条件のいずれかが成り立っていることをいう。

(1) $\neg\ \exists X_0 \subsetneq \Omega_0 \qquad \exists X_1 \subseteq \Omega_1$

$$X_0 \rightarrow X_1 \quad \underline{\text{or}} \quad X_0 \twoheadrightarrow X_1$$

(2) $(\Omega_1 = \phi) \wedge (R(\Omega_0)$ がBernsteinの第３正規形である。）

以下では，$A \in \Omega$に対して，$A \in \Omega_0$ のときは，これを明示するために $^*A$と書くことにする。

次に例を用いて，正規形の方法を示すことにする。

例題 2.2.1

$$R\ (^*A,\ ^*B,\ ^*C,\ D,\ E)$$

において，

$$^*A \quad \rightarrow \quad D$$

のとき，これを次のような分解によって正規化することができる。

$$R\ [\ ^*A,\ ^*B,\ ^*C,\ E\ ]$$
$$R\ [\ ^*A,\ D\ ]$$

この分解の意味は，Ｄがロール $^*A$，$^*B$，$^*C$ の間の関連性（association）に対する属性ではなく，$^*A$のみに対する属性であることを明確にし，ロール $^*A$ 自身に関する情報を他の情報から独立化させていることである。

さらに

$$^*A\ ^*B \quad \rightarrow \quad ^*C\ E$$

なる従属関係が成立すると，$R\ [^*A, ^*B, ^*C, E\ ]$は，

$$R\ [\ ^*A, ^*B, ^*C\ ]$$
$$R\ [\ ^*A, ^*B,\ E\ ]$$

と分解することにより正規化される。この分解の意味は，Ｅが，$^*A$と $^*B$のみの関連性に対する属性であり，$^*A$, $^*B$, $^*C$すべてを含むこれらの間の関連性に対する属性ではないことを明示することと，$^*A$, $^*B$の間の関連性と $^*A$, $^*B$, $^*C$の間の関連性とを，明確に区別することであ

る。

R（Ω_0, Ω_1）が正規形のとき，Ω_0 が単一のロール $^*A$ のみからなるときRを $^*A$ に対するオブジェクト・リレーション（O-リレーション）と呼ぶ。正規形の R（Ω_0, Ω_1）において，Ω_0 の要素数が1でなく $\Omega_0=\{^*A_1, ^*A_2, \cdots\cdots, \}^*A_n$ のとき，Rを $^*A_1, ^*A_2$ ……，$^*A_n$ の間の関連性を表わすアソシエーション・リレーション（A-リレーション）と呼ぶ。

例題 2.2.2

Ω_0, Ω_1 が次のように与えられているとする。

$$\Omega_0 = \{^*\mathrm{Project}, ^*\mathrm{Department}, ^*\mathrm{Employee}, ^*\mathrm{Project\text{-}Manager}, ^*\mathrm{Department\text{-}Manager}\}$$
$$(^*\mathrm{Proj}) \quad (^*\mathrm{Dept}) \quad (^*\mathrm{Emp}) \quad (^*\mathrm{PMng}) \quad (^*\mathrm{DMng})$$

$$\Omega_1 = \{\mathrm{Project\text{-}name}, \mathrm{budget}, \mathrm{start\text{-}year}, \mathrm{end\text{-}year}, \mathrm{department\text{-}name}, \mathrm{employee\text{-}name}, \mathrm{sex}, \mathrm{birth\text{-}year}, \mathrm{marigestatus}, \mathrm{salary}, \mathrm{keyword}\}$$
$$(\mathrm{pnam}) \quad (\mathrm{budy}) \quad (\mathrm{sy}) \quad (\mathrm{ey}) \quad (\mathrm{dnam}) \quad (\mathrm{enam}) \quad (\mathrm{birth}) \quad (\mathrm{mst}) \quad (\mathrm{sal}) \quad (\mathrm{key})$$

このとき，以下のような従属関係が存在する。

$^*\mathrm{Proj} \twoheadrightarrow \mathrm{key}$

$^*\mathrm{Proj} \rightarrow \mathrm{pnam}, \mathrm{budg}, \mathrm{sy}, \mathrm{ey}, ^*\mathrm{PMng}$

$^*\mathrm{Dept} \rightarrow ^*\mathrm{DMng}, \mathrm{dnam}$

$^*\mathrm{Enp} \twoheadrightarrow ^*\mathrm{Proj}, ^*\mathrm{PMng} \quad \mathrm{in} \quad \Omega_0$

$^*\mathrm{Emp} \twoheadrightarrow ^*\mathrm{Dept}, ^*\mathrm{DMng} \quad \mathrm{in} \quad \Omega_0$

$^*\mathrm{Emp} \twoheadrightarrow \mathrm{enam}, \mathrm{sex}, \mathrm{birth}, \mathrm{mst}, \mathrm{sal}$

これらを用いてR（Ω_0, Ω_1）を正規形に分解する。

A. $^*\mathrm{Proj} \twoheadrightarrow \mathrm{key}$ を用いる。

R〔$^*\underline{\mathrm{Proj}}$, key〕 (1)

R〔$\Omega - \{\mathrm{key}\}$〕 (2)

B. $^*\mathrm{Proj} \rightarrow \mathrm{pnam}, \mathrm{budg}, \mathrm{sy}, \mathrm{ey}, ^*\mathrm{PMng}$ を用いる。

(2)⟶R〔$^*\underline{\mathrm{Proj}}$, pnam, budg, sy, ey〕 (2.1)

R〔$^*\underline{\mathrm{Proj}}$, *PMng〕 (2.2)

R〔*Proj, *Dept, *Emp, DMng, dnam, enam, sex, birth, mst, sel〕 (2.3)

C. $^*\mathrm{Dept} \rightarrow \mathrm{dnam}, ^*\mathrm{DMng}$ を用いる。

(2.3)→R〔*Dept, dnam〕 (2.3.1)

R〔*Dept, *DMng〕 (2.3.2)

R〔*Proj, *Dept, *Emp, enam, sex, birth, mst, sal〕 (2.3.3)

D. *Emp → cnam, sex, birth, mst, sal を用いる。

(2.3.3)→R〔*Emp, enam, sex, birth, mst, sal〕 (2.3.3.1)

R〔*Emp, *Proj, *Dept〕 (2.3.3.2)

E. *Emp →→ *Proj in Ω_0

(2.3.3.2)→R〔*Emp, *Proj〕 (2.3.3.2.1)

R〔*Emp, *Dept〕 (2.3.3.2.2)

この正規化結果に適当な関係名を付けると次のようなスキーマが得られる。

P−PROFILE	(*Proj, key)
PROJ	(*Proj, pnam, budg, sy, ey)
P−MNG	(*Proj, *PMng)
DEPT	(*Dept, dnam)
D−MNG	(*Dept, *DMng)
P−ASSIGN	(*Emp, *Proj)
AFFILIATION	(*Emp, *Dept)
EMP	(*Emp, enam, sex, birth, mst, sal)

図 2.2.1 オブジェクト・モデルのスキーマ

3. オブジェクト言語

本章では，オブジェクトモデルに対するデータ操作言語について論じる。

3.1節では，オブジェクトモデルにおける操作言語の特色について述べる。3.2節では，述語論理に基づいた論理的データ操作言語について述べる。3.3節では，関係代数に基づいたデータ操作言語について論じる。

3.1 オブジェクト言語の特色

本節では，オブジェクトモデルに基づいたデータ操作言語の特徴について論じる。

オブジェクトモデルでは，リレーショナルモデルと異なり，オブジェクトの概念が導入されている。リレーションに対する演算に加えて，オブジェクトの生成と消滅を行なえなければならない。オブジェクトの生成とは，システムがデータベースシステム内でユニークなオブジェクト識別子を生成し，これを用いて必要な，リレーション(object relation)のタプルを生成することである。図2.2.1の例では"新人の入社""新プロジェクトの発足"がこれに相当する。

オブジェクトの消滅とは，あるオブジェクト識別子をロールの値として持つ全てのタプルをデータベース内から消去することである。図2.2.1の例では，ある職員の退社，あるプロジェクトの終了等がこれにあたる。前者では，EMP，AFFILIATION，P-ASSING（この人が部又は／とプロジェクトの管理者であればD-MNG又は／とP-MNG）リレーションから，あるオブジェクト識別子を持つ全てのタプルを消去しなければならない。

リレーションに対するデータ操作言語としては，次の2種類がある。

1) 論理的データ操作言語
2) 代数的データ操作言語

1)は，述語論理に基づいた言語であり，ALPHA，QUELが著名である。

2)は，関係代数に基づいている。

オブジェクトにおいても，原則的にこれら両方の言語を適用できるが，次の点が異なっている。

i) ロールの値を検索し，結果リレーションに格納できるが，ユーザは視ることができない。即ち，プリントしてもロールの値は出力されない。

ii) ロールの値を更新できない。ロールの値はオブジェクト識別子であるので，これを変えることはできない。即ち，オブジェクトの属性は変えることはできても，オブジェクト自身を変えることはできない。

iii) アグリゲート(aggregate)関数の引数としてロールを用いれるのは，COUNTだけである。オブジェクトの個数は数えられるが，その平均を取る，和をとるといった演算は無意味である。

iv) ロールについての結合として許されるのは，等価結合(=)，非等価結合(≒)だけである。iii)と同じく，オブジェクト識別子が等しいとか等しくないとかの比較は意味があるが，大小の比較は無意味である。

以降1）と2）の2つのデータ操作言語について論じる。

3．2　論理的データ操作言語

本節では，オブジェクトモデルに対する論理的データ操作言語について述べる。

データ操作言語としては，次の2つを考えねばならない。

1）オブジェクトの生成と消滅

2）各リレーションに対する検索と更新（挿入，消去，置換）

3.2.1　オブジェクトの生成と消滅

オブジェクトを生成し，消滅するコマンドについて考える。

A　オブジェクトの生成

オブジェクトは次のcreateコマンドによって生成される。

create　＜object－var＞；　（3.2.1）

このコマンドによって＜object－var＞に，システムによって自動生成されたオブジェクト識別子の値がセットされる。

例えば，ある従業員“太郎”オブジェクトを新たに生成し，これについての情報をリレーションに格納するためには，次の様にする。

create　TARO ；

変数TAROには，“太郎”についてのオブジェクト識別子がセットされる。この値は，データベース内でユニークである。

B　オブジェクトの消滅

オブジェクトはdestroyコマンドによって消滅される。

destroy＜v-role＞〔where ＜qual＞〕；　（3.2.2）

＜qual＞は，述語論理形式の条件式〔3.2.2を参照〕である。＜qual＞を満足するオブジェクト識別子を持つオブジェクトを消滅させる。

例　　開発部の従業員を解雇する。

```
range   ( d, DEPT) ( a,  AFFILIATION )
        ( e, EMP);
destroy  e.*EMP
where    e.*EMP   = a.*EMP        and
         e.*DEPT  = d.*DEPT       and
         d.dnam   = "DEVELOPMENT";
```

3.2.2　リレーションに対するオペレーション

リレーションに対するオペレーションとしては，検索，更新（消去，挿入，置換）がある。

これはリレーショナルモデルにおけるものと，次の点以外同じである。

1) ロールの値を検索することはできるが，視ることはできない（プリントできない。）

2) ロールの値を更新（置換）できない。

3) アグリゲート（aggregate）関数の中で，引数としてロールが使えるのはcount関数だけである。

4) ロールについての結合述語としては，等価結合（equi-join）と非等価結合（≠）だけである。 ${}^{*}R_1 \overset{*}{=} R_2$ 又は ${}^{*}R_1 \overset{*}{\neq} {}^{*}R_2$ は許されるが，${}^{*}R_1 > {}^{*}R_2$，${}^{*}R_1 \geq {}^{*}R_2$ は許されない。

以降QUELをオブジェクト・モデル言語として考えてみる。QUELは次の5つのコマンド文からなっている。

a) range文

b) retrieve文

c) append文

d) delete文

e) replace文

以降，この5つについて述べる。これらの言語は，単にリレーションを操作する基本的な言語（即ち，機械語）であることを注意しておく。

A. range 文

range文は次の形式を持ち，データ操作に用いるリレーションに対して，そのタブル変数を定義するものである。

range（＜var＞,＜rel-name≫){(＜var＞,＜rel-name≫)}； （3.2.3）

（＜var＞,＜rel-name＞）の対は，＜rel-name＞に対して，タブル変数＜var＞を定義する。

例えば，図2.2.1の例について考えよう。PROJ リレーションに対して，タブル変数pを定義するには次の様にする。

range （p, PROJ）；

B. retrieve文

次に検索文について考えてみる。これは，次の様な形式をもつ。

retrieve〔into ＜rel-name＞〕(＜target-list＞)
〔where ＜qual＞〕； （3.2.4）

＜target-list＞::＝＜t-clause＞{＜t-clause＞}

＜t-clause＞::＝＜attribute＞=＜a-exp＞|＜role＞=＜n-role＞

＜a-exp＞::＝＜constant＞|＜v-attribute＞|
≪constant＞と＜v-attribute＞上に対する算術式＞|
≪constant＞と＜v-attribute＞上に対する関数＞

＜qual＞::＝＜c-pred＞|(＜qual＞)|＜qual＞ and ＜qual＞|
＜qual＞or ＜qual＞|
not ＜qual＞

＜c-pred＞::＝＜a-exp＞＜cop＞＜a-exp＞|＜v-role＞＝＜v-role＞|
＜v-role＞≒＜v-role＞

＜cop＞::＝＜|≦|＝|≧|＞|≒

＜v-attribute＞::＝＜var＞.＜attribute＞

＜v-role＞::＝＜var＞.＜role＞

次に例を考えてみよう。

例3.2.1） 開発部の部長を求めよ。

```
range ( e, EMP ) ( d, DEPT ) ( dm, D-MNG );
retrieve ( e, enam )
where d. dnam = "DEVELOPMENT" and
      d.*DEPT = dm.*DEPT and dm.*DMng = e.*EMP ;
```

例3.2.2）プロジェクトのメンバが1つの部だけに属しているプロジェクトと部名を求めよ。

```
range ( pa, P-ASSIGN)( a, AFFILIATION )
      ( p, PROJ );
retrieve( p. pnam, d. dnam )
where
    count (a .*DEPT by pa.*PROJ
    where pa.*PROJ = p.*PROJ and pa.*EMP = a.*EMP) = 1 ;
```

例3.2.3） 全ての部からメンバが参加しているプロジェクト名とそのリーダを求めよ
（その結合リレーション名をRRとせよ）

```
range ( pm . P--MNG) ( m, EMP ) ;
retrieve into RR ( p-pnam, m. enam )
where
    count ( d.*DEPT ) = count ( a.*DEPT by pa.*PROJ
                        where pa.*PROJ = p.*PROJ and
                              pa.*EMP = a.*EMP)
and pa.*PROJ = pa.*PROJ and
    p.*PROJ = p.*PROJ and pm.*PMng = m .*ENG;
```

C. append 文

ある組／組の集合をリレーションに挿入するためには，次のようなappend文を用いる

```
append 〔 to 〕＜rel-name＞(＜target-list＞)
       〔 where ＜qual＞〕;                    ( 3.2.5 )
```

＜target-list＞は，＜rel-name＞に付加される属性値を記述する。＜target-list＞内に現われない属性の値は，null値がsetされる。

次に例について考える。

例3.2.5） 新人“日本太郎”（男，1960年生まれ，独身，給料は未定）を入社させる。

```
range ( e, EMP );
create  TARO ;
append to EMP (*EMP = TARO ,  enam = "NIHON  TARO",
                        birth = 1960 ,  mst ="s")
```

これによって，タプル（TARO，“NIHON-TARO”，1960，S，ⓐ）がEMPリレーションに挿入される。

例3.2.6） “日本太郎”を開発部に配属する。

```
range ( d, DEPT ) ;
append to AFFILIATION (*EMP=ⓐe.*EMP , *DEPT =ⓐd.*DEPT )
where  e.enam = "NIHON-TARO"  and
       d.dnam = "DEVELOPMENT"  ;
```

ⓐe.*EMPは，e.enam“NIHON TARO”のタプルがユニークでなければならないことを示している。もし，複数の日本太郎がいれば，システムはユーザに他の属性値（e.g. sex,birth）の入力をもとめる。ⓐd.*DEPTも同じである。

D delete文

delete文は，次の形式によって，リレーションからタプルの除去を行なう。

```
delete ＜var＞ 〔 where  ＜qual＞〕 ;                    ( 3.2.6 )
```

＜var＞は，deleteすべきリレーションに対するタプル変数である。このタプルは＜qual＞の条件を満足せねばならない。

次に例について考えてみる。

例3.2.7） 開発部を廃部せよ。

```
range ( d,  DEPT ) ;
delete  d  where  d. dnam = "DEVELOPMENT"  ;
```

例3.2.8） 総務部のメンバが1人でも属しているプロジェクトを廃止せよ。

```
range ( p,  PROJ )  (e,  EMP )  (pa, P-ASSIGN )
      ( d, DEPT )  (a,  AFFILIATION );
delete  p
where   d. dnam = "ADMINISTRATION"  and
```

d.*DEPT = a.*DEPT and a.*EMP = e.*EMP and
e.*EMP = pa.*EMP and pa.*PROJ = pa.*PROJ ;

E replace文

replace文は次の形式で，リレーション内の属性の値をreplaceする。

replace <var>(<target-list>)〔where<qual>〕; (3.2.7)

<qual>を満足するタプル(<var>)を，<target-list>に基づいて属性の値を書き換える。

次に例について考える。

例3.2.9) DDBS projectのメンバの給料を10％上げる。

```
range   ( p, PROJECT ) (e , EMP )
        ( pa, P-ASSIGN ) ;
replace   e ( sal = e. sal * 1.1)
where     e.*EMP = pa.*EMP and pa.*PROJ = p. *PROJ and
          p.pnam = "DDBS";
```

3. 3 代数的データ操作言語

オブジェクト集合O，値集合Dと，ロール名集合Ω_0，属性名集合Ω_1に関して定義される(Ω_0, Ω_1)上のオブジェクトモデルにおける関係は，$\Omega_0 \cup \Omega_1$を属性集合とし，O∪Dを値集合とするリレーショナルモデルにおける関係とみなすことができる。したがって，オブジェクトモデルの関係に対しても，リレーショナルモデルの関係代数において定義されるプロジェクション，リストリクション，セレクション，ジョイン，ディヴィジョンの関係演算を同様に定義することができる。

ただし，オブジェクトモデルの思想から，以下のような演算は許されない。

1. リストリクション：R〔A θ B〕 (θは比較子，$\theta \in \{<, \leq, =, \geq, >, \neq\}$)
   R(Ω_0, Ω_1)に対して，($A \in \Omega_0 \wedge B \in \Omega_1$)か($A \in \Omega_1 \wedge B \in \Omega_0$)の場合
2. セレクション：R〔A θ v〕 ($\theta \in \{<, \leq, =, \geq, >, \neq\}$)
   R(Ω_0, Ω_1)に対して，$A \in \Omega_0$の場合
   (例外として，後に述べる〔*A=(*A-reg)〕，〔*A≒(*A-reg)〕を許す。)

さらに，$A \in \Omega_0$に対する値の平均や，総和は許されないものとする。また，$A \in \Omega_0$に対する値は見ることができないものとする。

例題3.3.1

図3.3.1(a)のようなR(*A, *B, C, D)に対して，プロジェクション R〔C, D〕，R〔*A, *B, C, D〕，R〔*A, C, D〕をとった場合，見ることのできる情報は各々(b), (c), (d)のようになる。

R〔Ω_0, Ω_1〕に対して，我々の見ることのできる情報を$\vee$R(Ω_0, Ω_1)で表わすことに

する。

R	*A	*B	C	D	
	i_1	i_4	a	1	①
	i_1	i_5	a	1	②
	i_2	i_6	a	1	③
	i_2	i_4	a	2	④
	i_3	i_5	b	2	⑤
	i_3	i_7	c	3	⑥

invisible

(a)

$\vee R[C, D]$	C	D	
	a	1	①②③
(b)	a	2	④
	b	2	⑤
	c	3	⑥

$\vee R[*A, *B, C, D]$	C	D	
	a	1	①
	a	1	②
(c)	a	1	③
	a	2	④
	b	2	⑤
	c	3	⑥

$\vee R[*A, C, D]$	C	D	
	a	1	① ②
	a	1	③
(d)	a	2	④
	b	2	⑤
	c	3	⑥

図 3.3.1　プロジェクションの結果，我々が見ることのできる情報

$R(\Omega_0, \Omega_1)$, $X, Y \subset \Omega$, $A \in \Omega_1$ に対して $R[X <\text{sum}> Y; A]$ を

$$R[X <\text{sum}> Y; A] = \{\mu' \mid \mu' : X \cup (A) \to O \cup D,\ \mu' \mid_X \in R[X],\ \mu'(A) = \sum_{\mu \in R[X, Y, A]} \mu(A)\ \}$$

$$\mu \mid_X = \mu' \mid_X$$

を定義する。

例題 3.3.2

図 3.3.1 (a)の関係 R に対して,

$R[<\text{sum}> *A, *B; D] = \{10\}$

$R[<\text{sum}> *A; D] = \{9\}$

$R[<\text{sum}> *B; D] = \{10\}$

$R[<\text{sum}>; D] = \{6\}$

$R[*A <\text{sum}> *B; D]$	$*A$	D
	i_1	2
	i_2	3
	i_3	5

$R[*A <\text{sum}>; D]$	$*A$	D
	i_1	1
	i_2	3
	i_3	5

となる。

$X, Y \subset \Omega$, $X \cap Y = \phi$ に対して, $R[X <C> Y]$ を

$$R[X <C> Y] = \{\mu' \mid \mu' : X \cup \{C\} \to O \cup D,\ \mu' \mid_X \in R[X],\ \mu'(C) = |\{\mu \mid \mu \in R[X, Y],\ \mu \mid_X = \mu' \mid_X\}|\ \}$$

where

$|S|$: 集合 S の要素数 (カーディナリティ)

と定義する。

図 3.3.1 (a)の関係 R に対し,

$R[<E> *A, *B, C] = \{6\}$

$R[<E> *A, C] = \{4\}$

$R[<E> C] = \{3\}$

$R[*A <E> *B\ C]$	$*A$	E
	i_1	2
	i_2	2
	i_3	2

$R[*A <E> C]$	$*A$	E
	i_1	1
	i_2	1
	i_3	2

f を $(O \cup D)^n \to O \cup D$ ($n \geq 0$) なる関数とする。 n個の属性よりなる $X \subset \Omega$, $A \in \Omega$ に対し，$R(\Omega_0, \Omega_1)$ の変更演算 $R\langle A \leftarrow f(X)\rangle$ を

$$R\langle A \leftarrow f(X)\rangle = \{\mu' \mid \mu \in R,\ \mu' \mid \Omega - \{A\} = \mu \\ \mu'(A) = f(\mu(X))\}$$

と定義する。

例題 3.3.3　変更演算の例を示す。

R	*A	B	C	D
(a)	i_1	1	3	1
	i_2	1	5	2
	i_2	2	5	2

R〈B←6〉	*A	B	C	D
(b)	i_1	6	3	2
	i_2	6	5	2
	i_2	6	2	1

R〈B←D〉	*A	B	C	D
(c)	i_1	1	3	1
	i_2	2	5	2

R〈B←C+D〉	*A	B	C	D
(d)	i_1	4	3	1
	i_2	7	5	2

図 3.3.2　変更演算の例

$R(\Omega_0, \Omega_1)$, $S(\Omega_0, \Omega_1')$ に対し，$(\Omega_0, \Omega_1 \cup \Omega_1')$ 上の $R+S$ を

$$R+S = \{\mu \mid \mu \in R \text{ or } \mu \in S\}$$

と定義し，$R(\Omega_0, \Omega_1)$, $S(\Omega_0, \Omega_1)$, $X \subset (\Omega_0 \cup \Omega_1) \cup (\Omega_0 \cup \Omega_1)$ に対し，

$$R\langle X\rangle S = \{\mu \mid \mu \in R \text{ and } \mu \mid x \notin S\langle X\rangle\}$$

と定義する。 R の S による $X \subset \Omega_0$ に関しての置換 $R\langle X\rangle S$ を

$$R\langle X\rangle S = \{\mu \mid (\mu \in R \wedge \mu \mid x \notin S\langle X\rangle) \\ \vee (\mu \mid x \in R \wedge \mu \in S)\}$$

と定義する。

例題 3.3.4

R (*A, *B, C), S (*A, *B, C) を図 3.3.3 の(a), (b)のような関係とするとき，R＋S, $R\langle {}^*A{}^*B\rangle S$, $R\langle {}^*A\rangle S$ は各々，同図(c), (b), (c)のようになる。

R	*A	*B	C
(a)	i_1	i_2	a
	i_1	i_3	b
	i_4	i_3	c
	i_5	i_6	d

R	*A	*B	C
(b)	i_1	i_2	e
	i_1	i_5	f
	i_4	i_3	c
	i_7	i_6	a

R+S	*A	*B	C
(c)	i_1	i_2	a
	i_1	i_3	b
	i_4	i_3	c
	i_5	i_6	d
	i_1	i_2	e
	i_1	i_5	f
	i_4	i_3	c
	i_7	i_6	a

$R\langle {}^*A{}^*B\rangle S$	*A	*B	C
(d)	i_1	i_3	b
	i_5	i_6	d

$R\langle {}^*A\rangle S$	*A	*B	C
(e)	i_1	i_2	e
	i_1	i_5	f
	i_4	i_3	c
	i_5	i_6	d

図 3.3.3　付加，削除，置換の例

RとSを同じ（Ω_0, Ω_1）上の関係とし，A∈Ωに対して，R＜A＝v〕Sを

R＜A＝v〕S＝（R－〔A〕S）＋S〔A←v〕

と定義する。

（a_1/A_1, a_2/A_2, …, a_n/A_n）で$\mu(A_i)=a_i$なる関数

μ：{A_1, A_2, …, A_n}→DUOを表わす。

△を関数の集合とする。△中の各ロール*Aに対して，対応するレジスタ*A-regを考える。(*A-reg)で，*A-regの内容を表わす。*A-regは最後に検索されたロール*Aのオブジェクトである。

create objはOの要素で，現在使用されていないオブジェクト識別子を与えるものとする。

R(*A, Ω_1)を△中のO－リレーションとする。△中で各ロールのO－リレーションは唯一であるとする。ロール*Aに対する演算として，create, delete, change, mergeを以下のように定義する。

```
procedure create (*A)
    do
        *A-reg ← newobj(*A);
        R ← R + ((*A-reg)/*A)
    od ;
procedure delete(*A)
    do
        R ← R〔*A≒(*A-reg)〕;
        for each S(Ω'0, Ω'1) s.t. *A∈Ω'0
            do
                delete (S, *A)
            od
    od ;
procedure change (*A, A ← f(X))
    do
        R ← R＜*A〕R〔*A＝(*A-reg)〕〔A ← f(X)〕;
    od
procedure merge (*A, Xθv)
    do
        R ← R＜*A＝(*A-reg)〕(R〔Xθv〕);
        for each S(Ω'0, Ω'1) s.t. *A∈Ω'0
            do
                S ← S＜*A＝(*A-reg)〕(S〔*A＝*A〕R〔Xθv〕)〔Ω'0 ∪ Ω'1〕);
            od
    od ;
```

Aリレーション$R(*A_1, *A_2, \cdots, *A_n, Y)(Y \subset \Omega_1)$に対してcreate, delete, change を以下のように定義する。

procedure create (R)
  do
    for each i s.t. $1 \leq i \leq n$
      do
        create $(*A_i)$
      od ;
    $R \leftarrow R + ((*A_1\text{-reg})/*A, \cdots, (*A_n\text{-reg})/*A_n)$
  od ;

procedure delete(R, X) $(X \subset \Omega_0)$
  do
    for $*A \in X$
      do
        $R \leftarrow R[*A \fallingdotseq (*A\text{-reg})]$
      od
  od ;

procedure change (R, $\{*A_1, *A_2, \cdots, *A_n\}$, $A \leftarrow f(X)$)
  do
    $R \leftarrow R\langle *A_1, *A_2, \cdots, *A_n \rangle (R[*A_1 = (*A_1\ \text{reg})][*A_2 = *A_2 - \text{reg})]$
    $\cdots [*A_n = (*A_n\text{-reg})][A \leftarrow f(X)])$
  od ;

(Ω_0, Ω_1)上の関係Rに対して, $A \in \Omega = \Omega_0 \cup \Omega_1$, $B \in \Omega$とするとき, $R[A \backslash B]$を

$$R[A \backslash B] = \{\mu' \mid \mu' : (\Omega_0 - \{A\}) \cup \{B\} \to D \quad \exists \mu \in R\,(\mu' \mid_{\Omega_0 - \{A\}} = \mu \mid_{\Omega_0 - \{A\}}) \wedge (\mu'(B) = \mu(A))\}$$

と定義する。$[A \backslash B]$はRのタイプ名AをBに名前がえする。

X, $Y \subset \Omega$, $A \in \Omega_1$に対して$R[X < \text{avg} > Y ; A]$を

$R[X < \text{avg} > Y ; A]$

$= R[X < \text{sum} > Y ; A][X = X]R[X < C > Y])[A \leftarrow A/C]$ と定義する。

例題 3.3.5

図 3.3.1 のRに対して, $R[*A < \text{sum} > *B ; D]$, $R[*A < E > *B]$, $R[*A < \text{avg} > *B ; D]$ を図 3.3.4 に示す。

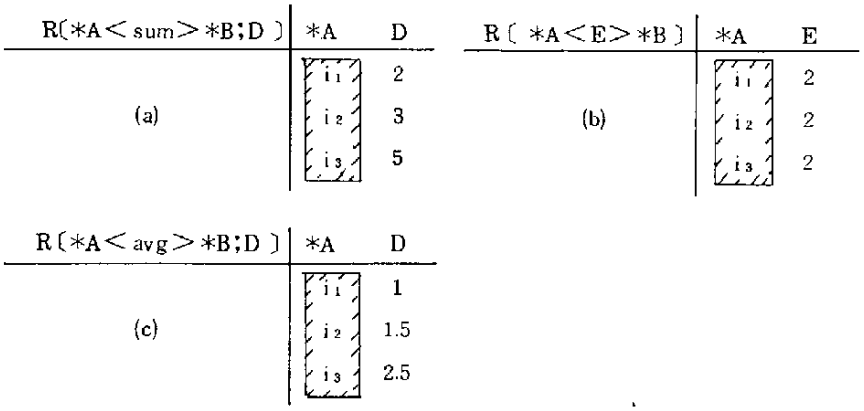


図 3.3.4　sum , count , average の例

例題　3.3.4

関係

PROJ (*Project , pname, budget , start−year , end − year)

において，1980年には終了していないプロジェクトで，開始年が1970年以前のプロジェクトの予算を80％に減額するとする。これに対する演算式は

PROJ ＜*Proj〕(PROJ〔end-year ＞ 1980〕〔start-year ≤ 1970 〕

〔budget ← budget ＊ 0.8〕)

と書ける。

4. ビュー とデータ抽象化

本章では，オブジェクトモデルとデータ抽象化手法とに基づいたビュー定義を例に即しながら論じる。

まず，4.1節では，ビューの更新問題について論じる。

4.2節では，例に基づいて論理的ビュー定義について述べる。

4.3節では，4.2節と同じ例について，代数的ビュー定義を行なう。

4. 1　ビュー更新問題

ビュー定義は，3章で2種の言語，即ち論理的データ操作言語と代数的データ操作言語によって定義できる。前者については4.2を，後者については4.3を参照されたい。ここではビューリレーションは結合と制限と射影によってつくられるものとして，null 値のある場合を考えないものとする。ビューリレーションには，ベースリレーションからつくられたユーザのデータの視方が反映されている。

問題は，ビューに対するオペレーションをこの様にベースリレーションに対するオペレーションに正しくマッピングできるかである。検索については，ビューの定義式を用いることによって自動的に変換できる。この手法の例としては，問合せ変形（ query modification ）手法がある。

しかし，ビューに対する更新要求に関しては，この様な手法によって，全ての場合正しく変換することはできない。この方法を，ビュー更新の自動変換手法と呼ぶことにする。

更新要求のベースリレーションへのマッピングに対する最近の試みは，ビューを抽象データ型とみて，これに対して許される操作（更新）オペレーションを明確に定義する方法である。定義されたビューオペレーションについては，正しく，ベースリフレーションに対する操作オペレーションの手続が定義される。したがって，ビューに対して，関連するビューオペレーションだけを行なうことできる。

我々はこの手法は，ビュー問題に対して，現実的で有効な解であると考えている。
これは，次の理由による。

1）ベースリレーションの世界とビューリレーションの世界とは異なったものである。従って，ベースリレーションに対して許されるデータ操作オペレーション（ e.g. 検索，追加，消去，置換）を無条件に与えることは出来ない。

ビューはユーザアプリケーションのデータの意味を反映したものである。このビューの使い方は，ビューの定義とともに決まっている。従って，この使い方をビューオペレーションの定義として明確に記述することは自然である。

2）ビューオペレーションをベースリレーションに対するオブジェクトの概念に基づいたオペレーションによって（手続的に）記述することによって，ビューの更新要求は正しくベースリレーションにマッピングできる。

ベースリレーションに対する更新について考えてみよう。ここでの重要な問題は，各リレーション及びリレーション間のインテグリティ条件をいかに保つかである。我々はこのために，オブジェクトの概念を導入した。リレーションをオブジェクトを表わすもの（Oリレーション）と，オブジェクト間のアソシエーションを表わすもの（Aリレーション）とに分けた。

オブジェクト自体の生成と消滅に関するリレーションの更新は，このオブジェクトに関するオブジェクトリレーションの更新とともに，関連するAリレーションに対する更新を起こす。この規則は，2.1節でのべた3つの規制（消去，挿入，置換）である。しかし，オブジェクトを表わす特徴（オブジェクトの属性の値）に対する更新は対応するタプルの属性の値だけの置換だけである。

我々は，現在のリレーショナルモデルにおける更新問題は，上記した2つの異なったオペレーションが混同していることによっていると考えている。即ち，

1）オブジェクト自体の生成・消滅，及びアソシエーション自体の生成，消滅

2）オブジェクトの特徴を表わす属性の値の変化

の2つの混同である。我々の提案するオブジェクトモデル（Oモデル）は，この2つを明確に分離することを目的としている。

4. 2　ビューの論理的定義

本節では，論理的データ操作言語を用いたビュー（ view ）の定義について，図 2.2.1 をベースリレーションとした場合を例にとって述べる。

ビューとしは，次の３つを考える。

1)　EXECUTIVE-VIEW

2)　PERSONNEL

3)　PROJECT-MANAGER-VIEW

1) は，社長クラスの人間から視た企業情報への view である。 2) は，人事部からみたビューである。 3) は，プロジェクト管理者が視たビューである。

4.2.1　EXECUTIVE ビュー

EXECUTIVE-VIEWリレーションは，図 2.2.1 に対して，社長クラスのトップマネジメントが必要とする情報のビューである。社長はこのビューを通して情報を得るとともに，部やプロジェクトの廃止，新設，統合，分割を行ない，会社のベースアップ率を決める。

A　ビューリレーションの定義

PROJ（pnam, sy, ey, p-mng , #-emp-proj）

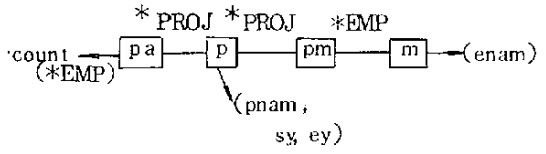
DEPT（dnam, #-emp-dept ）

ビューリレーションPROJ は，各プロジェクトの名前（ pnam），開始年（ sy ），終了年（ey）リーダー（ p-mng），メンバ数（ #-emp-proj ）を示している。

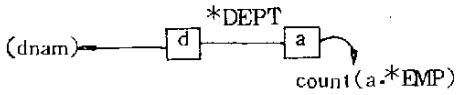
DEPT は，部名（ dnam），部員数（#-emp-dept ）を示している。

この２つのビューリレーションは，次のように定義される。

```
range   (p, PROJ)(e, EMP)(d, DEPT)(a, AFFILIATION), (pm, P-MNG)
        (dm, D-MNG)(pa, P-ASSIGN)(m, EMP);
define  PROJ
       (pnam = p.pnam , sy = p.sy , ey = p.ey , p-mng = m.enam ,
        #-emp-proj = count (pa.*EMP by p.*PROJ where p.*PROJ =
                                                    pa.*PROJ))
where   p.*PROJ = pm.*PROJ and pm.*EMP = m.*EMP and
        p.*PROJ = pa.*PROJ and pa.*EMP = e.*EMP;
```




```
define DEPT
       (dnam = d.dnam ,
        #-emp-dept = count(a.*EMP by d.*DEPT where a.*DEPT = d.*DEPT));
```



B　ビューオペレーションの定義

次にこの2つのビューリレーションに対するオペレーションについて考える。

1) abolish DEPT (dnam);

部名がdnam部を廃止する。この時部員は，どの部にも属さないようにする。

このオペレーションは，ベースリレーション〔図2.2.1〕に対するオペレーションによって，次の様に記述された。

```
retrieve into R ( dept = ⓐd. *DEPT)
        where  d. dnam = dnam ;
range ( r , R ) ;
delete  d  where  d.*DEPT = r.dept ;
delete  dm where dm.*DEPT =r.dept ;
delete  a  where  a.*DEPT = r.dept ;
delete  r ;
```

=

```
destroy (ⓐd:*DEPT)
where d. dnam
        = dnam ;
```

2) create DEPT (dnam) ;

部名が，dnamの部を更新する。オブジェクトリレーションDEPTにこのオブジェクトのタプルを挿入する。dnam以外の属性値はnull値をとる。

```
create DEPTM;
append to DEPT (*DEPT = DEPTM,
                dnam = dnam ) ;
```

3) merge DEPT (dnam 1, dnam 2, dnam) ;

2つの部　dnam 1 とdnam 2 とを統合してdnamとする。

2つの部員をdnamの部員とする。dnam 1 とdnam 2 の部長職を解く。

```
retrieve into R ( dept = d. *DEPT )
      where d.dnam = dnam 1  or  dnam = dnam 2 ;
range  ( r, R );
delete d where  d.*DEPT = r.dept ;
delete dm where dm.*DEPT = r.dept ;
create DEPTM :
append to DEPT (*DEPT = DEPTM, dnam = dnam );
replace  a  (*DEPT = DEPTM )
      where  a.*DEPT = r.dept;
```

……(＊)

```
delete r ;
```

(*)userは，ロールの値のreplaceを出来ない。しかし，viewのマッピングでは許す。これは次のものと等価である。

```
range ( ai, AFFILIATION );
append to AFFILIATION
 (*DEPT = DEPTM, *EMP=ai.*EMP)
 where ai.*DEPT = r.dept ;
 delete ai where ai.*DEPT=r.dept;
```

4) discontinue PROJ (pnam) ;

プロジェクト(pnam)を中止する。プロジェクトメンバーも同時に，このプロジェクトからはずす。

```
range ( pp, P-PROFILE );
retrieve into R (projet =@p.*PROJ ) where p.pnam = pnam ;
range ( r, R ) ;
delete p where p.*PROJ = r.projet;
delete pa where pa.*PROJ = r.projet;
delete pm where pm.*PROJ = r.projet;
delete pp where pp.*PROJ = r.projet;
delete r ;
```

5) create PROJ (pnam, sy, ey) ;

開始年(sy),終了年(ey)であるプロジェクト(pnam)を生成する。

```
create PROJCT ;
append to PROJECT (*PROJ = PROJCT , sy = sy, ey = ey )
```

6) merge PROJ (pnam 1, pnam 2, pnam) ;

プロジェクトpnam 1とpnam 2とを結合してpnamをつくる。

2つのprojectのメンバをpnamに移し，これらのprojectのリーダを解く。pnamプロジェクトのリーダはundefined。

```
retrieve into R ( projct = p.*PROJ) where p.pnam = pnam 1 or
                                           p.pnam = pnam 2 ;
range ( r, R ) ;
delate pm where p.*PROJ = r.projct ;
create PROJECT;
append to PROJ (*PROJ =PROJECT, pnam = pnam )
                budg = SUM (p.budget where p.*PROJ = r. projct ),
                sy = MIN (p.sy where p.*PROJ = r. projct ),
```

```
                ey = MAX ( p.ey  where p.*PROJ = r.projct )) ;
delete p where p.*PROJ = r.projct ;
{ replace pp (*PROJ = PROJCT )
        where pp.*PROJ = r.projct ;
  replace pa (*PROJ = PROJCT )
        where pa.*PROJ = r.projct ; } ……(*)
delete r ;
```

(*)これも次と等価である。

```
range ( pp1, P-PROFILE ) ;
append to P-PROFILE
     (*PROJ=PROJCT, keyword=pp1,
                          keyword )
   where pp1.*PROJ=r.projct;
delete pp1 where pp1.*PROJ=
                          r.projct);
range ( pc1, P-ASSIGN ) ;
append to P-ASSIGIGN
   ( *PROJ=PROJCT, *EMP=pa1.
                          *EMP)
   where pa1.*PROJ=r.projct ;
delete pa1 where pa1.*PROJ=r.projct ;
```

5) raise salary (n) ;

全従業員の給料を一律 n %昇げる。

```
replace e ( sal = e.sal* (1+n/100 ));
```

6) assign PMNG (pnam, pmng)

pnamプロジェクトに, pmngをmanagerとしてassignする。

```
retrieve into R1 ( pmngr =ⓐe.*EM )
     where e.enam = pmng ;
retrieve into R2( pname=ⓐp.*PROJ) where p.pnam=pnam ;
range ( r1, R2 ) ( r2, R2 ) ;
append to P-MNG (*PROJ=r2. pnam, *PMng = r1.pmngr ) ;
delete r1 ;
delete r2 ;
```

7) discharge (pmng)

従業員 Pmng を全てのproject leaderから降ろす。

```
retrieve into R1 ( pmgr =e。EMP)  where e.enam = pmng ;
range (r1, R1) ;
delete pm where pm.*PMng = r1.pmgr ;
delete r1;
```

4.2.2 Personnel ビュー

Personnel ビューは，人事部のビューである。人事部はこのビューを通して，職員の入社，昇進，降格，退職，結婚，転部等についての情報の更新を行なう。

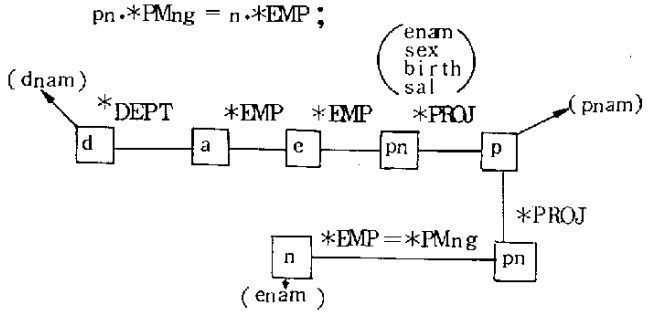
A ビューリレーションの定義

EDP-PROFILE(enam, sex , birth, salary, pnam, dnam, dmng)

このビューリレーションは，従業員の名前(enam)，性別(sex)，誕生日(birth)，給料(salary)，属する project 名(pnam)，部名(dnam)とその部長名(dmng)を表わしている。

このビューリレーションは次の様に定義される。

```
define EDP-PROFILE
      ( enam = e.enam ,  sex = e.sex,  birth = e.birth, salary = e.sal ,
      pnam = p.pnam , dnam = d.dnam , dmng = n.enam )
where
      e.*EMP = a.*EMP  and  a.*DEPT = d.*DEPT  and
      e.*EMP = pa.*PROJ  and  p.*PROJ = pn.*PROJ  and
      pn.*PMng = n.*EMP ;
```



B ビューオペレーションの定義

次に，このビューリレーションに対して許されるオペレーションについて述べる。

1) hire (enam , sex , birth);

新たに新入社員をやとうことを表わす。

EMP リレーションに登録する配属先は，総務部とする。salary は初任給が与えられる。

```
create EMPLY ;
append to EMP (*EMP = EMPLY , enam = enam , sex = sex, birth = birth,
```

```
                                        sal = FSAL (birth));
    append to AFFILIATION ( *EMP=EMPLY, *DEPT=d.*DEPT)
    where d dnam = "ADMINISTRATION ";
```

2) promote (enam, n) ;

従業員名 enam の給料を n % up する。

```
        replace e ( sal = e.sal * ( 1 + n/100 )) where ⓐ e.enam = enam;
```

3) degrade (enam , n);

enam の給料を n % down

```
        replace e ( sal = e.sal * (1 - n/100 )) where ⓐ e.enam = enam;
```

4) resign (enam)

従業員 enam が退職する。

関連する部, プロジェクトからも脱ける。

```
        retrieve into R ( ename = e. EMP) where e.enam =enam ;
        range ( r, R );
        delete  e  where e.*EMP = r.ename;
        delete pa where pa.*EMP= r.ename;
        delete pm where pm.*PMng = r.ename;
        delete dm where dm.*PMng = r.ename;
        delete  r ;
```

5) marriage (ename , nename)

従業員 enam が結婚する。女性たちは名前が nename にかわり, 男性ならばそのままである。

```
        replace e ( mst =M, ename = nename )
        where e.enam = ename   and
              e.sex = "M";
        replace e ( mst=M )
        where e.enam = ename   and
              e.sex = "F" ;
```

6) transpose (ename , ndept)

従業員 enam の部か, ndept にかかる。

```
        retrieve into R1 ( emply = e.*EMP )
        where e.enam = ename ;
        retrieve into R2 ( deptm = d.*DEPT )
        where d.dnam = ndept ;
        range   (r1, R1 ) ( r2, R2 );
        replace   a(*DEPT = r2.deptm )
```

```
where    a.*EMP = r1.emply ;
delete   r1 ;
delete   r2 ;
```

4.2.3 Project-manager ビュー

これは, project管理者がみるビューである。

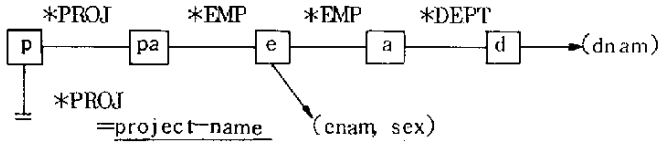
A ビューリレーションの定義

PROGECT (enam , sex , dnam)

管理者の属するプロジェクト内のメンバーの名前(enam),性別(sex),彼の属する部名(dnam)を表わしている。

```
define  PROJECT
    ( enam = e.enam , sex = e.sex , dnam = d.dnam )
where
    e.*EMP = pa.*EMP  and  pa.*PROJ = p.*PROJ
and
    p.*PROJ = project-name  and
    e.*EMP = a.*EMP  and  a.*DEPT = d.*DEPT ;
```

ここで, project-name は, このprojectのオブジェクト識別子である。



B ビューオペレーションの定義

1) name PROJ (pname) ;

自分のprojectに名前を与える。

```
replace  p  ( pnam = pname )
where  p.*PROJ = project-name ;
```

2) pickout (ename)

自分のプロジェクトのメンバとしてenameを加える。enameはビューリレーションにはないが, プロジェクトマネージャはこの人を知っているとする。

```
append to P-ASSIGN ( *EMP = e.*EMP, *PROJ = projname )
where
    e.ename = ename ;
```

4. 3　ビューの代数的定義

図 2. 2. 1 のような基本関係よりなるデータベースにおいて，Executive, personnel Department, Project-Manager に対するビューを代数的に定義する。

ビュー定義の一般形として，

```
define-view ＜view name＞
  view-relation
    do
      ＜view relation＞(＜view attribute＞, …　　)
    =関係演算式によるビューの定義；
                ⋮
    od;
  view-operation
    do
      procedure　＜operation name＞(＜argument＞,… )
        do
          ＜本　体＞
        od ;

    od ;
end-def ;
```

を用いる。

```
define-view　　executive's view
  view-relation
    do
      PROJ ( pnam , sy, ey, pmng, #-emp-proj )
       = ∨((EMP(*Emp , enam )(cnam\pmng )(*Emp = *PMng )
            P-Mng (*Proj = *proj )
            PROJ(*Proj , pnam, sy, ey )(*PROJ = *Proj )
            P-ASSIGN(*Proj <#-emp-proj > *Emp ))
           (*proj , *PMng, pnam, sy, ey, pmng, #-emp-proj))
      DEPT ( dnam , #-emp-dept )
       = ∨((Dept (*Dept, dnam )(*Dept = *Dept )
            AFFILIATION(*Dept <#-emp-dept > *Emp))
           (*Dept , dnam, #-emp-dept ));
    od;
  view-operation
```

```
do
  procedure   abolish-dept
    do
      delete (*Dept) ;
    od ;
  procedure   creat-dept
    do
      create(*Dept ) ;
    od ;
  procedure   merge(x)
    do
      merge (*Dept, dnam = x ) ;
    od ;
  procedure   discontinue proj
    do
      delete (*proj ) ;
    od ;
  procedure   create proj
    do
      create (*proj ) ;
    od ;
  procedure   merge proj(x)
    do
      merge (*proj, pnam = x ) ;
    od ;
  procedure   raisesal(x)
    do
      EMP[ sal ← sal*( 1+x/100 )] ;
    od ;
  procedure   assign
    do
      *PMng-reg ← (*Emp-reg ) ;
      delete ( D-MNG , *Dept ) ;
      create ( D-MNG) ;
    od ;
```

```
    procedure   discharge
        do
          *PMng-reg ← (*Emp-reg ) ;
          delete ( D-MNG, *PMng ) ;
        od ;
    od ;
  end-def ;
define view      personnel view
  view-relation
    do
      ENPPROFILE ( enam, sal, sex, birth, dmng, dnam, #-emp, pnam )
       =∨((((EMP〔*EMP, enam〕〔enam\dmng〕〔*Emp = *DMng〕
          D-MNG〔*Dept, *DMng〕)〔*DMng, dnam〕〔*Dept = *Dept 〕
          DEPT〔*Dept, dnam〕〔*Dept = *Dept 〕
          AFFILIATION〔*Dept, *Emp〕〔*Emp = *Emp〕
          EMP〔*Emp, enam, sal, sex, birth〕〔*Emp = *Emp〕
          P-ASSIGN〔*Emp, *roj〕〔*proj = *proj〕
          PROJ〔*proj, pnam〕)〔*Dept = *Dept 〕
          (AFFILIATION〔*Dept <#-emp> *Emp〕))
          〔*Emp, *Dept, *proj, *DMng, enam, sal, sex, birth, dmng, dnam,
          #-emp, pnam〕);
    od ;
  view-operation
    do
      procedure   hire
          do
            create ( *Emp) ;
          od ;
      procedure   promote(x)
          do
            EMP ← EMP < *Emp〕( EMP〔*Emp = (*Emp-reg)〕
                                〔sal ← sal * ( 1 + x/100 )〕) ;
          od ;
      procedure   degrade(x)
          do
```

```
            EMP ← EMP < *Emp ) ( EMP [ *Emp : ( *Emp-reg ) ]
                                    [ sal ← sal * ( 1 - x / 100 ) ] ) ;
        od;
      procedure    resign
        do
          delete ( *Emp ) ;
        od ;
      procedure    marriage
        do
          EMP ← EMP < *Emp ) ( EMP [ *Emp = ( *Emp-reg ) ]
                                    [ mst ← M ] ) ;
        od ;
      procedure    transpose
        do
          delete ( AFFILIATION *Emp ) ;
          create ( AFFILIATION ) ;
        od ;
define-view      project-manager's view
  view-relation
    do
      PMNG ( enam, sex, dnam )
       = ∨( ( EMP [ *Emp, enam, sex ] [ *EMP=*EMP ]
            AFFILIATION [ *Dept = *Dept ]
            DEPT ) [ *Emp, *Dept, enam, sex, dnam ] ) ;
    od ;
  view operation
    do
      procedure    name-proj(x)
        do
          PROJ ← PROJ < *Proj ] ( PROJ [ *Proj = ( *Proj-rel ) ] [ pnam←x ] ) ;
        od ;
      procedure    pick-out(x)
        do
          EMP [ enam=x ] ;        create ( P-ASSIGN ) ;
        od ;
```

4. まとめと今後の課題

本論文の成果を以下にまとめる。

(A) オブジェクトモデルの提案

オブジェクトの概念をデータベースに持ち込むことにより，データベースの更新操作と，実世界での実体とその値の可能な変化とを対応づけることができた。O－リレーションとA－リレーションの間の更新規則と，オブジェクトの生成消滅規則がこれである。

オブジェクトとは，データベースにモデル化する実世界における普遍な存在である。一方ロールとは，オブジェクトがあるとき，ある環境において持つ様相である。例えば，今ここに1人の人がいるとすると，この人は1つのオブジェクトである。彼が学生として，データベースにおいて認識されているとすると，“学生である彼”は“学生”というロールの取り得る1つの値である。又，彼は，就職すれば“会社員”というロールを持つ。

これに対して，E－Rモデルにおけるエンティティーは，オブジェクト・モデルのロールであり，E－Rモデルにはオブジェクトの概念はない。したがって，E－Rモデルでは，実世界の普遍な実体をデータベース中にモデル化することはできない。

オブジェクト・モデルの，数学的な定義も行ない，オブジェクト・モデルにおけるFDやMVDといった従属関係（i.e, $X \rightarrow Y$, $X \twoheadrightarrow Y$, ここでXはロール名の集合，Yは属性の集合）に対して，次の様な解釈を行なった。すなわち，YがX中のロール間の関連に固有な性質（属性）と解釈する。このことにより，これらの従属関係に，自然な実際的な意味を持たすことが可能となった。これによって，データベースの論理設計の本質的な問題の1つを解決した。

さらに，オブジェクト・モデルにおけるリレーションの正規形を定義し，実世界におけるオブジェクトとその性質や，オブジェクト間の関連とその固有な性質を論理的な枠組の中で，明確に定義できた。このことは，設計と利用との両面において有用である。とくに，更新の前後におけるデータベースのインテグリティの保存を考えるうえに有効な枠組を与えた。

(B) オブジェクト言語

関係代数をベースにして，これに更新機能を加えた代数的言語を提案した。これはオブジェクトモデルのみならず，リレーショナルモデルにも通用可能である。

更新処理は，オブジェクトの生成，消滅と，属性の値の変更とだけが許されるようにした。この制限以外は，リレーショナル・モデルにおけるデータ操作言語と，オブジェクト・モデルにおけるデータ操作言語とは同じである。

(C) ビューとデータの抽象化

ビューを単に，リレーションとして定義するのではなく，そのリレーションと，これに関連する更新オペレーションとの対として定義した。このことにより，ビュー演算を，オブジェクトの概念を基本にした基本関係に対する更新演算によって明確に定義できる。これにより，ビューの更新問題が解決できた。この手法は，プログラミング言語におけるデータ抽象化と類似している。

以上を，オフィスにおける情報処理を例にして，その実用性を示した。

以下に，オブジェクト・モデルの今後の課題を示す。

(1) ビュー・リレーションの定義において，問題となるnull値をオブジェクト・モデルに取り込む。特にDDBSのGCS(Global Conceptual Schema)をビューとして定義するためにnull値の取り扱いは重要である。

(2) ビューを基本関係の和として定義する。

(3) オブジェクト言語の仕様の明確化。特に関数的な言語が有効と考えている。

(4) 意味的な同時更新問題の解決。

ロックの単位を決定するために，オブジェクト・モデルの概念が有効であると考えている。

付記Ⅱ　同種化システム

付記II　同種化システム
（ローカル情報記述設計システム）

1. 概　要

同種化システムは，CODASYL，DBTGモデルに基づいたＬＩＳを基本にして，ローカル概念スキーマ（ LCS ）を生成するとともに，問合せ変換〔第７章〕で必要なDD/Dの異種性情報を生成するシステムである。このシステムは基本的に〔 TAKIM 79〕に基づいている。

DBTGモデルを，概念スキーマ層と内部スキーマ層との要素に，以下の様に分離した。

1）概念スキーマ層

i）レコード型	名　前
	データ項目構成
	（事象）キーとなる項目／項目集合
ii）セット型	名　前
	親レコード型と子レコード型
	メンバシップクラス（MANDATORY/AUTOMATIC(M/A)）
	構造制限（structural constraint）
iii）リンクレコード型	名　前
	関連づけるレコード型
	関数制（１：n or n：m）
	データ項目構成

2）内部スキーマ層

i）インデクス	主インデクス　DNA CALC
	２次インデクス　DA CALC
ii）アクセスリンク	NEXT，PRIOR，OWNERリンク
iii）順序づけ	ソート指定

逆に，DBTGモデルとしては，上述した要素のみ考えることにする。

スキーマ変換を考える時に，２つのスキーマが互いに基礎を置くデータモデルが扱い得るスキーマ層のレベルを考える必要がある。リレーショナルモデルは，概念スキーマ層のモデルである。しかし，DBTGモデルは，Aで述べた様に概念スキーマと内部スキーマとの２つの要素をもっている。このため，リレーショナルモデルで表わし得るのは，DBTGモデルの概念スキーマレベルの要素だけである。

データモデルの概念スキーマ層を，Ｅ－Ｒモデルによって考えることにする。各モデルを，まずＥ－Ｒモデルに対応づける。その後に対応するリレーショナルモデル記述を生成する。これは次のようにしてなされる。

1）レコード型を事象集合リレーション（ ESR ）とする。レコード型の項目をESRの属性とする。

2）ESRのキー属性を定める。

i）DNA，CALC項目／項目集合をキー属性とする。

ii) M／Aの単一セットの子レコード型の時，このセットがDNA指定した項目／項目集合をキー属性とする。

iii) この他の時は，データベースキー（DBキー）の代表者（surrogate）としての仮想属性を，ESRに設け，キーとする。

3) リンクレコード型を関係性集合リレーション（RSR）とする。このリンクレコード型が関連づけるレコード型のESRのキー属性の集合を，このRSRのキー属性とする。リンクレコード型がデータ項目を持てば，これをRSRの属性とする。

4) リンクレコード型と関連しないセット型を，関係性集合リレーション（RSR）とする。これの親と子レコード型のESRのキー属性を，RSRのキー属性とする。

3) と4) において，構造制限をもった子レコード型の項目が，親レコード型のキー項目の時は，この項目をRSRの属性とはしない。何故なら，RSRのキー属性としてESRから移入されるからである。

内部スキーマ層の情報は，問合せ変換〔TAKIM 80 a, b, 第6章〕の最適化において重要である。これらの情報は，異種性情報（HI）内のパフォーマンス情報として格納される。

HIは，次の3種の7つのリレーションから成っている。HIをリレーショナルモデルによって実現することは，システムのインプリメンテーションと運用とを容易にできる。HIも，他のLCSリレーションと同様に扱かえるからである。

a) LCSSI	RSR ESR ATT	LCSのリレーショナル記述とLISとの対応情報
b) LISSI	REC SET ITEM	LIS（CODASYLスキーマ）のリレーショナル記述
c) DMLI	DML	CODASYL DML情報

同種化システムは，図Ⅱ－1の様に2つの主要なモヂュールから構成されている。

1) LISG

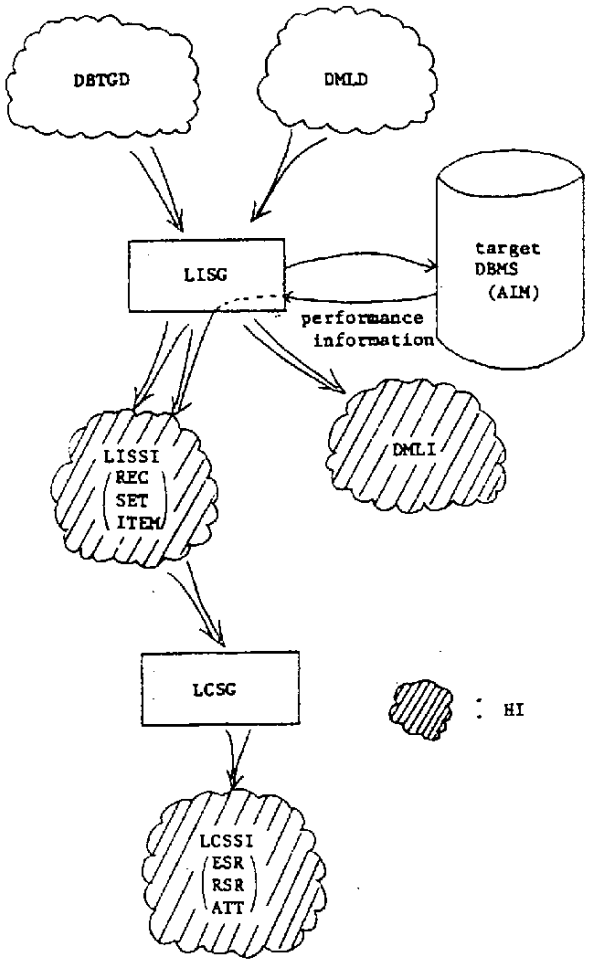
2) LCSG

LISGは，次のことを行なう。

○ DBTGスキーマ記述（DBTGDL〔図Ⅱ．2〕によって記述されていて，これをDBTGDと呼ぶ）を入力として，LISSIの3つのリレーションを生成する。この時，実際のDBS（i.e. PRDBS〔図Ⅱ3〕）をアクセスして，DBオカーランス情報をLISSIに格納する。

○ DML記述（DMLD）を入力として，DMLIを生成する。

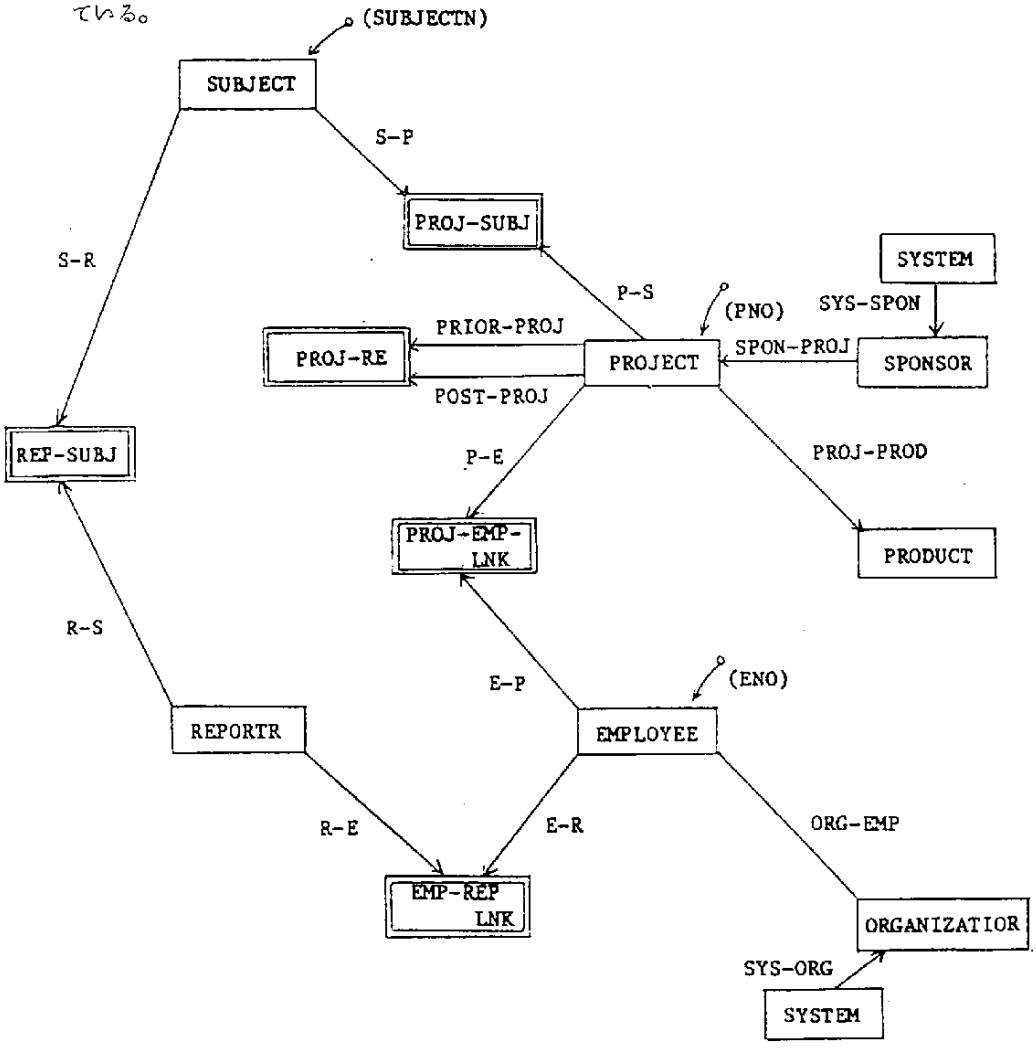
LCSSGは，LISSIを入力として，LCSSIを生成する。LISSIは，CODASYLスキーマのリレーショナル記述である。先に述べた変換アルゴリズムによって対応するLCSを生成する。



図Ⅱ．1　同種化システムの概要

2. プロジェクトデータベースシステム（PRDBS）

プロジェクトDBSのスキーマ（LIS）を図Ⅱ．3に示し，このDDLを図Ⅱ．4に示す。PRDBSはプロジェクト管理を目的とするDBSで，当協会開発部のプロジェクト情報の一部が格納されている。PRDBSは又，当協会のM－160上のAIM DBMSを用いてインプリメントされている。



図Ⅱ．3　PRDBSのスキーマ

```
INPUT CONTROL STATEMENT PROCESS (SCHEMA COMMAND)
ST-NO ....*....1....*....2....*....3....*....4....*....5....*....6....*...
      *                    ********************
      *                    * SCHEMA COMMAND   *
      *                    ********************
      *                    ********************
      *                    * SCHEMA ENTRY     *
      *                    ********************
    1 SCHEMA NAME IS PROJDBS FOR DB?
      AUTHOR IS H-SUZUKI.
      DATE-WRITTEN 1980-08-18.
      REMARKS PROJECT DATABASE SYSTEM.
      *                    ********************
      *                    * RECORD ENTRY     *
      *                    ********************
    2 RECORD NAME IS EMPLOYEE;
        LOCATION MODE IS RANDOM USING EMO
          DUPLICATES ARE NOT ALLOWED.
    3   02 EMO PIC IS 9(05).
    4   02 ENAME PIC IS X(10).
    5   02 ERNAME PIC IS X(20).
    6   02 ESEX PIC IS X.
    7   02 BIRTH-PLACE PIC IS X(10).
    8   02 BIRTH-YEAR PIC IS 9(02).
    9   02 BIRTH-MONTH PIC IS 9(02).
   10   02 ALMA-MATER PIC IS X(30).
   11   02 MAJOR PIC IS X(30).
   12   02 DEGREE PIC IS X.
   13   02 DEG-YEAR PIC IS 9(02).
   14   02 HIRED-YEAR PIC IS 9(02).
   15   02 RETIRED-YEAR PIC IS 9(02).
   16   02 DEPT PIC IS X(20).
   17   02 EPOSITION PIC IS X(20).
   18   02 TEL PIC IS X(12).
   19   02 EXT PIC IS 9(04).
   20   02 FIL PIC IS X(27).

   21 RECORD NAME IS PROJECT;
        LOCATION MODE IS RANDOM USING PNO
          DUPLICATES ARE NOT ALLOWED.
   22   02 PNO PIC IS X(08).
   23   02 PNAME PIC IS X(100).
   24   02 PROJ-TYPE PIC IS X.
   25   02 STATUS-P PIC IS X(03).
   26   02 SYEAR PIC IS 9(02).
   27   02 EYEAR PIC IS 9(02).
   28   02 BUDGET PIC IS 9(11).
   29   02 FIL PIC IS X(73).

   30 RECORD NAME IS REPORTS;
        LOCATION MODE IS RANDOM USING RNO
          DUPLICATES ARE NOT ALLOWED.
   31   02 RNO PIC IS X(08).
   32   02 TITLE PIC IS X(200).
   33   02 SOURCE-D PIC IS X(100).
   34   02 VOL PIC IS X(08).
   35   02 NUMB PIC IS 9(03).
   36   02 MONTH PIC IS 9(02).
   37   02 YEAR PIC IS 9(02).
   38   02 FROM-PAGE PIC IS 9(05).
   39   02 TO-PAGE PIC IS 9(05).
   40   02 TOTAL-PAGE PIC IS 9(05).
   41   02 NO-OF-AUTHOR PIC IS 9(02).
   42   02 LANGUAGE PIC IS X(10).
   43   02 WRITTEN-YEAR PIC IS 9(02).
   44   02 COST PIC IS 9(09).
   45   02 RTYPE PIC IS X.
   46   02 RSTATE PIC IS X.
   47   02 FIL PIC IS X(32).
```

PRDBSのスキーマ記述(1)

```
   48 RECORD NAME IS SPONSOR.
        * NO CALC KEY *
   49   02 SNAME PIC IS X(20).
   50   02 STYPE PIC IS X(10).
   51   02 FIL PIC IS X(70).

   52 RECORD NAME IS SUBJECTN;
        LOCATION MODE IS RANDOM USING SUBJECTN
          DUPLICATES ARE NOT ALLOWED.
   53   02 SUBJECTN PIC IS X(100).
   54   02 FIL PIC IS X(100).

   55 RECORD NAME IS PRODUCT.
        * NO CALC KEY *
   56   02 PD-NO PIC IS X(05).
   57   02 PD-NAME PIC IS X(30).
   58   02 PTYPE PIC IS X(01).
   59   02 COST PIC IS 9(09).
   60   02 YEAR PIC IS 9(02).
   61   02 LANGUAGE PIC IS X(10).
   62   02 PSIZE PIC IS 9(05).
   63   02 OS-NAME PIC IS X(15).
   64   02 FIL PIC IS X(23).

   65 RECORD NAME IS ORGANIZATION.
   66   02 ORG-NAME PIC IS X(30).
   67   02 ORG-TYPE PIC IS X(10).
   68   02 FIL PIC IS X(90).
      *                    ********************
      *  LINK RECORD TYPE  *
      *                    ********************
   69 RECORD NAME IS PROJ-EMP-LINK.
   70   02 STATUSS PIC IS X(06).
   71   02 SYEAR PIC IS 9(02).
   72   02 SMONTH PIC IS 9(02).
   73   02 EYEAR PIC IS 9(02).
   74   02 EMONTH PIC IS 9(02).
   75   02 POSITION PIC IS X(10).
   76   02 ROLE PIC IS X(10).
   77   02 PERCENTAGE PIC IS 9(03).

   78 RECORD NAME IS EMP-REP-LINK.
   79   02 AUTHNO PIC IS 9(02).

   80 RECORD NAME IS REP-SUBJ.

   81 RECORD NAME IS PROJ-SUBJ.

   82 RECORD NAME IS PROJ-RE.
      *                    ********************
      *                    * SET ENTRY        *
      *                    ********************
   83 SET NAME IS LEADER;
        OWNER IS EMPLOYEE;
        MEMBER IS PROJECT LINKED TO OWNER PRIOR;
        INSERTION IS SORTED USING PNO
          DUPLICATES ARE NOT ALLOWED.

   84 SET NAME IS SPON-PROJ;
        OWNER IS SPONSOR;
        MEMBER IS PROJECT LINKED TO OWNER PRIOR;
        INSERTION IS SORTED USING PNO
          DUPLICATES ARE NOT ALLOWED.
```

PRDBSのスキーマ記述(2)

図Ⅱ．4 PRDBSのDDL

```
 85    SET NAME IS        PROJ-PROJ;
       OWNER IS           PROJECT;
       MEMBER IS          PROJECT      LINKED TO OWNER PRIOR;
       INSERTION IS       SORTED USING   PO-NO
            DUPLICATES ARE NOT ALLOWED.

 86    SET NAME IS        P-E;
       OWNER IS           PROJECT;
       MEMBER IS          PROJ-EMP-LNK   LINKED TO OWNER;
       INSERTION IS       LAST.

 87    SET NAME IS        E-P;
       OWNER IS           EMPLOYEE;
       MEMBER IS          PROJ-EMP-LNK   LINKED TO OWNER;
       INSERTION IS       LAST.

 88    SET NAME IS        P-S;
       OWNER IS           PROJECT;
       MEMBER IS          PROJ-SUBJ      LINKED TO OWNER;
       INSERTION IS       LAST.

 89    SET NAME IS        S-P;
       OWNER IS           SUBJECT;
       MEMBER IS          PROJ-SUBJ      LINKED TO OWNER;
       INSERTION IS       LAST.

 90    SET NAME IS        S-R;
       OWNER IS           SUBJECT;
       MEMBER IS          REP-SUBJ       LINKED TO OWNER;
       INSERTION IS       LAST.

 91    SET NAME IS        R-S;
       OWNER IS           REPORTER;
       MEMBER IS          REP-SUBJ       LINKED TO OWNER;
       INSERTION IS       LAST.

 92    SET NAME IS        R-E;
       OWNER IS           REPORTER;
       MEMBER IS          EMP-REP-LNK    LINKED TO OWNER;
       INSERTION IS       SORTED USING   AUTH-NO
            DUPLICATES ARE NOT ALLOWED.

 93    SET NAME IS        E-R;
       OWNER IS           EMPLOYEE;
       MEMBER IS          EMP-REP-LNK    LINKED TO OWNER;
       INSERTION IS       LAST.

 94    SET NAME IS        PRIOR-PROJ;
       OWNER IS           PROJECT;
       MEMBER IS          PROJ-RE        LINKED TO OWNER.

 95    SET NAME IS        POST-PROJ;
       OWNER IS           PROJECT;
       MEMBER IS          PROJ-RE        LINKED TO OWNER.

 96    SET NAME IS        ORG-EMP;
       OWNER IS           ORGANIZATION;
       MEMBER IS          EMPLOYEE       LINKED TO OWNER PRIOR;
       INSERTION IS       SORTED USING   EMP
            DUPLICATES ARE NOT ALLOWED.

 97    SET NAME IS        SYS-ORG;
       OWNER IS           SYSTEM;
       MEMBER IS          ORGANIZATION.

 98    SET NAME IS        SYS-SPON;
       OWNER IS           SYSTEM;
       MEMBER IS          SPONSOR.
```

PRDBSのスキーマ記述(3)

```
                          ********************
                          * RANGE ENTRY      *
                          ********************

 99    RANGE NAME IS      RANGEA;
       RECORD NAME IS     PROJECT;
       PRIME IS           4;
       EXCLUSIVE UNIT IS  RANGE.

100    RANGE NAME IS      RANGEB;
       RECORD NAME IS     SPONSOR;
       PRIME IS           2;
       EXCLUSIVE UNIT IS  RANGE.

101    RANGE NAME IS      RANGEC;
       RECORD NAME IS     PRODUCT;
       PRIME IS           3;
       EXCLUSIVE UNIT IS  RANGE.

102    RANGE NAME IS      RANGED;
       RECORD NAME IS     SUBJECT, PROJ-SUBJ;
       PRIME IS           4;
       OVERFLOW IS        8.

103    RANGE NAME IS      RANGEE;
       RECORD NAME IS     REPORTER, REP-SUBJ;
       PRIME IS           2;
       OVERFLOW IS        5;
       LOGICAL PAGE IS    4;
       EXCLUSIVE UNIT IS  RANGE.

104    RANGE NAME IS      RANGEF;
       RECORD NAME IS     EMPLOYEE, EMP-REP-LNK;
       PRIME IS           2;
       OVERFLOW IS        2;
       EXCLUSIVE UNIT IS  RANGE.

105    RANGE NAME IS      RANGEG;
       RECORD NAME IS     PROJ-EMP-LNK;
       PRIME IS           2.

106    RANGE NAME IS      RANGEH;
       RECORD NAME IS     PROJ-RE;
       PRIME IS           2.

107    RANGE NAME IS      RANGEI;
       RECORD NAME IS     ORGANIZATION;
       PRIME IS           2.

                          ********************
                          * DATASET ENTRY    *
                          ********************

108    DATASET NAME IS    PROJDB;
       PAGE CONTAINS      2000.

                          ***** VOLUME SUBENTRY *****

109    VOLUME IS          USER01;
       UNIT NAME IS       FA191
       LOCATE             RANGEA, RANGEB, RANGEC, RANGED, RANGEE,
                          RANGEF, RANGEG, RANGEH, RANGEI.

110    END.

%KD060I-I  SCHEMA PROJDBS SOURCE CHECK FINISHED.
```

PRDBSのスキーマ記述(4)

図Ⅱ.4　PRDBSのDDL

```
INPUT  CONTRUL  STATEMENT  PROCESS.     (SUBSCHEMA COMMAND)
ST-NO   ....+....1....+....2....+....3....+....4....+....5....+....6....+
                *
                *                          ********************
                *                          *  SUBSCHEMA COMMAND *
                *                          ********************
                *                                  ************************
                *                                  *  SUBSCHEMA  ENTRY    *
                *                                  ************************
                *
   1            SUBSCHEMA NAME IS       PROJSUB1;
                    SCHEMA NAME IS      PROJDB$1
                    AUTHOR IS           M-SUZUKI;
                    DATE-WRITTEN        1980-08-20.
                *   REMARKS             ALL FEATURE OF SCHEMA PROJDB$.
                *
                *                                  ******************
                *                                  *  RECORD ENTRY  *
                *                                  ******************
                *
   2                01  EMPLOYEE.
   3                01  PROJECT.
   4                01  REPORTER.
   5                01  SPONSOR.
   6                01  SUBJECT.
   7                01  PRODUCT.
   8                01  ORGANIZATION.
   9                01  PROJ-EMP-LINK.
  10                01  EMP-REP-LINK.
  11                01  REP-SUBJ.
  12                01  PROJ-SUBJ.
  13                01  PROJ-PE.
                *
                *                                  ****************
                *                                  *  SET ENTRY   *
                *                                  ****************
                *
  14                SET NAME IS   LEADER.
  15                SET NAME IS   SPON-PROJ.
  16                SET NAME IS   PROJ-PROD.
  17                SET NAME IS   P-E.
  18                SET NAME IS   E-P.
  19                SET NAME IS   P-S.
  20                SET NAME IS   S-P.
  21                SET NAME IS   S-R.
  22                SET NAME IS   R-S.
  23                SET NAME IS   R-E.
  24                SET NAME IS   E-R.
  25                SET NAME IS   PRIOR-PROJ.
  26                SET NAME IS   POST-PROJ.
  27                SET NAME IS   ORG-EMP.
                *
  28            END.
JX00A01-I   SUBSCHEMA PROJSUB1 SOURCE CHECK FINISHED.
```

ＰＲＤＢＳのサブスキーマ記述

図Ⅱ．４　ＰＲＤＢＳのＤＤＬ

3. DBTGDL と DMLD

DBTGDL は，ＤＢＴＧスキーマの共通記述等として位置づけられる。この形式を以下に示す。
2.のPRDBSに対するDBTGDL 記述を図Ⅱ．5に示す。

```
〈DBTGDL〉::=〈DBTG statement〉{;〈DBTG statement〉}
〈DBTG statement〉::=〈record-type-description〉|〈set-type-description〉
〈record-type-description〉::=REC〔ORD-TYPE〕:〈area-name〉〔:〈rec-mode〉〕
                                    〈record-type〉
                                        〔(〈field-list〉)〕
〈rec-mode〉::= LINK|NORMAL*
〈field-list〉::=〈field-description〉{, 〈field-description〉}
〈field-description〉::=〈field-name〉/〈format〉
                                    〔/〈access-mode〉〕
〈access-mode〉::=〈calc-mode〉/〈dna-mode〉
〈dna-mode〉::= DNA/DA*
〈calc-mode〉::= CALC/NCALC*
〈format〉::=〈type〉〈length〉
〈type〉::= C|D
```

(註) ＊は省略時解釈を示す。

ＤＢＴＧＤＬの定義(1)

```
〈set-type-description〉::= SET〔-TYPE〕〈set-type〉
                                    〈owner-description〉
                                    〈member-description〉
                                    〔〈structural constraint〉〕
〈owner-description〉::= OWNE〔R〕〈record-type〉
〈member-description〉::= MEMB〔ER〕〈record-type〉
                                〔/〈access-mode〉〕
〈access-mode〉::=〈dna-att〉/〈sort-att〉〔/〈link-mode〉
                        〔/〈insertion-mode〉/〈retension-mode〉〕〕
〈dna-att〉::=〈attribute-name〉|NIL
〈sort-att〉::=〈attribute-name〉|NIL
```

〈insertion−mode〉::= AUTOMATIC*/MANUAL

〈retension−mode〉::= FIXED/OPTIONAL/MANDATORY*

〈link−mode〉::= NEXT | OWNER

〈structural−constraint〉::= STRUCTURE (〈att−name−of−o−rec〉,
〈att−name−of−m−rec〉)

DBTGDLの定義(2)

DML記述は，DMLDLによって記述される。DMLDは，問合せ変換（QTP）において用いられるDMLブロックを記述する。

この記述は，DMLブロック番号毎に，COBOL DMLの文集合を記述する。

DMLDLの構文を以下に示す。

〈DMLD〉::=〈DML−block−def〉 {〈DML−block−def〉}

〈DML−block−def〉::=〈block−no〉//〈DML−statement〉
{// 〈DML−statement〉};

〈block−no〉::= P〈number〉

DMLDLの定義

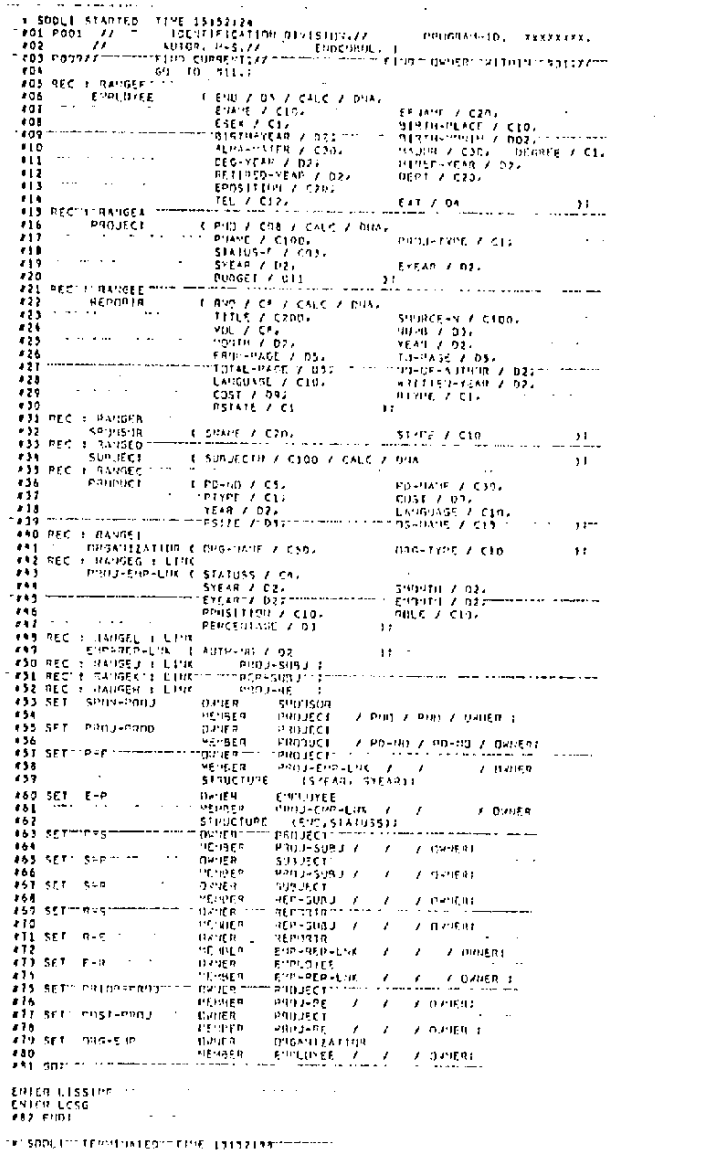


図４　２３　ＰＲＤＢＳのＤＢＴＧＤ

3. PRDBSの異種性情報

A LCSSI

1) ESR

** DISPLAY HIESR

ES-REL NAME	DEGREE	TARGET-NAME	TARGET-TYPE
SPONSOR	2	SPONSOR	0
PROJECT	7	PROJECT	0
PRODUCT	8	PRODUCT	0
EMPLOYEE	17	EMPLOYEE	0
SUBJECT	1	SUBJECT	0
REPORTR	16	REPORTR	0
ORGANIZATIOR	2	ORGANIZATIOR	0

2) RSR

** DISPLAY HIRSR

RS-REL NAME	DEGREE	S-ES-REL	D-ES-REL	TARGET-NAME	TARGET-TYPE
SPON-PROJ	2	SPONSOR	PROJECT	SPON-PROJ	2
PROJ-PROD	2	PROJECT	PRODUCT	PROJ-PROD	2
PROJ-EMP-LNK	10	PROJECT	EMPLOYEE	PROJ-EMP-LNK	1
PROJ-SUBJ	2	PROJECT	SUBJECT	PROJ-SUBJ	1
REP-SUBJ	2	SUBJECT	REPORTR	REP-SUBJ	1
EMP-REP-LNK	3	REPORTR	EMPLOYEE	EMP-REP-LNK	1
PROJ-RE	2	PROJECT	PROJECT	PROJ-RE	1
ORG-EMP	2	ORGANIZATIOR	EMPLOYEE	ORG-EMP	2

3) ATT

```
** DISPLAY _HIATT
```

ES-RS-REL	REL-TYPE	ATT#	ATT-NAME	DOMAIN	C/D	LENGTH	TARGET#	TARGET-NAME	ROLE
SPON-PROJ	1:1	1	SNAME	SNAME	C	20	1	SPONSOR	01
SPON-PROJ	1:1	2	PNO	PNO	C	8	1	PROJECT	11
SPONSOR	01	1	SNAME	SNAME	C	20	1	SPONSOR	01
SPONSOR	01	2	STYPE	STYPE	C	10	2	SPONSOR	01
PROJECT	01	1	PNO	PNO	C	8	1	PROJECT	11
PROJECT	01	2	PNAME	PNAME	C	100	2	PROJECT	01
PROJECT	01	3	PROJ-TYPE	PROJ-TYPE	C	1	3	PROJECT	01
PROJECT	01	4	STATUS-F	STATUS-F	C	3	4	PROJECT	01
PROJECT	01	5	SYEAR	SYEAR	D	2	5	PROJECT	01
PROJECT	01	6	EYEAR	EYEAR	D	2	6	PROJECT	01
PROJECT	01	7	BUDGET	BUDGET	D	11	7	PROJECT	01
PROJ-PROD	1:1	1	PNO	PNO	C	8	1	PROJECT	11
PROJ-PROD	1:1	2	PD-NO	PD-NO	C	5	1	PRODUCT	01
PRODUCT	01	1	PD-NO	PD-NO	C	5	1	PRODUCT	01
PRODUCT	01	2	PD-NAME	PD-NAME	C	30	2	PRODUCT	01
PRODUCT	01	3	PTYPE	PTYPE	C	1	3	PRODUCT	01
PRODUCT	01	4	COST	COST	D	9	4	PRODUCT	01
PRODUCT	01	5	YEAR	YEAR	D	2	5	PRODUCT	01
PRODUCT	01	6	LANGUAGE	LANGUAGE	C	10	6	PRODUCT	01
PRODUCT	01	7	PSIZE	PSIZE	D	5	7	PRODUCT	01
PRODUCT	01	8	OS-NAME	OS-NAME	C	15	8	PRODUCT	01
PROJ-EMP-LNK	1:1	1	PNO	PNO	C	8	1	PROJECT	11
PROJ-EMP-LNK	1:1	2	ENO	ENO	D	5	1	EMPLOYEE	11
PROJ-EMP-LNK	1:1	3	STATUSS	STATUSS	C	4	1	PROJ-EMP-LNK	01
PROJ-EMP-LNK	1:1	4	SYEAR	SYEAR	D	2	2	PROJ-EMP-LNK	01
PROJ-EMP-LNK	1:1	5	SMONTH	SMONTH	D	2	3	PROJ-EMP-LNK	01
PROJ-EMP-LNK	1:1	6	EYEAR	EYEAR	D	2	4	PROJ-EMP-LNK	01
PROJ-EMP-LNK	1:1	7	EMONTH	EMONTH	D	2	5	PROJ-EMP-LNK	01
PROJ-EMP-LNK	1:1	8	PPOSITION	PPOSITION	C	10	6	PROJ-EMP-LNK	01
PROJ-EMP-LNK	1:1	9	ROLE	ROLE	C	10	7	PROJ-EMP-LNK	01
PROJ-EMP-LNK	1:1	10	PERCENTAGE	PERCENTAGE	D	3	8	PROJ-EMP-LNK	01
EMPLOYEE	01	1	ENO	ENO	D	5	1	EMPLOYEE	11
EMPLOYEE	01	2	ENAME	ENAME	C	10	2	EMPLOYEE	01
EMPLOYEE	01	3	EFNAME	EFNAME	C	20	3	EMPLOYEE	01
EMPLOYEE	01	4	ESEX	ESEX	C	1	4	EMPLOYEE	01
EMPLOYEE	01	5	BIRTH-PLACE	BIRTH-PLACE	C	10	5	EMPLOYEE	01
EMPLOYEE	01	6	BIRTH-YEAR	BIRTH-YEAR	D	2	6	EMPLOYEE	01
EMPLOYEE	01	7	BIRTH-MONTH	BIRTH-MONTH	D	2	7	EMPLOYEE	01
EMPLOYEE	01	8	ALMA-MATER	ALMA-MATER	C	30	8	EMPLOYEE	01
EMPLOYEE	01	9	MAJOR	MAJOR	C	30	9	EMPLOYEE	01
EMPLOYEE	01	10	DEGREE	DEGREE	C	1	10	EMPLOYEE	01
EMPLOYEE	01	11	DEG-YEAR	DEG-YEAR	D	2	11	EMPLOYEE	01
EMPLOYEE	01	12	HIRED-YEAR	HIRED-YEAR	D	2	12	EMPLOYEE	01
EMPLOYEE	01	13	RETIRED-YEAR	RETIRED-YEAR	D	2	13	EMPLOYEE	01
EMPLOYEE	01	14	DEPT	DEPT	C	20	14	EMPLOYEE	01
EMPLOYEE	01	15	EPOSITION	EPOSITION	C	20	15	EMPLOYEE	01
EMPLOYEE	01	16	TEL	TEL	C	12	16	EMPLOYEE	01
EMPLOYEE	01	17	EXT	EXT	D	4	17	EMPLOYEE	01
PROJ-SUBJ	1:1	1	PNO	PNO	C	8	1	PROJECT	11
PROJ-SUBJ	1:1	2	SUBJECTN	SUBJECTN	C	100	1	SUBJECT	11
SUBJECT	01	1	SUBJECTN	SUBJECTN	C	100	1	SUBJECT	11
REP-SUBJ	1:1	1	SUBJECTN	SUBJECTN	C	100	1	SUBJECT	11
REP-SUBJ	1:1	2	RNO	RNO	C	8	1	REPORTR	11
REPORTR	01	1	RNO	RNO	C	8	1	REPORTR	11
REPORTR	01	2	TITLE	TITLE	C	200	2	REPORTR	01
REPORTR	01	3	SOURCE-N	SOURCE-N	C	100	3	REPORTR	01
REPORTR	01	4	VOL	VOL	C	8	4	REPORTR	01
REPORTR	01	5	NUMB	NUMB	D	3	5	REPORTR	01
REPORTR	01	6	MONTH	MONTH	D	2	6	REPORTR	01
REPORTR	01	7	YEAR	YEAR	D	2	7	REPORTR	01
REPORTR	01	8	FROM-PAGE	FROM-PAGE	D	5	8	REPORTR	01
REPORTR	01	9	TO-PAGE	TO-PAGE	D	5	9	REPORTR	01
REPORTR	01	10	TOTAL-PAGE	TOTAL-PAGE	D	5	10	REPORTR	01
REPORTR	01	11	NO-OF-AUTHOR	NO-OF-AUTHOR	D	2	11	REPORTR	01
REPORTR	01	12	LANGUAGE	LANGUAGE	C	10	12	REPORTR	01
REPORTR	01	13	WRITTEN-YEAR	WRITTEN-YEAR	D	2	13	REPORTR	01
REPORTR	01	14	COST	COST	D	9	14	REPORTR	01
REPORTR	01	15	RTYPE	RTYPE	C	1	15	REPORTR	01
REPORTR	01	16	RSTATE	RSTATE	C	1	16	REPORTR	01
EMP-REP-LNK	1:1	1	RNO	RNO	C	8	1	REPORTR	11
EMP-REP-LNK	1:1	2	ENO	ENO	D	5	1	EMPLOYEE	11
EMP-REP-LNK	1:1	3	AUTH-NO	AUTH-NO	D	2	1	EMP-REP-LNK	01
PROJ-RE	1:1	1	PNO	PNO	C	8	1	PROJECT	11
PROJ-RE	1:1	2	PRIOR-PR-PNO	PNO	C	8	1	PROJECT	11
ORG-EMP	1:1	1	ORG-NAME	ORG-NAME	C	50	1	ORGANIZATION	01
ORG-EMP	1:1	2	ENO	ENO	D	5	1	EMPLOYEE	11
ORGANIZATION	01	1	ORG-NAME	ORG-NAME	C	50	1	ORGANIZATION	01
ORGANIZATION	01	2	ORG-TYPE	ORG-TYPE	C	10	2	ORGANIZATION	01

ESR SUBJECT (subjectn)

SPONSOR (sname, stype)

PROJECT (pno, pname, proj-type, status-f, syear, eycar, budget)

PRODUCT (pd-no, pd-name, ptype, cost, year, language, psize)

EMPLOYEE(eno, ename, efname, esex, birth-place, birth-year, birth-month, alma-mater, major, degree, deg-year, hired-year, retired-year, dept, position, tel, ext)

REPORTR (rno, title, source-n, vol, numb, month, year, from-page, to-page, total-page, no-of-author, languag written-year, cost, rtype, rstate)

ORGANIZATIOR (org-name, org-type)

RSR SPON-PROJ (sname, pno)

PROJ-PROD(pno, pd-no)

PROJ-EMP-LNK (pno, eno, smonth, eyear, emonth, pposition, role, percentage)

PROJ-SUBJ (pno, subjectn)

REP-SUBJ (subjectn, rno)

EMP-REP-LNK (rno, eno, auth-no)

PROJ-RE (pno, prior-pr-pno)

ORG-EMP (org-name, eno)

PRDBSOLCS

B LISSI

1) RECリレーション

RECORD-TYPE	REC-MODE	DEGREE	CARDINALITY	LOC-MODE	AREA-NAME
SYSTEM	0	0	1	2	
EMPLOYEE	0	17	20	1	RANGEF
PROJECT	0	7	5	1	RANGEA
REPORTR	0	16	26	1	RANGEE
SPONSOR	0	2	1	2	RANGEB
SUBJECT	0	1	86	1	RANGED
PRODUCT	0	8	7	2	RANGEC
ORGANIZATION	0	2	2	2	RANGEI
PROJ-EMP-LNK	1	8	33	2	RANGEG
EMP-REP-LNK	1	1	69	2	RANGEL
PROJ-SUBJ	1	0	24	2	RANGEJ
REP-SUBJ	1	0	149	2	RANGEK
PROJ-RE	1	0	3	2	RANGEH

2) SETリレーション

SET-TYPE	O-REC-TYPE	M-REC-TYPE	CONNECTIVITY	SORT-ATT#	UNIQ-ATT#	LINK-MODE	INSERT	UNI RETENTION
STRUC-OWNR	STRUC-MEMB							
SPON-PROJ	SPONSOR	PROJECT	5000	1	1	1	0	0
	0	0						
PROJ-PROD	PROJECT	PRODUCT	1400	1	1	1	0	0
	0	0						
P-E	PROJECT	PROJ-EMP-LNK	6800	0	0	1	0	0
	0	0						
E-P	EMPLOYEE	PROJ-EMP-LNK	1550	0	0	1	0	0
	0	0						
P-S	PROJECT	PROJ-SUBJ	5600	0	0	1	0	0
	0	0						
S-P	SUBJECT	PROJ-SUBJ	272	0	0	1	0	0
	0	0						
S-R	SUBJECT	REP-SUBJ	1659	0	0	1	0	0
	0	0						
R-S	REPORTR	REP-SUBJ	5538	0	0	1	0	0
	0	0						
R-E	REPORTR	EMP-REP-LNK	2038	0	0	1	0	0
	0	0						
E-R	EMPLOYEE	EMP-REP-LNK	2450	0	0	1	0	0
	0	0						
PRIOR-PROJ	PROJECT	PROJ-RE	600	0	0	1	0	0
	0	0						
POST-PROJ	PROJECT	PROJ-RE	600	0	0	1	0	0
	0	0						
ORG-EMP	ORGANIZATION	EMPLOYEE	10000	0	0	1	0	0
	0	0						
SYS-ORG	SYSTEM	ORGANIZATION	0	0	0	0	0	0
	0	0						
SYS-SPON	SYSTEM	SPONSOR	0	0	0	0	0	0
	0	0						

3) ＩＴＥＭリレーション

RECORD-TYPE	ATT#	ATT-NAME	INDX-MODE	UNIQUENESS	SELECTIVITY	TYPE	LENGTH	DOMAIN
EMPLOYEE	1	ENO	1	1	5001D		5	ENO
EMPLOYEE	2	ENAME	0	0	5001C		10	ENAME
EMPLOYEE	3	EFNAME	0	0	526IC		20	EFNAME
EMPLOYEE	4	ESEX	0	0	50001C		1	ESEX
EMPLOYEE	5	BIRTH-PLACE	0	0	10001C		10	BIRTH-PLACE
EMPLOYEE	6	BIRTH-YEAR	0	0	33331D		2	BIRTH-YEAR
EMPLOYEE	7	BIRTH-MONTH	0	0	33331D		2	BIRTH-MONTH
EMPLOYEE	8	ALMA-MATER	0	0	667IC		30	ALMA-MATER
EMPLOYEE	9	MAJOR	0	0	14291C		30	MAJOR
EMPLOYEE	10	DEGREE	0	0	50001C		1	DEGREE
EMPLOYEE	11	DEG-YEAR	0	0	1667ID		2	DEG-YEAR
EMPLOYEE	12	HIRED-YEAR	0	0	1667ID		2	HIRED-YEAR
EMPLOYEE	13	RETIRED-YEAR	0	0	100001D		2	RETIRED-YEAR
EMPLOYEE	14	DEPT	0	0	50001C		20	DEPT
EMPLOYEE	15	EPOSITION	0	0	33331C		20	EPOSITION
EMPLOYEE	16	TEL	0	0	50001C		12	TEL
EMPLOYEE	17	EXT	0	0	667ID		4	EXT
PROJECT	1	PNO	1	1	20001C		8	PNO
PROJECT	2	PNAME	0	0	20001C		100	PNAME
PROJECT	3	PROJ-TYPE	0	0	5001C		1	PROJ-TYPE
PROJECT	4	STATUS-F	0	0	5001C		3	STATUS-F
PROJECT	5	SYEAR	0	0	2001D		2	SYEAR
PROJECT	6	EYEAR	0	0	2001D		2	EYEAR
PROJECT	7	BUDGET	0	0	10001D		11	BUDGET
REPORTR	1	RNO	1	1	385IC		8	RNO
REPORTR	2	TITLE	0	0	556IC		200	TITLE
REPORTR	3	SOURCE-N	0	0	526IC		100	SOURCE-N
REPORTR	4	VOL	0	0	33331C		8	VOL
REPORTR	5	NUMB	0	0	5001D		3	NUMB
REPORTR	6	MONTH	0	0	2501D		2	MONTH
REPORTR	7	YEAR	0	0	1001D		2	YEAR
REPORTR	8	FROM-PAGE	0	0	625IC		5	FROM-PAGE
REPORTR	9	TO-PAGE	0	0	5001D		5	TO-PAGE
REPORTR	10	TOTAL-PAGE	0	0	769ID		5	TOTAL-PAGE
REPORTR	11	NO-OF-AUTHOR	0	0	1667ID		2	NO-OF-AUTHOR
REPORTR	12	LANGUAGE	0	0	2501C		10	LANGUAGE
REPORTR	13	WRITTEN-YEAR	0	0	2501D		2	WRITTEN-YEAR
REPORTR	14	COST	0	0	2001D		9	COST
REPORTR	15	RTYPE	0	0	5001C		1	RTYPE
REPORTR	16	RSTATE	0	0	33331C		1	RSTATE
SPONSOR	1	SNAME	0	0	100001C		20	SNAME
SPONSOR	2	STYPE	0	0	10001C		10	STYPE
SUBJECT	1	SUBJECTN	1	1	11*1C		100	SUBJECTN
PRODUCT	1	PD-NO	0	0	14291C		5	PD-NO
PRODUCT	2	PD-NAME	0	0	14291C		30	PD-NAME
PRODUCT	3	PTYPE	0	0	100001C		1	PTYPE
PRODUCT	4	COST	0	0	100001D		9	COST
PRODUCT	5	YEAR	0	0	33331D		2	YEAR
PRODUCT	6	LANGUAGE	0	0	33331C		10	LANGUAGE
PRODUCT	7	PSIZE	0	0	100001D		5	PSIZE
PRODUCT	8	OS-NAME	0	0	100001C		15	OS-NAME
ORGANIZATIOR	1	ORG-NAME	0	0	5001C		50	ORG-NAME
ORGANIZATIOR	2	ORG-TYPE	0	0	5001C		10	ORG-TYPE
PROJ-EMP-LNK	1	STATUSS	0	0	5001C		4	STATUSS
PROJ-EMP-LNK	2	SYEAR	0	0	1*291D		2	SYEAR
PROJ-EMP-LNK	3	SMONTH	0	0	5001D		2	SMONTH
PROJ-EMP-LNK	4	EYEAR	0	0	2501D		2	EYEAR
PROJ-EMP-LNK	5	EMONTH	0	0	5001D		2	EMONTH
PROJ-EMP-LNK	6	PPOSITION	0	0	33331C		10	PPOSITION
PROJ-EMP-LNK	7	ROLE	0	0	33331C		10	ROLE
PROJ-EMP-LNK	8	PERCENTAGE	0	0	33331D		3	PERCENTAGE
EMP-REP-LNK	1	AUTH-NO	0	0	2001D		2	AUTH-NO

C. DML I

```
!BLOCK#!STATEMENT#!DML STATEMENT
!-------------------------------------------------------------------------------
!P302  !      1!      **** INIT-CVTL                          (%00)
!P302  !      2!      *     SET CALC VALUES    AT    CVTL.
!P302  !      3!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !      4!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !      5!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !      6!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !      7!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !      8!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !      9!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !     10!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !     11!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !     12!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !     13!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !     14!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !     15!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !     16!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !     17!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !     18!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !     19!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !     20!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !     21!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
!P302  !     22!            ADD 1 TO CCVTL.     MOVE #31 TO CVTL (CCVTL).
                                                 (%00)

!P450  !      1!      **** DML-RESULT OUTPUT
!P450  !      2!           MOVE %00 TO I1.
!P450  !      3!           PERFORM  NEXT1.
!P450  !      4!      *         I = THE COLMN. NUMB. OF SRSLT
!P450  !      5!      *             WHICH %00 IS STORED
!P450  !      6!           IF  MARK IN STDBKEY (1, J)  =  OFFF    GO TO %17.
!P450  !      7!      *         CLEAR MARK
!P450  !      8!           MOVE OFFF TO MARK IN STDBKEY (1, J).
!P450  !      9!      ***       FIND #01  DB-KEY IS DBKEY IN STDBKEY (1, J).
!P450  !     10!           MOVE DBKEY IN STDBKEY (1, J)  TO  PGCS OF FCOM.
!P450  !     11!      ***       GET  #01.
!P450  !     12!           GET  CURRENT.
!P450  !     13!                PERFORM  GCOUNTP.
!P450  !     14!      *    SET RESULT    TO   THE   RESULT   RELATION ( RSLT ).
!P450  !     15!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     16!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     17!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     18!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     19!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     20!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     21!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     22!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     23!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     24!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     25!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     26!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     27!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     28!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     29!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     30!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     31!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     32!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     33!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     34!           MOVE #13 IN ##01 TO #14%00 IN RSLT.
!P450  !     35!      *.
!P450  !     36!      %17.

!P100  !      1!      **** DML-ID
!P100  !      2!       IDENTIFICATION                    DIVISION.
!P100  !      3!       PROGRAM-ID.                       DBTGMDML.
!P100  !      4!       AUTHOR.                           JDD8S.
!P100  !      5!      *
!P100  !      6!      *
!P100  !      7!       ENVIRONMENT                       DIVISION.
!P100  !      8!       CONFIGURATION                     SECTION.
!P100  !      9!       SUBSCHEMA-NAME.                   'PROJSUB1'.
!P100  !     10!       SOURCE-COMPUTER.                  FACOM-M160.
!P100  !     11!       OBJECT-COMPUTER.                  FACOM-M160.
!P100  !     12!      *
!P100  !     13!       INPUT-OUTPUT                      SECTION.
!P100  !     14!       FILE-CONTROL.
!P100  !     15!            SELECT    RSLT-F          ASSIGN    TO    DA-S-U01.
```

```
IP210 | *----------------------------------------------------------------*
IP210 | **** DATA-VSI
IP210 | WORKING-STORAGE          SECTION.
IP210 | *
IP210 |  01  NDTBL.
IP210 |    02  NDTBLO                 OCCURS      *81 TIMES.
IP210 |      03  NPT                  PIC         9(05).
IP210 |      03  NODE-NO              PIC         9(05).
IP210 |  01  CNDTBL                   PIC         9(05)     VALUE      0.
IP210 |  01  MAXNDTBL                 PIC         9(05)     VALUE      *81.
IP210 | *
IP210 | *    SCHEME OF THE RESULT RELATION
IP210 |  01  SRSLT.
IP210 |    02  SRSLTO                 OCCURS      *84 TIMES.
IP210 |      03  NPT                  PIC         9(05).
IP210 |      03  NODE-NO              PIC         9(05).
IP210 |        03  CVPT               PIC         9(05).
IP210 |      03  REC-TYPE             PIC         X(16).
IP210 |      03  ATT-NAME             PIC         X(16).
IP210 |  01  CSRSLT                   PIC         9(05)     VALUE      0.
IP210 |  01  MAXSRSLT                 PIC         9(05)     VALUE      *84.
IP210 |  01  CSR                      PIC         9(05)     VALUE      1.
IP210 | *    TABLE FOR STORING DBKEYS OF SATISFIABLE OCCURRENCES.
IP210 |  01  STDBKEY.
IP210 |      02  STDBKEY1                         OCCURS     *84  TIMES.
IP210 |      03  STDBKEY2                         OCCURS     *83  TIMES.
IP210 |        04  MARK                           PIC        9(02).
IP210 | ***    04  DBKEY                          USAGE IS   DB-KEY.
IP210 |        04  DBKEY                          PIC        X(08).
IP210 | *
IP210 | *    STATUS   VEHICLE FROM   PRECEDING NODE
IP210 |  01  LFOUND                               PIC        9(02).
IP210 | *    THE NUMB. OF ACCESSED OCCURRENCES
IP210 |  01  CNT                                  PIC        9(08)     VALUE 0.
IP210 |  01  GCNT                                 PIC        9(08)     VALUE 0.
IP210 |  01  GCNTO                                PIC        9(08)     VALUE 0.
IP210 | *
IP210 | *    VARIABLES FOR STORING ELAPSE TIME.
IP210 |  01  T1                                   PIC        9(08).
IP210 |  01  T2                                   PIC        9(08).
IP210 |  01  T3                                   PIC        9(08).
IP210 |  01  T4                                   PIC        9(08).
IP210 |  01  T5                                   PIC        9(08).
IP210 | *    CONSTANT
IP210 |  01  FALSE                                PIC        9(02)     VALUE 0.
IP210 |  01  TRUE                                 PIC        9(02)     VALUE 1.
IP210 |  01  ENDD                                 PIC        9(02)     VALUE 0.
IP210 |  01  YES                                  PIC        9(02)     VALUE 1.
IP210 |  01  NOO                                  PIC        9(02)     VALUE 0.
IP210 |  01  ONN                                  PIC        9(02)     VALUE 1.
IP210 |  01  OFFF                                 PIC        9(02)     VALUE 0.
IP210 | *
IP210 | *
IP210 | *    WORKING  VARIABLES
IP210 |  01  I                                    PIC        S9(08).
IP210 |  01  J                                    PIC        S9(08).
IP210 |  01  K                                    PIC        S9(08).
IP210 |  01  L                                    PIC        S9(08).
IP210 |  01  M                                    PIC        S9(08).
IP210 |  01  N                                    PIC        S9(08).
IP210 |  01  II                                   PIC        S9(08).
IP210 |  01  JJ                                   PIC        S9(08).
IP210 |  01  KK                                   PIC        S9(08).
IP210 |  01  PT                                   PIC        9(08).
IP210 |  01  PPT                                  PIC        9(08).
IP210 | *77  DBK                                  USAGE IS   DB-KEY.
IP210 |  01  DBK                                  PIC        X(08).
IP210 | *    NAME OF RECORD-TYPE (REN) AND ITEM (ITN)
IP210 |  01  REN                                  PIC        X(16).
IP210 |  01  ITN                                  PIC        X(16).
IP210 |  01  ERR1                                 PIC        9(03).
IP210 |  01  ERR2                                 PIC        9(02).
IP210 | *
IP210 | *    SCHEME OF THE RESULT
IP210 |  01  PSCHEME                              PIC        X(*86)    VALUE
IP210 |        '*85'.
IP210 | *
IP210 | *    FOR RSLT RELATION
IP210 |  01  CRSLT                                PIC        9(05)     VALUE 0.
IP210 |  01  MAXRSLT                              PIC        9(05)     VALUE *83.
```

```
IP200    1|      **** DATA-FD1
IP200    2|       DATA                        DIVISION.
IP200    3|       FILE                        SECTIUN.
IP200    4|       *
IP200    5|        FD  RSLT-F    LABEL RECORD   IS   STANDARD.
IP200    6|        01  RSLT.
IP201    1|      **** DATA-FD2                          (%00)                  #05
IP201    2|            03  #11%00     PIC  #51(#52).
IP201    3|            03  #11%00     PIC  #51(#52).
IP201    4|            03  #11%00     PIC  #51(#52).
IP201    5|            03  #11%00     PIC  #51(#52).
IP201    6|            03  #11%00     PIC  #51(#52).
IP201    7|            03  #11%00     PIC  #51(#52).
IP201    8|            03  #11%00     PIC  #51(#52).
IP201    9|            03  #11%00     PIC  #51(#52).
IP201   10|            03  #11%00     PIC  #51(#52).
IP201   11|            03  #11%00     PIC  #51(#52).
IP201   12|            03  #11%00     PIC  #51(#52).
IP201   13|            03  #11%00     PIC  #51(#52).
IP201   14|            03  #11%00     PIC  #51(#52).
IP201   15|            03  #11%00     PIC  #51(#52).
IP201   16|            03  #11%00     PIC  #51(#52).
IP201   17|            03  #11%00     PIC  #51(#52).
IP201   18|            03  #11%00     PIC  #51(#52).
IP201   19|            03  #11%00     PIC  #51(#52).
IP201   20|            03  #11%00     PIC  #51(#52).
IP201   21|            03  #11%00     PIC  #51(#52).
```

```
IP211 I        1I      **** DATA-WS2                                (500)
IP211 I        2I      *    TABLE FOR STORING CALC VALUES
IP211 I        3I       01  CVTL0.
IP211 I        4I           02  CVTL1                               OCCURS    #80  TIMES.
IP211 I        5I               03  CVTL                            PIC       #32(#33).
IP211 I        6I       01  MAXCVTL                                 PIC       9(05)   VALUE  #80.
IP211 I        7I       01  CCVTL                                   PIC       9(05)   VALUE  0.
IP211 I        8I      *    PARAMETERS OF GNV.
IP211 I        9I       01  MODEE                                   PIC       9(05).
IP211 I       10I       01  CVAL                                    PIC       #32(#33).
```

```
IP300 I        1I      *-----------------------------------------------------------------
IP300 I        2I      **** DML-INIT
IP300 I        3I       PROCEDURE                                   DIVISION.
IP300 I        4I       DML-M.
IP300 I        5I               READY.
IP300 I        6I               OPEN       OUTPUT          RSLT-F.
IP300 I        7I      *
IP300 I        8I               DISPLAY    PSCHEME.
IP300 I        9I               ACCEPT     TI    FROM  TIME.
IP300 I       10I      *
IP300 I       11I      *    INITIALIZATION
IP300 I       12I      *
IP300 I       13I               MOVE 1     TO   TRUE.      MOVE 0     TO   FALSE.
IP300 I       14I               MOVE 1     TO   DNN.       MOVE 0     TO   OFFF.
IP300 I       15I               MOVE 1     TO   YES.       MOVE 0     TO   NOO.
IP300 I       16I               MOVE 1     TO   ENDD.
IP300 I       17I      *
IP300 I       18I               MOVE            FALSE      TO        LFOUND.
IP300 I       19I      *    SRSLT
IP300 I       20I               MOVE     0     TO   CSRSLT.    MOVE 1     TO   CSR.
IP300 I       21I               MOVE     #84   TO   MAXSRSLT.
IP300 I       22I      *
IP300 I       23I      *
IP300 I       24I      *    RSLT
IP300 I       25I               MOVE     #83   TO   MAXRSLT.
IP300 I       26I      *
IP300 I       27I               MOVE  1  TO  I.
IP300 I       28I       INIT0.
IP300 I       29I               MOVE  0  TO  CVPT IN SRSLT (I).
IP300 I       30I               IF  I  <  MAXSRSLT
IP300 I       31I                   ADD 1 TO I      GO TO INIT0.
IP300 I       32I      *    CLEAR    MARK
IP300 I       33I               MOVE 0     TO   I.
IP300 I       34I       INIT1.  IF   I  =  MAXSRSLT        GO   TO   INIT3.
IP300 I       35I               ADD  1     TO   I.
IP300 I       36I                    MOVE 0     TO   J.
IP300 I       37I       INIT2.       IF   J  =  MAXRSLT    GO   TO   INIT1.
IP300 I       38I                    ADD  1     TO   J.
IP300 I       39I                         MOVE OFFF TO MARK IN STDBKEY (I, J).
IP300 I       40I                                           GO   TO   INIT2.
IP300 I       41I       INIT3.
IP300 I       42I      *    SET NODE-NO IN NDTBL.
IP300 I       43I               MOVE 0     TO   I.
IP300 I       44I       INIT4.  IF   I  =  MAXNDTBL        GO   TO   INIT5.
IP300 I       45I               ADD  1     TO   I.
IP300 I       46I                         MOVE  I  TO  NODE-NO IN NDTBL (I).
IP300 I       47I                                           GO   TO   INIT4.
IP300 I       48I       INIT5.
```

```
IP301 |     **** INIT-SUB                                   ($00)
IP301 |                MOVE FALSE        TO     $20.
IP301 |                ADD  1            TO     CNDTBL.
IP301 |                MOVE   $00   TO   NPT IN NOTBL (CNDTBL).
IP301 |                MOVE   NODE-NO IN NOTBL (CNDTBL)   TO   I.        *05|
IP301 |                MOVE '#01'        TO     REN.                     *05|
IP301 |                MOVE $00          TO     J.                       *05|
IP301 |     *                                                            *05|
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
IP301 |                MOVE '#12' TO ITN.  PERFORM    S-SRSLT.
```

```
IP401 |     *-----------------------------------------------------------*|
IP401 |     **** FIRST                                      ($00)
IP401 |     ****      FIND FIRST      #01         WITHIN  #00 .
IP401 |               FIND FIRST      '#01'       WITHIN '#00' RANGE.
IP401 |               IF    DB-EXCEPTION IS 11   GO TO TERM.
IP401 |               IF    DB-EXCEPTION IS 12   GO TO TERM.
IP401 |                                          GO TO $02.
IP401 |     $01.
IP401 |     ****      FIND NEXT       #01         WITHIN  #00.
IP401 |               FIND NEXT       '#01'       WITHIN '#00' RANGE.
IP401 |               IF    DB-EXCEPTION IS 12   GO TO TERM.
IP401 |               IF    DB-EXCEPTION IS 11   GO TO TERM.
IP401 |     $02.
IP401 |               MOVE $00   TO    PT.
IP401 |                                          PERFORM    COUNTR.
IP401 |     ***       GET   ##01.
IP401 |               GET   CURRENT.                                    *93|
IP401 |                  PERFORM  GCOUNT.                               *93|
IP401 |               IF ( #03 AND #04 )
IP401 |                  NEXT SENTENCE   ELSE GO    TO    $01.          *93|
IP401 |     ***       ACCEPT      $40   FROM      #02 CURRENCY.
IP401 |               MOVE        PGCS OF FCOM    TO    $40.
IP401 |               MOVE        $40   TO    $30.
IP401 |               MOVE        $40   TO    DBK.                      *05|
IP401 |                                          PERFORM    RESULT.     *05|
```

```
IP402 |     *-----------------------------------------------------------*|
IP402 |     **** CN-FIRST AND SN-FIRST              ($04)
IP402 |     ***       FIND FIRST '#01' WITHIN '#00'.
IP402 |               FIND FIRST '#01' WITHIN '#00' RANGE.
IP402 |               IF    DB-EXCEPTION IS 11   GO    TO    TERM.
IP402 |               IF    DB-EXCEPTION IS 12   GO    TO    TERM.
IP402 |                                 GO TO $03.
IP402 |     $01.
IP402 |     ***       FIND #01  DB-KEY IS $30.
IP402 |               MOVE $30   TO   PGCS OF FCOM.
IP402 |               FIND CURRENT.
IP402 |     $02.    FIND NEXT        '#02'   WITHIN '#00' RANGE.
IP402 |               IF    DB-EXCEPTION IS 11   GO    TO    TERM.
IP402 |               IF    DB-EXCEPTION IS 12   GO    TO    TERM.
IP402 |     $03.
IP402 |               MOVE $00   TO    PT.
IP402 |                                          PERFORM    COUNTR.
IP402 |     ***       GET   ##01.
IP402 |               GET   CURRENT.                                    *93|
IP402 |                  PERFORM  GCOUNT.                               *93|
IP402 |               IF ( #03 AND #04 )
IP402 |                  NEXT SENTENCE   ELSE GO    TO    $02.          *93|
IP402 |     *    SATISFIED
IP402 |     ***       ACCEPT      $30   FROM #02   CURRENCY.
IP402 |               MOVE        PGCS OF FCOM    TO    $30.
IP402 |               MOVE        $30   TO DBK.                         *05|
IP402 |                                          PERFORM    RESULT.     *05|
```

```
IP403   |     1|  *-------------------------------------------------------------*
IP403   |     2|  **** D-CALC                              (%00)
IP403   |     3|   %01.
IP403   |     4|                 PERFORM   GNV.                                  #09
IP403   |     5|                 IF   MODEE = ENDD         GO   TO   TERM.
IP403   |     6|                 MOVE      CVAL       TO   #08 IN #01.
IP403   |     7|    %02.
IP403   |     8|                 FIND  ANY  '#01'.
IP403   |     9|                 IF   DB-EXCEPTION IS 13  GO   TO   %01.
IP403   |    10|                                          PERFORM   COUNTR.
IP403   |    11|  ***            GET  #01.
IP403   |    12|                 GET  CURRENT.                                   #94
IP403   |    13|                     PERFORM   GCOUNT.                           #94
IP403   |    14|                 IF   #04   NEXT
IP403   |    15|                      SENTENCE  ELSE       GO   TO   %02.        #94
IP403   |    16|  ***            ACCEPT    %30  FROM #02  CURRENCY.
IP403   |    17|                 MOVE      PGCS OF FCUM   TO   %30.
IP403   |    18|                 MOVE      %00  TO   PT.  MOVE %30  TO DBK.      #05
IP403   |    19|                                          PERFORM   RESULT.      #05
```

```
IP404   |     1|  *-------------------------------------------------------------*
IP404   |     2|  **** A-CALC                              (%00)
IP404   |     3|   %01. MOVE %00 TO PT.
IP404   |     4|                 PERFORM   GNV.                                  #09
IP404   |     5|                 IF   MODEE = ENDD         GO   TO   TERM.
IP404   |     6|                 MOVE      CVAL       TO   #08 IN #01.
IP404   |     7|  ***            FIND      ANY        #01.
IP404   |     8|                 FIND      ANY        '#01'.
IP404   |     9|                 IF   DB-EXCEPTION IS 13  GO   TO   %01.
IP404   |    10|                                          PERFORM   COUNTR.
IP404   |    11|  ***            GET  #01.
IP404   |    12|                 GET  CURRENT.                                   #94
IP404   |    13|                     PERFORM   GCOUNT.                           #94
IP404   |    14|                 IF   #04
IP404   |    15|                      NEXT SENTENCE  ELSE GO   TO   %01.         #94
IP404   |    16|  *              SATISFIED
IP404   |    17|  ***            ACCEPT    %30  FROM #02  CURRENCY.              #05
IP404   |    18|                 MOVE      PGCS OF FCUM   TO   %30.              #05
IP404   |    19|                 MOVE      %30  TO DBK.                          #05
IP404   |    20|                                          PERFORM   RESULT.      #05
```

```
IP411   |     1|  *-------------------------------------------------------------*
IP411   |     2|  **** FIRST CONFLUENT NODE
IP411   |     3|  **** FCN-NEXT                            (%00)
IP411   |     4|   %01.
IP411   |     5|                 IF   %20 = TRUE           GO   TO   %02.
IP411   |     6|                 MOVE      FALSE      TO   LFOUND.
IP411   |     7|                 MOVE      TRUE       TO   %20.      MOVE FALSE TO %21.
IP411   |     8|                                          GO   TO   %03.
IP411   |     9|    %02.
IP411   |    10|                 IF   LFOUND = TRUE        MOVE TRUE TO %21.
IP411   |    11|  ***            FIND      #01  DB-KEY IS %40.
IP411   |    12|                 MOVE      %40  TO   PGCS OF FCUM.
IP411   |    13|                 FIND      CURRENT.
IP411   |    14|    %03.
IP411   |    15|                 FIND NEXT '#01' WITHIN    '#02'.
IP411   |    16|                 IF   DB-EXCEPTION IS 11  GO   TO   %05.
IP411   |    17|                 IF   DB-EXCEPTION IS 12  GO   TO   %05.
IP411   |    18|                 MOVE      %00 TO PT.
IP411   |    19|                                          PERFORM   COUNTR.
IP411   |    20|                 MOVE      %21 TO LFOUND. MOVE FALSE TO %20.
IP411   |    21|                 IF   %21 = TRUE GO TO %04.
IP411   |    22|  *              FAILURE   RETURN    (FRTRN)
IP411   |    23|                 MOVE #60 TO PPT.    PERFORM   FRTRN.
IP411   |    24|                                          GO   TO   #06.
IP411   |    25|  *              SUCCES RETURN
IP411   |    26|    %04.
IP411   |    27|                 MOVE #70 TO PPT.    PERFORM   SRTRN.
IP411   |    28|                                          GO   TO   #07.
IP411   |    29|  *              END  OF   SET  OCCURRENCE.
IP411   |    30|    %05.
IP411   |    31|  ***            ACCEPT    %40  FROM #02  CURRENCY.
IP411   |    32|                 MOVE      PGCS OF FCUM   TO   %40.
IP411   |    33|  ***            GET  #01.
IP411   |    34|                 GET  CURRENT.                                   #93
IP411   |    35|                     PERFORM   GCOUNT.                           #93
IP411   |    36|                 IF ( #03 AND #04 )
IP411   |    37|                      NEXT SENTENCE  ELSE GO   TO   %03.         #93
IP411   |    38|                 MOVE %40 TO %30.
IP411   |    39|                                          MOVE %30 TO DBK.      #05
IP411   |    40|                                          PERFORM   RESULT.     #05
```

```
*----------------------------------------------------------------*
**** CN-NEXT                              (%00)      (#10).
 %01. MOVE %00 TO PT.
          IF   %20 = THUE         GO TO %02.
          MOVE      FALSE       TO    LFOUND.
          MOVE      TRUE TO %20.    MOVE FALSE TO %21.
                                   GO    TO   %03.
 %02.     IF    LFOUND = TRUE       MOVE TRUE TO %21.
***       FIND      #01  DB-KEY     IS %40.
          MOVE      %40  TO PGCS OF FCOM.
          FIND CURRENT.
 %03.
          FIND NEXT '#01' WITHIN '#02'.
          IF  DB-EXCEPTION  IS  13   GO TO %04.
                                     GO TU %05.
 %04.
*     NOT FOUND
          MOVE %21  TO LFOUND. MOVE FALSE TO %20.
          IF    %21 = FALSE   NEXT SENTENCE
                        ELSE        GO    TO    %04.
*     FAILURE  RETURN
          MOVE #60 TO PPT. PERFORM FRTRN.
                                   GO    TO    #06.
*     SUCCES RETURN
          MOVE #70 TO PPT. PERFORM SRTHN.
                                   GU    TU    #07.
*     NOT END
 %05.
***   ACCEPT   %40  FRUH #02  CURRENCY.
          MUVE PGCS OF FCOM   TO    %40.
          IF   %40 = #10  NEXT SENTENCE  ELSE  GO TU %03.
```

```
*----------------------------------------------------------------*
**** SN-NEXT                              (%00)
 %01. MOVE %00 TO PT.
          IF   %20 = THUE  GO TO %02.
          MOVE      FALSE       TO    LFOUND.
          MUVE TRUE TU %20.    MUVE FALSE TO %21.
                               GO    TU    %03.
 %02.
          IF    LFOUND = TRUE       MOVE TRUE TU %21.
***       FIND      #01  DB-KEY IS %40.
          MUVE      %40 TO PGCS OF FCOM.
          FIND      CURRENT.
 %03.
          FIND NEXT '#01' WITHIN '#02'.
          IF   DB-EXCEPTIUN IS 11  GU  TO    %04.
          IF   DB-EXCEPTION IS 12  GU  TO    %04.
                                   GO  TO    %06.
 %04.
          MUVE FALSE TU  %20. MOVE %21 TO LFOUND.
          IF    %21 = TRUE         GU  TO    %05.
*     FAILURE  RETURN
          MUVE #60 TO PPT. PERFORM FRTRN.
                                   GO    TO    #06.
*     SUCCESS  RETURN
 %05.
          MUVE #70 TO PPT. PENFORM SRTHN.
                                   GO    TO    #07.
*     NOT END
 %06.                      PERFORM COUNTR.
***       ACCEPT    %40  FROM #02  CUHRENCY.
          MUVE PGCS UF FCUM    TO    %40.
***       GET       #01.
          GET  CURRENT.                                    #93
               PERFORM  GCOUNT.                            #93
          IF ( #03 AND #04 )
               NEXT SENTENCE  ELSE GO    TO    %03.        #93
          MUVE %40 TU DBK.                                 #05
                                PERFORM    RESULT.         #05
```

```
IP414 | *--------------------------------------------------------------------------*
IP414 | **** NEXT                                    (%00)                         |
IP414 |  %01.                                                                      |
IP414 |             MOVE        %00  TO PT.                                        |
IP414 |             IF    %20 = TRUE    GO TO %02.                                 |
IP414 |             MOVE        FALSE      TO   LFOUND.                            |
IP414 |             MOVE TRUE TO %20.   MOVE FALSE TO %21.                         |
IP414 |                                      GO    TO    %03.                      |
IP414 |  %02.                                                                      |
IP414 |             IF    LFOUND = TRUE  MOVE TRUE TO %21.                         |
IP414 |  %03.                                                                      |
IP414 |             FIND NEXT '#01' WITHIN '#02'.                                  |
IP414 |             IF    DB-EXCEPTION = 11    GO    TO    %04.                    |
IP414 |             IF    DB-EXCEPTION = 12    GO    TO    %04.                    |
IP414 |                                      GO    TO    %06.                      |
IP414 | *     END OF   SET  OCCURENCE                                              |
IP414 |  %04.                                                                      |
IP414 |             MOVE FALSE TO %20.  MOVE %21 TO LFOUND.                        |
IP414 |             IF   %21 = TRUE  GO TO %05.                                    |
IP414 | *     FAILURE  RETURN                                                      |
IP414 |             MOVE #60 TO PPT. PERFORM FRTRN.                                |
IP414 |                                      GO    TO    #06.                      |
IP414 | *     SUCCESS  RETURN                                                      |
IP414 |  %05.                                                                      |
IP414 |             MOVE #70 TO PPT. PERFORM SRTRN.                                |
IP414 |                                      GO    TO    #07.                      |
IP414 | *     NOT END  OF    SET                                                   |
IP414 |  %06.                                PERFORM    COUNTR.                    |
IP414 | ***         ACCEPT     %40 FROM #02 CURRENCY.                              |
IP414 |             MOVE       PGCS OF FCDM    TO %40.                             |
IP414 | ***         GET  #01.                                                      |
IP414 |             GET   CURRENT.                                              #93|
IP414 |                  PERFORM  GCOUNT.                                       #93|
IP414 |             IF ( #03 AND #04 )                                             |
IP414 |                   NEXT SENTENCE  ELSE GO   TO   %03.                     #93|
IP414 |             MOVE %40 TO DBK.                                            #05|
IP414 |                                      PERFORM  RESULT.                   #05|
```

```
IP415 | *--------------------------------------------------------------------------*
IP415 | **** OWNER                                   (%00)                         |
IP415 |  %01. MOVE %00 TO PT.                                                      |
IP415 |             FIND OWNER      WITHIN '#02'.                                  |
IP415 |             IF    DB-EXCEPTION IS 12  GO    TO    %02.                     |
IP415 |                                      GO    TO    %03.                      |
IP415 | *     NOT FOUND                                                            |
IP415 |  %02.                                                                      |
IP415 |             MOVE FALSE       TO   LFOUND.                                  |
IP415 | *     FAILURE  RETURN                                                      |
IP415 |             MOVE #60  TO PPT. PERFORM FRTRN.                               |
IP415 |                                      GO    TO    #06.                      |
IP415 | *     FOUND                                                                |
IP415 |  %03.                                 PERFORM COUNTR.                      |
IP415 | ***         ACCEPT     %40  FROM  #02 CURRENCY.                            |
IP415 |             MOVE       PGCS OF FCDM   TO %40.                              |
IP415 |             IF    %40 = #10 GO TO %04. GO TO %02.                          |
IP415 |  %04.                                                                      |
IP415 | ***         GET   #01.                                                     |
IP415 |             GET   CURRENT.                                              #93|
IP415 |                  PERFORM  GCOUNT.                                       #93|
IP415 |             IF ( #03 AND #04 )                                             |
IP415 |                   NEXT SENTENCE  ELSE GO    TO    %02.                   #93|
IP415 |             MOVE TRUE TO LFOUND.                                           |
IP415 |             MOVE %40   TO DBK.                                          #05|
IP415 |             MOVE       %40  TO   %30.                                      |
IP415 |                                      PERFORM    RESULT.                 #05|
```

```
IP421  |      1| *-----------------------------------------------------------------*
IP421  |      2| **** LAST-NEXT                          (%00)
IP421  |      3|  %01. MOVE %00 TO PI.
IP421  |      4|       IF   %20 = TRUE     GO TO %02.
IP421  |      5|       MOVE FALSE TO  %21. MOVE TRUE TO %20.
IP421  |      6| *
IP421  |      7|  %02.
IP421  |      8|       FIND NEXT '*01'     WITHIN '*02'.
IP421  |      9|       IF   DB-EXCEPTION IS 11  GO   TO   %03.
IP421  |     10|       IF   DB-EXCEPTION IS 12  GO   TO   %03.
IP421  |     11| *    NOT END
IP421  |     12|                                        PERFORM   COUNTR.
IP421  |     13| ***    ACCEPT    %40  FROM *02  CURRENCY.
IP421  |     14|        MOVE      PGCS OF FCOM   TO   %40.
IP421  |     15|        IF  %40 = *10 NEXT SENTENCE ELSE GO TO %02.
IP421  |     16| ***    GET  **01.
IP421  |     17|        GET  CURRENT.                                          *93I
IP421  |     18|            PERFORM  GCOUNT.                                  *93I
IP421  |     19|        IF ( *03 AND *04 )
IP421  |     20|            NEXT SENTENCE  ELSE GO   TO   %02.                *93I
IP421  |     21| *    SATISFIED
IP421  |     22|        MOVE TRUE TO %21.
IP421  |     23|        MOVE %40 TO DBK.                                       *05I
IP421  |     24|                                        PERFORM   RESULT.      *05I
IP421  |     25|                                        GO   TO   %02.
IP421  |     26| *    END OF   SET
IP421  |     27|  %03.
IP421  |     28|        MOVE FALSE     TO %20. MOVE %21 TO LFOUND.
IP421  |     29|        IF   %21 = TRUE  GO TO %04.
IP421  |     30| *    FAILURE  RETURN
IP421  |     31|        MOVE *60 TO PPT.  PERFORM FRTRN.
IP421  |     32|                                        GO   TO   *06.
IP421  |     33| *    SUCCESS  RETURN
IP421  |     34|  %04.
IP421  |     35|        MOVE *70 TO PPT. PERFORM SRTRN.
IP421  |     36|                                        GO   TO   *07.
```

```
IP422  |      1| *-----------------------------------------------------------------*
IP422  |      2| **** LAST-OWNER                         (%00)
IP422  |      3|  %01. MOVE %00 TO PT.
IP422  |      4|       FIND OWNER      WITHIN   '*02'.
IP422  |      5|       IF   DB-EXCEPTION IS 12  GO   TO   %02.
IP422  |      6|                                        GO   TO   %03.
IP422  |      7| *    NOT FOUND.
IP422  |      8|  %02.
IP422  |      9|       MOVE FALSE TO LFOUND.
IP422  |     10| *    FAILURE  RETURN.
IP422  |     11|       MOVE *60 TO PPT. PERFORM FRTRN.
IP422  |     12|                                        GO   TO   *06.
IP422  |     13| *    FOUND
IP422  |     14|  %03.                                  PERFORM COUNTR.
IP422  |     15| ***    ACCEPT    %40  FROM *02  CURRENCY.
IP422  |     16|        MOVE      PGCS OF FCOM   TO   %40.
IP422  |     17|        IF  %40 = *10  NEXT SENTENCE  ELSE  GO TO %02.
IP422  |     18|        GET  CURRENT.                                          *93I
IP422  |     19|            PERFORM  GCOUNT.                                  *93I
IP422  |     20|        IF ( *03 AND *04 )
IP422  |     21|            NEXT SENTENCE  ELSE GO   TO   %02.                *93I
IP422  |     22| *    SATISFIED.
IP422  |     23|        MOVE TRUE TO   LFOUND.
IP422  |     24|        MOVE %40 TO DBK. PERFORM RESULT.                       *05I
IP422  |     25| *    SUCCESS  RETURN
IP422  |     26|        MOVE  *70  TO  PPT.      PERFORM  SRTRN.
IP422  |     27|                                        GO   TO   *07.
```

```
IP431  |      1| *-----------------------------------------------------------------*
IP431  |      2| **** THE ACCESS TREE CONSISTS OF ONLY ONE NODE.
IP431  |      3| **** 1-FIRST                            (%00)
IP431  |      4|       FIND  FIRST  '*01'  WITHIN  '*00'  RANGE.
IP431  |      5|       IF  DB-EXCEPTION  IS  11  GO  TO  TERM.
IP431  |      6|       IF  DB-EXCEPTION  IS  12  GO  TO  TERM.
IP431  |      7|                       MOVE  %00  TO  PI.
IP431  |      8|                                        GO   TO   %02.
IP431  |      9|  %01.
IP431  |     10|       FIND  NEXT  '*01'  WITHIN  '*00'  RANGE.
IP431  |     11|       IF   DB-EXCEPTION IS 11  GO   TO   TERM.
IP431  |     12|       IF   DB-EXCEPTION IS 12  GO   TO   TERM.
IP431  |     13|  %02.
IP431  |     14|                                        PERFORM   COUNTR.
IP431  |     15| *      GET  **01.
IP431  |     16|        GET  CURRENT.                                          *93I
IP431  |     17|            PERFORM  GCOUNT.                                  *93I
IP431  |     18|        IF ( *03 AND *04 )
IP431  |     19|            NEXT SENTENCE  ELSE GO   TO   %01.                *93I
IP431  |     20| ***    ACCEPT    %40  FROM *02 CURRENCY.
IP431  |     21|        MOVE      PGCS OF FCOM   TO   %40.
IP431  |     22|        MOVE      %40  TO DBK.                                 *05I
IP431  |     23|                                        PERFORM   RESULT.      *05I
IP431  |     24|        MOVE  PT  TO  PPI.      PERFORM  SRTRN.               *05I
IP431  |     25|                                        GO   TO   %01.
```

```
IP432  |      1|      *-------------------------------------------------------------*
IP432  |      2|      ****
IP432  |      3|      **** I-A-CALC                          (%00)
IP432  |      4|       %01. MOVE %00 TO PT.
IP432  |      5|       %02.    PERFORM   GNV.                                  #09|
IP432  |      6|               IF   MODEE = ENDD          GO   TO   TERM.
IP432  |      7|               MOVE CVAL TO   #08 IN #01.
IP432  |      8|               FIND ANY  '#01'.
IP432  |      9|               IF   DB-EXCEPTION IS 13  GO   TO   %02.
IP432  |     10|                                         PERFORM   COUNTR.
IP432  |     11|      ***      GET  #01.
IP432  |     12|               GET  CURRENT.                                    #94|
IP432  |     13|                   PERFORM  GCOUNT.                            #94|
IP432  |     14|               IF   #04                                          |
IP432  |     15|                    NEXT SENTENCE  ELSE GO   TO   %02.        #94|
IP432  |     16|      ***      ACCEPT    DBK  FROM #02  CURRENCY.               #05|
IP432  |     17|               MOVE      PGCS OF FCOM   TO   DBK.              #05|
IP432  |     18|                                         PERFORM   RESULT.     #05|
IP432  |     19|               MOVE  PT  TO  PPT.       PERFORM  SRTRN.       #05|
IP432  |     20|                                         GO   TO   %02.
```

```
IP433  |      1|      *-------------------------------------------------------------*
IP433  |      2|      **** I-D-CALC                          (%00)
IP433  |      3|       %01. MOVE %00 TO PT.
IP433  |      4|       %03.    PERFORM   GNV.
IP433  |      5|               IF   MODEE = ENDD          GO   TO   TERM.
IP433  |      6|               MOVE CVAL TO   #08 IN #01.
IP433  |      7|       %02.
IP433  |      8|               FIND DUPLICATE '#01'.
IP433  |      9|               IF   DB-EXCEPTION IS 13  GO   TO   %03.
IP433  |     10|      *        GET  #01.
IP433  |     11|               GET  CURRENT.                                    #94|
IP433  |     12|                   PERFORM  GCOUNT.                            #94|
IP433  |     13|               IF   #04                                          |
IP433  |     14|                    NEXT SENTENCE  ELSE GO   TO   %02.        #94|
IP433  |     15|      ***      ACCEPT    DBK  FROM #02  CURRENCY.               #05|
IP433  |     16|               MOVE      PGCS OF FCOM   TO   DBK.              #05|
IP433  |     17|                                         PERFORM   RESULT.     #05|
IP433  |     18|               MOVE  PT  TO  PPT.       PERFORM  SRTRN.       #05|
IP433  |     19|                                         GO   TO   %02.
```

```
IP601  |      1|      *-------------------------------------------------------------*
IP601  |      2|      *                                                             |
IP601  |      3|      *    GNV        CVAL =    THE CALC  VALUE                    |
IP601  |      4|      *               MOVE =    END       IF CVAL IS ACQUIRED      |
IP601  |      5|      *                         NO        OTHERWISE                |
IP601  |      6|      *                                                             |
IP601  |      7|       GNV            SECTION.
IP601  |      8|                IF   CCVTL NOT > 0        GO   TO   GNV1.
IP601  |      9|                MOVE CVTL (CCVTL)   TO   CVAL.
IP601  |     10|                MOVE NUD            TO   MODEE.
IP601  |     11|                COMPUTE   CCVTL     =    CCVTL - 1.
IP601  |     12|                                         GO   TO   GNV2.
IP601  |     13|      *    CCVTL =< 0
IP601  |     14|       GNV1.    MOVE ENDD           TO   MODEE.
IP601  |     15|       GNV2.    EXIT.
```

```
IP602  |      1|      *-------------------------------------------------------------*
IP602  |      2|      *                                                             |
IP602  |      3|      *    RESULT   PT   = THE POINTER  TO THE NODE                 |
IP602  |      4|      *             DBK  = DB-KEY                                   |
IP602  |      5|      *                                                             |
IP602  |      6|       RESULT         SECTION.
IP602  |      7|                MOVE 0    TO   K.
IP602  |      8|       RESULT1.
IP602  |      9|                IF   K    =    CSRSLT  GO   TO   RESULT2.
IP602  |     10|                ADD  1    TO   K.
IP602  |     11|                IF   NPT IN SRSLT (K)  =  PT  NEXT SENTENCE
IP602  |     12|                                         ELSE GO   TO   RESULT1.
IP602  |     13|                COMPUTE  L  =  CVPT IN SRSLT (K)  +  1.
IP602  |     14|                MOVE  ONN   TO   MARK   IN STOBKEY (K, L).
IP602  |     15|                MOVE  DBK   TO   DBKEY  IN STOBKEY (K, L).
IP602  |     16|                MOVE  L     TO   CVPT   IN SRSLT   (K).
IP602  |     17|                                         GO   TO   RESULT3.
IP602  |     18|       RESULT2.     MOVE 602 TO ERR1.  MOVE  1 TO ERR2.  PERFORM ERRP.
IP602  |     19|      *    RETURN   TO   CALLER.
IP602  |     20|       RESULT3.     EXIT.
```

```
*--------------------------------------------------------------*
*
*   FRTRN   PT   THE POINTER TO THE CURRENT NODE
*           PPT  THE POINTER TO THE FAILURE RETURNED NODE
*
FRTRN           SECTION.
*    SEARCH     NOTBL
     MOVE       0    TO   I.
FRTRN1.
     IF   I     = MAXNDTBL          GO   TO   FRTRN9.
     ADD 1      TO   I.
         IF   NPT IN NOTBL (I)  =  PPT   NEXT SENTENCE
                              ELSE GO    TO   FRTRN1.
              MOVE   NODE-NO IN NOTBL (I)  TO  K.
*
     COMPUTE  II = I + 1.   COMPUTE   J =  MAXNDTBL + 1.
FRTRN2.
     IF   J     = II                GO   TO   FRTRN10.
     COMPUTE  J    =    J - 1.
         IF   NPT IN NOTBL (J)  =  PT    NEXT SENTENCE
                              ELSE GO    TO   FRTRN2.
              MOVE   NODE-NO IN NDTBL (J)  TO  L.
         IF   K > NODE-NO IN SRSLT (CSRSLT)  GO TO FRTRN15.
*    K   -= THE FIRST COLOMN NUMB. OF THE FAILURE RETURNED NODE.
*    L   = THE LAST  COLOMN NUMB. OF THE CURRENT NODE.
*
     MOVE      0    TO    I.
FRTRN3.
     IF   I     =    CSRSLT         GO   TO   FRTRN11.
     ADD 1      TO   I.
         IF   NODE-NO IN SRSLT (I)  <  K      GO TO FRTRN3.
         COMPUTE  M = CVPT IN SRSLT (I)  -  1.
*****
         COMPUTE  N   =   I - 1.
FRTRN4.  IF    N      = CSRSLT          GO    TO    FRTRN15.
         ADD   1     TO    N.
         IF   NODE-NO IN SRSLT (N)  >  L      GO TO FRTRN15.
*
              COMPUTE  II = CVPT IN SRSLT (N) + 1.
              COMPUTE    JJ = M - 1.
         IF   JJ = 0     GO TO FRTRN15.
*        CLEAR      MARK.
FRTRN5.       IF  II  =  JJ              GO TO FRTRN6.
              COMPUTE    II = II - 1.
              MOVE  OFFF  TO  MARK  IN STDBKEY (N, II).
                                        GO    TO    FRTRN5.
FRTRN6.      MOVE  M  TO  CVPT IN SRSLT (N).
                                        GO    TO    FRTRN4.
*****
*    ERROR      STOP
FRTRN9.        MOVE 603 TO ERR1.  MOVE  9 TO ERR2.  PERFORM ERRP.
FRTRN10.       MOVE 603 TO ERR1.  MOVE 10 TO ERR2.  PERFORM ERRP.
FRTRN11.       MOVE 603 TO ERR1.  MOVE 11 TO ERR2.  PERFORM ERRP.
FRTRN12.       MOVE 603 TO ERR1.  MOVE 12 TO ERR2.  PERFORM ERRP.
*    RETURN     TO    CALLER
FRTRN15.                   EXIT.
```

```
IP604  *-----------------------------------------------------------------*
IP604  *
IP604  *     SRTRN    PT : THE POINTER TO THE CURRENT NODE.
IP604  *              PPT: THE POINTER TO THE SUCCESS RETURNED NODE
IP604  *
IP604   SRTRN         SECTION.
IP604          MOVE 0     TO    I.
IP604   SRTRN1. IF    I =  MAXNDTBL        GO   TO   SRTRN9.
IP604          ADD  1     TO    I.
IP604          IF   NPT IN NDTBL (I)  =  PPT       NEXT SENTENCE
IP604                                ELSE GO   TO   SRTRN1.
IP604              MOVE  NODE-NO IN NDTBL (I)  TO  K.
IP604  *
IP604          COMPUTE    I     =     MAXNDTBL   +    1.
IP604   SRTRN2. IF    I     =    1          GO   TO   SRTRN10.
IP604          COMPUTE    I     =     I     - 1.
IP604          IF   NPT IN NDTBL (I)  =  PT        NEXT SENTENCE
IP604                                ELSE GO   TO   SRTRN2.
IP604              MOVE  NODE-NO IN NDTBL (I)  TO  L.
IP604  *
IP604          IF    K     >     L          GO   TO   SRTRN15.
IP604  *
IP604  *     K    THE FIRST COLLUMN NUMB. OF THE SUCCESS RETURNED NODE.
IP604  *     L    THE LAST  COLLUMN NUMB. OF THE CURRENT NODE.
IP604  *
IP604          COMPUTE    I     =     CSRSLT     + 1.
IP604   SRTRN3. IF    I     = 1             GO   TO   SRTRN15.
IP604          COMPUTE    I     =     I - 1.
IP604          IF   NODE-NO IN SRSLT (I)  >  L       GO TO SRTRN3.
IP604              MOVE  CVPT IN SRSLT (I)  TO  M.
IP604  *             SAVE I
IP604          MOVE I TO II.
IP604  *
IP604          ADD 1 TO I.       MOVE 0 TO KK.
IP604   SRTRN4.
IP604          SUBTRACT 1 FROM I.
IP604          IF   NODE-NO IN SRSLT (I)  <  K      GO  TO  SRTRN6.
IP604          IF   CVPT IN SRSLT (I)  >  M
IP604                  MOVE CVPT IN SRSLT (I) TO M.
IP604          IF   I  >  1   GO   TO   SRTRN4.
IP604          MOVE  1  TO  KK.
IP604  *
IP604   SRTRN6.
IP604          IF   I  >  II         GO TO SRTRN7.
IP604               MOVE  M  TO  CVPT IN SRSLT (I).
IP604             ADD 1 TO I.   GO TO SRTRN6.
IP604  *            SUCCESS RETURN TO ROOT NODE
IP604  *                OUTPUT
IP604   SRTRN7.
IP604          IF   KK  NOT =  1    GO TO SRTRN15.
IP604  *     SET THE LOWEST TUPLE NUMB. TO EVERY CVPT.
IP604          MOVE CSRSLT     TO    I.
IP604   SRTRN80.     MOVE CVPT IN SRSLT (I)    TO    M.
IP604          COMPUTE    I     =     CSRSLT + 1.
IP604   SRTRN82.     IF    I = 1          GO    TO    SRTRN88.
IP604          SUBTRACT  1    FROM I.
IP604          IF   CVPT IN SRSLT (I)  >     M     GO TO SRTRN80.
IP604              MOVE M     TO    CVPT IN SRSLT (I).
IP604                                  GO    TO    SRTRN82.
IP604  *    OUTPUT
IP604   SRTRN88.
IP604          PERFORM  OUTPT.
IP604             GO TO SRTRN15.
IP604  *     ERROR      STOP
IP604   SRTRN9.       MOVE 604 TO ERR1.  MOVE  9 TO ERR2.  PERFORM ERRP.
IP604   SRTRN10.      MOVE 604 TO ERR1.  MOVE 10 TO ERR2.  PERFORM ERRP.
IP604   SRTRN11.      MOVE 604 TO ERR1.  MOVE 11 TO ERR2.  PERFORM ERRP.
IP604   SRTRN12.      MOVE 604 TO ERR1.  MOVE 12 TO ERR2.  PERFORM ERRP.
IP604  *     RETURN    TO    CALLER.
IP604   SRTRN15.               EXIT.
```

```
IP605  *-----------------------------------------------------------------*
IP605   ACCOUNT  SECTION.
IP605  *
IP605  *     COUNT
IP605  *
IP605   COUNTR. ADD  1      TO    CNT.
IP605   COUNTO. EXIT.
IP606  *
IP606  *    GCOUNT
IP606  *
IP606   GCOUNT. ADD   1     TO     GCNTO.
IP606   GCOUNTO.        EXIT.
IP606  *
IP606  *     OUTPUT COUNT
IP606  *
IP606   GCOUNTP.  ADD 1 TO GCNT.
IP606   GCOUNTP-EXIT. EXIT.
```

```
IP608 |      1| *-----------------------------------------------------------------*
IP608 |      2| *
IP608 |      3| *    S-SRSLT          |    NODE NUMBER
IP608 |      4| *                     REN  RECORD-TYPE
IP608 |      5| *                     ITN  RESULT ATTRIBUTE NAME
IP606 |      6| *                     J    POINTER TO THE VALUE
IP608 |      7| *
IP608 |      8|  S-SRSLT              SECTION.
IP608 |      9|                       ADD  1    TO   CSRSLT.
IP608 |     10|                       IF  CSRSLT  >  MAXSRSLT    GO TO SRSLT2.
IP606 |     11| *
IP608 |     12|                       MOVE  I      TO  NODE-NO   IN SRSLT (CSRSLT).
IP608 |     13|                       MOVE  REN    TO  REC-TYPE  IN SRSLT (CSRSLT).
IP608 |     14|                       MOVE  ITN    TO  ATT-NAME  IN SRSLT (CSRSLT).
IP608 |     15|                       MOVE  J      TO  NPT       IN SRSLT (CSRSLT).
IP608 |     16|                       MOVE  0      TO  CVPI      IN SRSLT (CSRSLT).
IP608 |     17|                             GO TO SRSLT3.
IP608 |     18|       SRSLT2.    MOVE 606 TO ERR1.  MOVE  1 TO ERR2.  PERFORM ERRP.
IP608 |     19|       SRSLT3. EXIT.
```

```
IP212 |      1|       **** DATA-WS3                          (300)
IP212 |      2|       01  %%20                    PIC  9(08).
IP212 |      3|       01  %%21                    PIC  9(08).
IP212 |      4|       01  %%22                    PIC  9(08).
IP212 |      5|       01  %%23                    PIC  9(08).
IP212 |      6|       01  %%24                    PIC  9(08).
IP212 |      7|       01  %%25                    PIC  9(08).
IP212 |      8|       01  %%26                    PIC  9(08).
IP212 |      9|       01  %%27                    PIC  9(08).
IP212 |     10|       01  %%28                    PIC  9(08).
IP212 |     11|       01  %%29                    PIC  9(08).
IP212 |     12| *                                 USAGE IS  DBKEY.
IP212 |     13|       01  %%30                    PIC  X(08).
IP212 |     14|       01  %%31                    PIC  X(08).
IP212 |     15|       01  %%32                    PIC  X(08).
IP212 |     16|       01  %%33                    PIC  X(08).
IP212 |     17|       01  %%34                    PIC  X(08).
IP212 |     18|       01  %%35                    PIC  X(08).
IP212 |     19|       01  %%36                    PIC  X(08).
IP212 |     20|       01  %%37                    PIC  X(08).
IP212 |     21|       01  %%38                    PIC  X(08).
IP212 |     22|       01  %%39                    PIC  X(08).
IP212 |     23|       01  %%40                    PIC  X(08).
```

```
IP501 |      1| *-----------------------------------------------------------------*
IP501 |      2| * TERMINATION
IP501 |      3|  TERM.
IP501 |      4|                       ACCEPT    T2    FROM  TIME.
IP501 |      5| *    GET THE RECORD OCCURRENCE USING THE DB-KEY
IP501 |      6| *        IN THE RSLTO.
IP501 |      7| *
IP502 |      1|                       ACCEPT    T3    FROM  TIME.
IP502 |      2|                       FINISH.
IP502 |      3|                       CLOSE  RSLT-F.
IP502 |      4| *    PRINT ACCOUNTING.
IP502 |      5|                       DISPLAY 'THE NUMBER OF LOCATED  OCCURRENCES = '  CNT.
IP502 |      6|                       DISPLAY 'THE NUMBER OF ACCESSED OCCURRENCES = '  GCNT0.
IP502 |      7|                       DISPLAY 'THE NUMBER OF OUTPUT   OCCURRENCES = '  GCNT.
IP502 |      8|                            COMPUTE  T4 = T2 - T1.
IP502 |      9|                            COMPUTE  T5 = T3 - T2.
IP502 |     10|                       DISPLAY 'THE ELAPSE TIME FOR ACCESSING = '  T4.
IP502 |     11|                       DISPLAY 'THE ELAPSE TIME FOR OUTPUT    = '  T5.
IP502 |     12|                            ADD  T4  TO  T5.
IP502 |     13|                       DISPLAY 'THE TOTAL TIME                = '  T5.
IP502 |     14|                       STOP RUN.
```

```
IP615 |      1| *-----------------------------------------------------------------*
IP615 |      2| *
IP615 |      3| *    NEXT1     INPUT  I = THE NODE#
IP615 |      4| *              OUTPUT  J = THE COLUMN OF SRSLT
IP615 |      5| *                          IN WHICH II IS STORED
IP615 |      6|  NEXT1         SECTION.
IP615 |      7|  NEXT0.
IP615 |      8|                ADD 1 TO J.
IP615 |      9|                IF  NPT IN SRSLT (J)  NOT =  I
IP615 |     10|                                     GO TO NEXT0.
IP615 |     11|  NEXT1.  EXIT.
```

```
IP610   1         11      ERRP          SECTION.
IP610   1         21      ERRP1.
IP610   1         31          DISPLAY 'PATTERN NUMB. = ' ERR1 ', ERROR CODE = ' ERR2.
IP610   1         41          STOP RUN.
IP610   1         51      ERR-EXIT.     EXIT.
```

```
IP611   1         11      *---------------------------------------------------------------*
IP611   1         21      *                                                               *
IP611   1         31      *              OUTPUT    PRINT                                  *
IP611   1         41      *                                                               *
IP611   1         51       OUTPT         SECTION.
IP611   1         61       OUTPT0.
IP611   1         71           MOVE 0  TO I J.
IP611   1         81       OUTPT1. ADD 1 TO J.
```

```
IP612   1         11           DISPLAY  RSLT.
IP612   1         21           WRITE  RSLT  INVALID KEY
IP612   1         31                    DISPLAY  'RESULT FILE OVERFLOW'
IP612   1         41                    GO TO OUTPT10.
IP612   1         51           MOVE  SPACE  TO  RSLT.
IP612   1         61           IF  J  NOT <  CVPT IN SRSLT (1)
IP612   1         71                                    GO TO OUTPT10.
IP612   1         81           MOVE  0  TO  I.
IP612   1         91           GO  TO  OUTPT1.
IP612   1        101       *        END OF OUTPUT
IP612   1        111       OUTPT10.   MOVE 0 TO I.
IP612   1        121       OUTPT11.   ADD  1 TO I.
IP612   1        131                  IF  I  >  CSRSLT  GO TO OUTPT12.
IP612   1        141           MOVE  0  TO  CVPT IN SRSLT (I).
IP612   1        151                  GO TO OUTPT11.
IP612   1        161       OUTPT12.
IP612   1        171                  EXIT.
```

付記Ⅲ　問合せ変換システムの例

1. 例

TEST V081

view の定義(1)

```
#00020 RANGE    ( E, EMPLOYEE ) ( J, PROJECT ) ;
#00030 RANGE    ( JE, PROJ-EMP-LNK ) ;
#00040 DEFINE   DOBS-EMP ( E.ENO, NAME=E.EFNAME, JE.PPOSITION, JE.ROLE )
#00050 WHERE
#00060      E.ENO          = JE.ENO          AND  JE.PNO          = J.PNO          AND
#00070      J.PNAME        = "DISTRIBUTED DATABASE SYSTEMS" ;
```

view の定義(2)

```
#00080 RANGE    ( E, EMPLOYEE ) ( R, REPORTR ) ( S, SUBJECT ) ;
#00090 RANGE    ( ER, EMP-REP-LNK ) ( RS, REP-SUBJ ) ;
#00100 DEFINE   DBS-REP ( NAME=E.EFNAME, R.TITLE, R.YEAR )
#00110 WHERE
#00120      E.ENO          = ER.ENO          AND  ER.RNO          = R.RNO          AND
#00130      R.RNO          = RS.RNO          AND
#00140      RS.SUBJECTN    = "FOUR-SCHEMA STRUCTURE";
```

問合せ

```
#00150 RANGE    ( DE, DOBS-EMP ) ( DR, DBS-REP ) ;
#00160 RETRIEVE INTO TESTV001 ( DE.NAME, DR.TITLE )
#00170 WHERE
#00180      DE.NAME        = DR.NAME;
```

viewに対する問合せ

```
RETRIEVE INTO TESTV001 ( NAME=DE.NAME , TITLE=DR.TITLE ) WHERE DE.
   NAME = DR.NAME ;
RETRIEVE INTO TESTV001 ( NAME=E.EFNAME , TITLE=R.TITLE ) WHERE E.
EENAME = E.EFNAME AND E.ENO = ER.ENO AND ER.RNO = R.RNO
    AND R.RNO  = RS.RNO  AND RS.SUBJECTN  = "FOUR-SCHEMA STRU
CTURE" AND E.ENO  = JE.ENO  AND JE.PNO  = J.PNO  AND J.PNAME
    = "DISTRIBUTED DATABASE SYSTEMS" ;
```

QM後のLCSリレーションに対する問合せ

COBOLプログラム

```
000001  010001**** DML-ID
000002  010002 IDENTIFICATION                    DIVISION.
000003  010003 PROGRAM-ID.                       DBTGMDML.
000004  010004 AUTHOR.                          JDBS.
000005  010005*
000006  010006*
000007  010007 ENVIRONMENT                       DIVISION.
000008  010008 CONFIGURATION                     SECTION.
000009  010009 SUBSCHEMA-NAME.                   'PROJSUB1'.
000010  010010 SOURCE-COMPUTER.                  FACOM-M160.
000011  010011 OBJECT-COMPUTER.                  FACOM-M160.
000012  010012*
000013  010013 INPUT-OUTPUT                      SECTION.
000014  010014 FILE-CONTROL.
000015  010015     SELECT  RSLT-F      ASSIGN  ID  DA-S-U01.
000016  020001**** DATA-FD1
000017  020002 DATA                  DIVISION.
000018  020003 FILE                  SECTION.
000019  020004*
000020  020005 FD  RSLT-F   LABEL RECORD  IS  STANDARD.
000021  020006 01  RSLT.
000022  020111**** DATA-FD2                    (0159)
000023  020112      03  TITLE0159    PIC  X(200).
000024  020123**** DATA-FD2                    (0161)
000025  020124      03  EFNAME0161   PIC  X(020).
000026  021007*-------------------------------------------------------
000027  021008**** DATA-WS1
000028  021009 WORKING-STORAGE       SECTION.
000029  021010*
000030  021011 01  NDTBL.
000031  021012     02  NDTBL0          OCCURS    007 TIMES.
000032  021013         03  NPT         PIC       9(05).
000033  021014         03  NODE-NO     PIC       9(05).
000034  021015 01  CNDTBL              PIC       9(05)     VALUE     0.
000035  021016 01  MAXNDTBL            PIC       9(05)     VALUE     007.
000036  021017*
000037  021018*    SCHEME OF THE RESULT RELATION
000038  021019 01  SRSLT.
000039  021020     02  SRSLT0          OCCURS    30 TIMES.
000040  021021         03  NPT         PIC       9(05).
000041  021022         03  NODE-NO     PIC       9(05).
000042  021023             03  CVPT    PIC       9(05).
000043  021024         03  REC-TYPE    PIC       X(16).
000044  021025         03  ATT-NAME    PIC       X(16).
000045  021026 01  CSRSLT              PIC       9(05)     VALUE     0.
000046  021027 01  MAXSRSLT            PIC       9(05)     VALUE     30.
000047  021028 01  CSR                 PIC       9(05)     VALUE     1.
000048  021029*    TABLE FOR STORING DBKEYS OF SATISFIABLE OCCURRENCES.
000049  021030 01  STDBKEY.
000050  021031     02  STDBKEY1        OCCURS    30  TIMES.
000051  021032     03  STDBKEY2        OCCURS    00100  TIMES.
000052  021033         04  MARK        PIC       9(02).
000053  021034***      04  DBKEY       USAGE IS  DB-KEY.
000054  021035         04  DBKEY       PIC       X(08).
000055  021036*
000056  021037*    STATUS   VEHICLE FROM   PRECEDING NODS
000057  021038 01  LFOUND              PIC       9(02).
000058  021039*    THE NUMB. OF ACCESSED OCCURRENCES
000059  021040 01  CNT                 PIC       9(08)     VALUE 0.
000060  021041 01  GCNT                PIC       9(08)     VALUE 0.
000061  021042 01  GCNT0               PIC       9(08)     VALUE 0.
000062  021043*
000063  021044*    VARIABLES FOR STORING ELAPSE TIME.
000064  021045 01  T1                  PIC       9(08).
000065  021046 01  T2                  PIC       9(08).
000066  021047 01  T3                  PIC       9(08).
000067  021048 01  T4                  PIC       9(08).
000068  021049 01  T5                  PIC       9(08).
000069  021050*    CONSTANT
000070  021051 01  FALSE               PIC       9(02)     VALUE 0.
000071  021052 01  TRUE                PIC       9(02)     VALUE 1.
000072  021053 01  ENDD                PIC       9(02)     VALUE 0.
000073  021054 01  YES                 PIC       9(02)     VALUE 1.
000074  021055 01  NOO                 PIC       9(02)     VALUE 0.
000075  021056 01  ONN                 PIC       9(02)     VALUE 1.
000076  021057 01  OFFF                PIC       9(02)     VALUE 0.
000077  021058*
000078  021059*
000079  021060*    WORKING  VARIABLES
000080  021061 01  I                   PIC       S9(08).
000081  021062 01  J                   PIC       S9(08).
000082  021063 01  K                   PIC       S9(08).
000083  021064 01  L                   PIC       S9(08).
000084  021065 01  M                   PIC       S9(08).
000085  021066 01  N                   PIC       S9(08).
000086  021067 01  II                  PIC       S9(08).
000087  021068 01  JJ                  PIC       S9(08).
000088  021069 01  KK                  PIC       S9(08).
000089  021070 01  PT                  PIC       9(08).
000090  021071 01  PPT                 PIC       9(08).
000091  021072-77  DBK                 USAGE IS  DB-KEY.
000092  021073 01  DBK                 PIC       X(08).
000093  021074*    NAME OF RECORD-TYPE (REN) AND ITEM (ITN)
000094  021075 01  REN                 PIC       X(16).
000095  021076 01  ITN                 PIC       X(16).
000096  021077 01  ERR1                PIC       9(03).
000097  021078 01  ERR2                PIC       9(02).
000098  021079*
```

COBOLプログラム

```
000099  021080*     SCHEME OF THE RESULT
000100  021081 01   PSCHEME                          PIC       X(029)    VALUE
000101  021082       '( (TITLE (FFNAME ( ) ) ) ).
000102  021083*
000103  021084*     FOR RSLT RELATION
000104  021085 01   CRSLT                            PIC       9(05)     VALUE 0.
000105  021086 01   MAXRSLT                          PIC       9(05)     VALUE 00100.
000106  021087**** DATA-WS3                          (0157)
000107  021088 01   L0120                    PIC 9(08).
000108  021089*                              USAGE IS  DBKEY.
000109  021090 01   L0130                    PIC X(08).
000110  021091**** DATA-WS2                          (0157)
000111  021092*     TABLE FOR STORING CALC VALUES
000112  021093 01   CVTL0.
000113  021094       02  CVTL1                                OCCURS   01  TIMES.
000114  021095           03  CVTL                    PIC       X(100).
000115  021096 01   MAXCVTL                          PIC       9(05)     VALUE  01.
000116  021097 01   CCVTL                            PIC       9(05)     VALUE  0.
000117  021098*     PARAMETERS OF GNV.
000118  021099 01   MODEE                            PIC       9(05).
000119  021100 01   CVAL                             PIC       X(100).
000120  021101**** DATA-WS3                          (0158)
000121  021102 01   L0220                    PIC 9(08).
000122  021103 01   L0221                    PIC 9(08).
000123  021104*                               USAGE IS  DBKEY.
000124  021105 01   L0240                    PIC X(08).
000125  021106**** DATA-WS3                          (0159)
000126  021107 01   L0320                    PIC 9(08).
000127  021108*                               USAGE IS  DBKEY.
000128  021109 01   L0330                    PIC X(08).
000129  021110 01   L0340                    PIC X(08).
000130  021111**** DATA-WS3                          (0160)
000131  021114 01   L0420                    PIC 9(08).
000132  021115 01   L0421                    PIC 9(08).
000133  021116*                               USAGE IS  DBKEY.
000134  021117 01   L0440                    PIC X(08).
000135  021118**** DATA-WS3                          (0161)
000136  021119 01   L0520                    PIC 9(08).
000137  021120*                               USAGE IS  DBKEY.
000138  021121 01   L0530                    PIC X(08).
000139  021122 01   L0540                    PIC X(08).
000140  021125**** DATA-WS3                          (0162)
000141  021126 01   L0620                    PIC 9(08).
000142  021127 01   L0621                    PIC 9(08).
000143  021128*                               USAGE IS  DBKEY.
000144  021129 01   L0640                    PIC X(08).
000145  021130**** DATA-WS3                          (0163)
000146  021131 01   L0720                    PIC 9(08).
000147  021132*                               USAGE IS  DBKEY.
000148  021133 01   L0740                    PIC X(08).
000149  050001*-----------------------------------------------------------------
000150  050002**** DML-INIT
000151  050003 PROCEDURE                                   DIVISION.
000152  050004 DML-M.
000153  050005           READY.
000154  050006           OPEN      OUTPUT          RSLT-F.
000155  050007*
000156  050008           DISPLAY   PSCHEME.
000157  050009           ACCEPT    T1      FROM  TIME.
000158  050010*
000159  050011*     INITIALIZATION
000160  050012*
000161  050013           MOVE  1     TO   TRUE.      MOVE  0     TO   FALSE.
000162  050014           MOVE  1     TO   ONN.       MOVE  0     TO   OFFF.
000163  050015           MOVE  1     TO   YES.       MOVE  0     TO   NOO.
000164  050016           MOVE  1     TO   ENDD.
000165  050017*
000166  050018           MOVE                FALSE       TO          LFOUND.
000167  050019*     SRSLT
000168  050020           MOVE      0     TO   CSRSLT.    MOVE  1     TO   CSR.
000169  050021           MOVE      30    TO   MAXSRSLT.
000170  050022*
000171  050023*
000172  050024*     RSLT
000173  050025           MOVE      00100  TO   MAXRSLT.
000174  050026*
000175  050027           MOVE  1   TO  I.
000176  050028 INIT0.
000177  050029           MOVE  0   TO  CVPT IN SRSLT (I).
000178  050030           IF    I   <   MAXSRSLT
000179  050031                 ADD 1 TO I        GO TO INIT0.
000180  050032*          CLEAR     MARK.
000181  050033           MOVE 0      TO    I.
000182  050034 INIT1.    IF    I =   MAXSRSLT        GO    TO    INIT3.
000183  050035           ADD   1     TO    I.
000184  050036                 MOVE 0      TO    J.
000185  050037 INIT2.          IF    J =   MAXRSLT   GO    TO    INIT1.
000186  050038                 ADD   1     TO    J.
000187  050039                       MOVE OFFF TO MARK IN STDBKEY (I, J).
000188  050040                                       GO    TO    INIT2.
000189  050041 INIT3.
000190  050042*     SET NODE-NO IN NOTBL.
000191  050043           MOVE 0      TO    I.
000192  050044 INIT4.    IF    I =   MAXNDTBL        GO    TO    INIT5.
000193  050045           ADD   1     TO    I.
000194  050046                 MOVE  I   TO  NODE-NO IN NOTBL (I).
000195  050047                                       GO    TO    INIT4.
000196  050048 INIT5.
```

COBOLプログラム

```
000197  050049****  INIT-SUB                                (0157)
000198  050050          MOVE FALSE      TO   L0120.
000199  050051          ADD  1          TO   CNDTBL.
000200  050052          MOVE  0157  TO  NPT IN NDTBL (CNDTBL).
000201  050053****  INIT-CVTL                               (0157)
000202  050054**    SET CALC VALUES     AT   CVTL.
000203  050055          ADD 1 TO CCVTL.     MOVE 'FOUR-SCHEMA STRUCTURE'  TO CVTL
000204  050056      (CCVTL).
000205  050057****  INIT-SUB                                (0158)
000206  050058          MOVE FALSE      TO   L0220.
000207  050059          ADD  1          TO   CNDTBL.
000208  050060          MOVE  0158  TO  NPT IN NDTBL (CNDTBL).
000209  050061****  INIT-SUB                                (0159)
000210  050062          MOVE FALSE      TO   L0320.
000211  050063          ADD  1          TO   CNDTBL.
000212  050064          MOVE  0159  TO  NPT IN NDTBL (CNDTBL).
000213  050065          MOVE  NODE-NO IN NDTBL (CNDTBL)  TO  I.
000214  050066          MOVE 'REPORTR'      TO   REN.
000215  050067          MOVE 0159           TO   J.
000216  050068*
000217  050069          MOVE 'TITLE' TO ITN.  PERFORM  S-SRSLT.
000218  050070****  INIT-SUB                                (0160)
000219  050071          MOVE FALSE      TO   L0420.
000220  050072          ADD  1          TO   CNDTBL.
000221  050073          MOVE  0160  TO  NPT IN NDTBL (CNDTBL).
000222  050074****  INIT-SUB                                (0161)
000223  050075          MOVE FALSE      TO   L0520.
000224  050076          ADD  1          TO   CNDTBL.
000225  050077          MOVE  0161  TO  NPT IN NDTBL (CNDTBL).
000226  050078          MOVE  NODE-NO IN NDTBL (CNDTBL)  TO  I.
000227  050079          MOVE 'EMPLOYEE'.    TO   REN.
000228  050080          MOVE 0161           TO   J.
000229  050081*
000230  050082          MOVE 'EFNAME' TO ITN.  PERFORM   S-SRSLT.
000231  050083****  INIT-SUB                                (0162)
000232  050084          MOVE FALSE      TO   L0620.
000233  050085          ADD  1          TO   CNDTBL.
000234  050086          MOVE  0162  TO  NPT IN NDTBL (CNDTBL).
000235  050087****  INIT-SUB                                (0163)
000236  050088          MOVE FALSE      TO   L0720.
000237  050089          ADD  1          TO   CNDTBL.
000238  050090          MOVE  0163  TO  NPT IN NDTBL (CNDTBL).
000239  060001*-------------------------------------------------------------*
000240  060002****  0-CALC                                  (0157)
000241  060003  L0101.
000242  060004          PERFORM   GNV.
000243  060005          IF   MODEE = ENOD       GO   TO   TERM.
000244  060006          MOVE       CVAL         TO   SUBJECTN IN SUBJECT.
000245  060007  L0102.
000246  060008          FIND  ANY  'SUBJECT'.
000247  060009          IF   DB-EXCEPTION IS 12  GO   TO   L0101.
000248  060010                                   PERFORM   COUNTR.
000249  060011***       GET  SUBJECT.
000250  060012***       ACCEPT    L0130  FROM                        CURRENCY.
000251  060013          MOVE       PGCS OF FCOM    TO   L0130.
000252  060014*-------------------------------------------------------------*
000253  060015****  NEXT                                    (0158)
000254  060016  L0201.
000255  060017          MOVE       0158  TO PT.
000256  060018          IF   L0220 = TRUE       GO TO L0202.
000257  060019          MOVE       FALSE        TO   LFOUND.
000258  060020          MOVE TRUE TO L0220.   MOVE FALSE TO L0221.
000259  060021                                  GO   TO   L0203.
000260  060022  L0202.
000261  060023          IF   LFOUND = TRUE  MOVE TRUE TO L0221.
000262  060024  L0203.
000263  060025          FIND NEXT 'REP-SUBJ' WITHIN 'S-R'.
000264  060026          IF   DB-EXCEPTION = 11   GO   TO   L0204.
000265  060027          IF   DB-EXCEPTION = 12   GO   TO   L0204.
000266  060028                                   GO   TO   L0206.
000267  060029*    END OF    SET  OCCURENCE
000268  060030  L0204.
000269  060031          MOVE FALSE TO L0220.  MOVE L0221 TO LFOUND.
000270  060032          IF   L0221 = TRUE  GO TO L0205.
000271  060033*    FAILURE  RETURN
000272  060034          MOVE 00157 TO PPT.  PERFORM FRTRN.
000273  060035                                  GO   TO   L0101.
000274  060036*    SUCCESS  RETURN
000275  060037  L0205.
000276  060038          MOVE 00157 TO PPT.  PERFORM SRTRN.
000277  060039                                  GO   TO   L0101.
000278  060040*    NOT  END  OF   SET
000279  060041  L0206.                                PERFORM   COUNTR.
000280  060042***       ACCEPT    L0240 FROM S-R CURRENCY.
000281  060043          MOVE       PGCS OF FCOM    TO L0240.
000282  060044***       GET  REP-SUBJ.
000283  060045*-------------------------------------------------------------*
000284  060046****  OWNER                                   (0159)
000285  060047  L0301.  MOVE 0159 TO PT.
000286  060048          FIND OWNER      WITHIN 'M-S'.
000287  060049          IF   DB-EXCEPTION IS 12  GO   TO   L0302.
000288  060050                                   GO   TO   L0303.
000289  060051*    NOT FOUND
000290  060052  L0302.
000291  060053          MOVE FALSE      TO   LFOUND.
000292  060054*    FAILURE  RETURN
000293  060055          MOVE 00158  TO PPT.  PERFORM FRTRN.
000294  060056                                  GO   TO   L0201.
```

COBOLプログラム

```
000293  060057*     FOUND
000296  060058 L0303.                              PERFORM COUNTR.
000297  060059***          ACCEPT    L0340  FROM  R-S CURRENCY.
000298  060060            MOVE      PGCS OF FCOM   TO L0340.
000299  060061 L0304.
000300  060062***         GET  REPORTR.
000301  060063            MOVE TRUE TO LFOUND.
000302  060064            MOVE L0340  TO DBK.
000303  060065            MOVE      L0340  TO   L0330.
000304  060066                                    PERFORM   RESULT.
000305  060067*-------------------------------------------------------------------
000306  060068***** NEXT                                (0160)
000307  060069 L0401.
000308  060070            MOVE      0160  TO PT.
000309  060071            IF   L0420 = TRUE     GO TO L0402.
000310  060072            MOVE      FALSE     TO   LFOUND.
000311  060073            MOVE TRUE TO L0420.   MOVE FALSE TO L0421.
000312  060074                                    GO   TO   L0403.
000313  060075 L0402.
000314  060076            IF   LFOUND = TRUE  MOVE TRUE TO L0421.
000315  060077 L0403.
000316  060078            FIND NEXT 'EMP-REP-LNK' WITHIN 'R-E'.
000317  060079            IF   DB-EXCEPTION = 11   GO   TO   L0404.
000318  060080            IF   DB-EXCEPTION = 12   GO   TO   L0404.
000319  060081                                    GO   TO   L0406.
000320  060082*     END OF    SET  OCCURENCE
000321  060083 L0404.
000322  060084*          MOVE FALSE TO L0420.  MOVE L0421 TO LFOUND.
000323  060085            IF   L0421 = TRUE   GO TO L0405.
000324  060086*     FAILURE   RETURN
000325  060087            MOVE 00158 TO PPT. PERFORM FRTRN.
000326  060088                                    GO   TO   L0201.
000327  060089*     SUCCESS   RETURN
000328  060090 L0405.
000329  060091            MOVE 00158 TO PPT. PERFORM SRTRN.
000330  060092                                    GO   TO   L0201.
000331  060093*     NOT END OF SET.
000332  060094 L0406.                                 PERFORM   COUNTR.
000333  060095***         ACCEPT    L0440 FROM R-E CURRENCY.
000334  060096            MOVE      PGCS OF FCOM   TO L0440.
000335  060097***         GET  EMP-REP-LNK.
000336  060098*-------------------------------------------------------------------
000337  060099***** OWNER                             (0161)
000338  060100 L0501. MOVE 0161 TO PT.
000339  060101            FIND OWNER    WITHIN 'E-R'.
000340  060102            IF   DB-EXCEPTION IS 12   GO   TO   L0502.
000341  060103                                    GO   TO   L0503.
000342  060104*     NOT FOUND
000343  060105 L0502.
000344  060106            MOVE FALSE      TO   LFOUND.
000345  060107*     FAILURE   RETURN
000346  060108            MOVE 00160 TO PPT. PERFORM FRTRN.
000347  060109                                    GO   TO   L0401.
000348  060110*     FOUND
000349  060111 L0503.                              PERFORM COUNTR.
000350  060112***         ACCEPT    L0540  FROM  E-R CURRENCY.
000351  060113            MOVE      PGCS OF FCOM   TO L0540.
000352  060114 L0504.
000353  060115***         GET  EMPLOYEE.
000354  060116            GET  CURRENT.
000355  060117              PERFORM  GCOUNT.
000356  060118            IF (  EFNAME = EFNAME )
000357  060119                NEXT SENTENCE  ELSE GO   TO   L0502.
000358  060120            MOVE TRUE TO LFOUND.
000359  060121            MOVE L0540  TO DBK.
000360  060122            MOVE      L0540  TO   L0530.
000361  060123                                    PERFORM   RESULT.
000362  060124*-------------------------------------------------------------------
000363  060125***** NEXT                                (0162)
000364  060126 L0601.
000365  060127            MOVE      0162  TO PT.
000366  060128            IF   L0520 = TRUE     GO TO L0602.
000367  060129            MOVE      FALSE     TO   LFOUND.
000368  060130            MOVE TRUE TO L0620.   MOVE FALSE TO L0621.
000369  060131                                    GO   TO   L0603.
000370  060132 L0602.
000371  060133            IF   LFOUND = TRUE  MOVE TRUE TO L0621.
000372  060134 L0603.
000373  060135            FIND NEXT 'PROJ-EMP-LNK' WITHIN 'E-P'.
000374  060136            IF   DB-EXCEPTION = 11   GO   TO   L0604.
000375  060137            IF   DB-EXCEPTION = 12   GO   TO   L0604.
000376  060138                                    GO   TO   L0606.
000377  060139*     END OF    SET  OCCURENCE
000378  060140 L0604.
000379  060141            MOVE FALSE TO L0620.  MOVE L0621 TO LFOUND.
000380  060142            IF   L0621 = TRUE  GO TO L0605.
000381  060143*     FAILURE   RETURN
000382  060144            MOVE 00160 TO PPT. PERFORM FRTRN.
000383  060145                                    GO   TO   L0401.
000384  060146*     SUCCESS   RETURN
000385  060147 L0605.
000386  060148            MOVE 00160 TO PPT. PERFORM SRTRN.
000387  060149                                    GO   TO   L0401.
000388  060150*     NOT END OF SET.
000389  060151 L0606.                                 PERFORM   COUNTR.
000390  060152***         ACCEPT    L0640 FROM E-P CURRENCY.
000391  060153            MOVE      PGCS OF FCOM   TO L0640.
000392  060154***         GET  PROJ-EMP-LNK.
```

COBOLプログラム

```
000393  060155*-------------------------------------------------------------*
000394  060156**** LAST-OWNER                    (0163)
000395  060157 L0701.  MOVE 0163 TO PT.
000396  060158          FIND OWNER     WITHIN  'O-E'.
000397  060159          IF   DB-EXCEPTION IS 12  GO   TO   L0702.
000398  060160                                   GO   TO   L0703.
000399  060161*    NOT FOUND.
000400  060162 L0702.
000401  060163          MOVE FALSE TO  LFOUND.
000402  060164*    FAILURE  RETURN.
000403  060165          MOVE 00162 TO PPT. PERFORM FRTRN.
000404  060166                                   GO   TO   L0601.
000405  060167*    FOUND
000406  060168 L0703.                          PERFORM COUNTR.
000407  060169***       ACCEPT    L0740  FROM P-E  CURRENCY.
000408  060170          MOVE      PGCS OF FCOM    TO   L0740.
000409  060171          GET  CURRENT.
000410  060172               PERFORM  GCOUNT.
000411  060173          IF ( PNAME = 'DISTRIBUTED DATABASE SYSTEMS'  )
000412  060174               NEXT SENTENCE  ELSE GO   TO   L0702.
000413  060175*    SATISFIED.
000414  060176          MOVE TRUE TO   LFOUND.
000415  060177*    SUCCESS  RETURN
000416  060178          MOVE  00162  TO  PPT.      PERFORM  SRTRN.
000417  060179                                   GO   TO   L0601.
000418  070001*-------------------------------------------------------------*
000419  070002* TERMINATION
000420  070003 TERM.
000421  070004*         ACCEPT    T2  FROM  TIME.
000422  070005*    GET THE RECORD OCCURRENCE USING THE DB-KEY
000423  070006*        IN THE RSLT0.
000424  070007*
000425  090001          ACCEPT    T3  FROM  TIME.
000426  090002          FINISH.
000427  090003          CLOSE  RSLT-F.
000428  090004**    PRINT ACCOUNTING.
000429  090005          DISPLAY 'THE NUMBER OF LOCATED  OCCURRENCES  = '  CNT.
000430  090006          DISPLAY 'THE NUMBER OF ACCESSED OCCURRENCES  = '  GCNT0.
000431  090007          DISPLAY 'THE NUMBER OF OUTPUT   OCCURRENCES  = '  GCNT.
000432  090008               COMPUTE  T4 = T2 - T1.
000433  090009               COMPUTE  T5 = T3 - T2.
000434  090010          DISPLAY 'THE ELAPSE TIME FOR ACCESSING = '  T4.
000435  090011          DISPLAY 'THE ELAPSE TIME FOR OUTPUT    = '  T5.
000436  090012               ADD  T4  TO  T5.
000437  090013          DISPLAY 'THE TOTAL TIME                = '  T5.
000438  090014          STOP RUN.
000439  100001*-------------------------------------------------------------*
000440  100002*
000441  100003*    GNV       CVAL =    THE CALC  VALUE
000442  100004*              MOVE =    END       IF CVAL IS ACQUIRED
000443  100005*                        NO        OTHERWISE
000444  100006*-------------------------------------------------------------*
000445  100007 GNV            SECTION.
000446  100008          IF   CCVTL NOT > 0        GO   TO   GNV1.
000447  100009          MOVE CVTL (CCVTL)   TO    CVAL.
000448  100010          MOVE NOD            TO    NODEE.
000449  100011          COMPUTE   CCVTL     =     CCVTL - 1.
000450  100012                                    GO   TO   GNV2.
000451  100013*    CCVTL =< 0
000452  100014 GNV1.    MOVE ENDD           TO    NODEE.
000453  100015 GNV2.    EXIT.
000454  100050*-------------------------------------------------------------*
000455  100051*
000456  100052*    RESULT  PT  =  THE POINTER TO THE NODE
000457  100053*            DBK  = DB-KEY
000458  100054*
000459  100055 RESULT         SECTION.
000460  100056          MOVE 0    TO   K.
000461  100057 RESULT1.
000462  100058          IF   K    =    CSRSLT  GO   TO   RESULT2.
000463  100059          ADD  1    TO   K.
000464  100060          IF  NPT IN SRSLT (K)  =  PT   NEXT SENTENCE
000465  100061                                ELSE GO   TO   RESULT1.
000466  100062          COMPUTE  L  =  CVPT IN SRSLT (K)  +  1.
000467  100063          MOVE  GNN   TO  MARK   IN STDBKEY (K, L).
000468  100064          MOVE  DBK   TO  DBKEY  IN STDBKEY (K, L).
000469  100065          MOVE  L     TO  CVPT   IN SRSLT   (K).
000470  100066                                  GO   TO   RESULT3.
000471  100067 RESULT2.      MOVE 602 TO ERR1.  MOVE  1 TO ERR2.  PERFORM ERRP.
000472  100068*     RETURN   TO   CALLER.
000473  100069 RESULT3.      EXIT.
000474  100070*-------------------------------------------------------------*
000475  100071*
000476  100072*     FRTRN    PT   THE POINTER TO THE CURRENT NODE
000477  100073*              PPT  THE POINTER TO THE FAILURE RETURNED NODE
000478  100074*
000479  100075 FRTRN          SECTION.
000480  100076*    SEARCH   NDTBL
000481  100077          MOVE      0     TO   I.
000482  100078 FRTRN1.
000483  100079          IF   I    = MAXNDTBL          GO   TO   FRTRN9.
000484  100080          ADD  1    TO   I.
000485  100081          IF  NPT IN NDTBL (I)  =  PPT  NEXT SENTENCE
000486  100082                                ELSE GO   TO   FRTRN1.
000487  100083               MOVE  NODE-NO IN NDTBL (I)  TO  K.
000488  100084**
000489  100085          COMPUTE  II = I + 1.    COMPUTE   J =  MAXNDTBL + 1.
000490  100086 FRTRN2.
```

COBOLプログラム

```
000491  100087      IF  J    = 11                    GO   TO   FRTRN10.
000492  100088      COMPUTE  J    =    J - 1.
000493  100089              IF  NPT IN NOTBL (J)  =  PT  NEXT SENTENCE
000494  100090                                  ELSE GO   TO   FRTRN2.
000495  100091              MOVE  NODE-NO IN NOTBL (J)  TO  L.
000496  100092          IF  K > NODE-NO IN SRSLT (CSRSLT)  GO TO FRTRN15.
000497  100093*     K   = THE FIRST COLOMN NUMB. OF THE FAILURE RETURNED NODE.
000498  100094*     L   = THE LAST  COLOMN NUMB. OF THE CURRENT NODE.
000499  100095*-----------------------------------------------------------------
000500  100096      MOVE      0    TO   I.
000501  100097 FRTRN3.
000502  100098      IF  I     =    CSRSLT          GO   TU   FRTRN11.
000503  100099      ADD 1     TO   I.
000504  100100          IF  NODE-NO IN SRSLT (I)  <  K      GO TO FRTRN3.
000505  100101          COMPUTE  M = CVPT IN SRSLT (I)  -  1.
000506  100102*****
000507  100103          COMPUTE  N  =  I - 1.
000508  100104 FRTRN4.  IF   N    = CSRSLT          GO   TU   FRTRN15.
000509  100105          ADD  1    TO   N.
000510  100106          IF  NODE-NO IN SRSLT (N)  >  L     GO TO FRTRN15.
000511  100107*-----------------------------------------------------------------
000512  100108              COMPUTE  II = CVPT IN SRSLT (N) + 1.
000513  100109              COMPUTE   JJ = M + 1.
000514  100110          IF  JJ = 0    GO TO FRTRN15.
000515  100111*         CLEAR     MARK.
000516  100112 FRTRN5.      IF  II  =  JJ          GO TO FRTRN6.
000517  100113              COMPUTE  II = II - 1.
000518  100114              MOVE  OFFF  TO  MARK IN STOBKEY (N, II).
000519  100115                                     GO   TU   FRTRN5.
000520  100116 FRTRN6.      MOVE  M  TO  CVPT IN SRSLT (N).
000521  100117                                     GO   TU   FRTRN4.
000522  100118*****
000523  100119*     ERROR     STOP
000524  100120 FRTRN9.      MOVE 603 TO ERR1.  MOVE  9 TO ERR2.  PERFORM ERRP.
000525  100121 FRTRN10.     MOVE 603 TO ERR1.  MOVE 10 TO ERR2.  PERFORM ERRP.
000526  100122 FRTRN11.     MOVE 603 TO ERR1.  MOVE 11 TO ERR2.  PERFORM ERRP.
000527  100123 FRTRN12.     MOVE 603 TO ERR1.  MOVE 12 TO ERR2.  PERFORM ERRP.
000528  100124*    RETURN    TO   CALLER
000529  100125 FRTRN15.                EXIT.
000530  100126*-----------------------------------------------------------------
000531  100127*
000532  100128*     SRTRN     PT : THE POINTER TO THE CURRENT NODE.
000533  100129*                PPT: THE POINTER TO THE SUCCESS RETURNED NODE
000534  100130*
000535  100131 SRTRN        SECTION.
000536  100132          MOVE 0     TO   I.
000537  100133 SRTRN1.  IF   I  =  MAXNOTBL        GO   TU   SRTRN9.
000538  100134          ADD  1     TO   I.
000539  100135          IF  NPT IN NOTBL (I)  =  PPT      NEXT SENTENCE
000540  100136                                  ELSE GO   TU   SRTRN1.
000541  100137              MOVE  NODE-NO IN NOTBL (I)  TO  K.
000542  100138*-----------------------------------------------------------------
000543  100139          COMPUTE   I    =    MAXNOTBL  +  1.
000544  100140 SRTRN2.  IF   I    =    1          GO   TO   SRTRN10.
000545  100141          COMPUTE   I    =    I    - 1.
000546  100142          IF  NPT IN NOTBL (I)  =  PT       NEXT SENTENCE
000547  100143                                  ELSE GO   TO   SRTRN2.
000548  100144              MOVE  NODE-NO IN NOTBL (I)  TU  L.
000549  100145*
000550  100146          IF   K    >    L          GO   TO   SRTRN15.
000551  100147*
000552  100148*     K   THE FIRST COLLUMN NUMB. OF THE SUCCESS RETURNED NODE.
000553  100149*     L   THE LAST  COLLUMN NUMB. OF THE CURRENT NODE.
000554  100150*-----------------------------------------------------------------
000555  100151          COMPUTE   I    =    CSRSLT    + 1.
000556  100152 SRTRN3.  IF   I    = 1             GO   TO   SRTRN15.
000557  100153          COMPUTE   I    =    I - 1.
000558  100154          IF  NODE-NO IN SRSLT (I)  >  L     GO TO SRTRN3.
000559  100155              MOVE  CVPT IN SRSLT (I)  TO  M.
000560  100156*-----------------------------------------------------------------SAVE I
000561  100157          MOVE I TO II.
000562  100158*
000563  100159          ADD 1 TO I.     MOVE 0 TO KK.
000564  100160 SRTRN4.
000565  100161          SUBTRACT 1 FROM I.
000566  100162          IF  NODE-NO IN SRSLT (I)  <  K     GO  TO  SRTRN6.
000567  100163          IF  CVPT IN SRSLT (I)  >  M
000568  100164                  MOVE CVPT IN SRSLT (I) TO M.
000569  100165          IF  I  >  1    GO  TO  SRTRN4.
000570  100166          MOVE  1  TO  KK.
000571  100167*
000572  100168 SRTRN6.  -----------------------------------------------------------
000573  100169          IF  I  >  II    GO TO SRTRNT.
000574  100170              MOVE  M  TO  CVPT IN SRSLT (I).
000575  100171             ADD 1 TO I.   GO TO SRTRN6.
000576  100172*              SUCCESS RETURN TO ROOT NODE
000577  100173*                 OUTPUT
000578  100174 SRTRNT.  -----------------------------------------------------------
000579  100175          IF  KK  NOT =  1   GO TO SRTRN15.
000580  100176          PERFORM  OUTPT.
000581  100177            GO TO SRTRN15.
000582  100178*     ERROR     STOP
000583  100179 SRTRN9.      MOVE 604 TO ERR1.  MOVE  9 TO ERR2.  PERFORM ERRP.
000584  100180 SRTRN10.     MOVE 604 TO ERR1.  MOVE 10 TO ERR2.  PERFORM ERRP.
000585  100181 SRTRN11.     MOVE 604 TO ERR1.  MOVE 11 TO ERR2.  PERFORM ERRP.
000586  100182 SRTRN12.     MOVE 604 TO ERR1.  MOVE 12 TO ERR2.  PERFORM ERRP.
000587  100183*     RETURN    TO   CALLER.
000588  100184 SRTRN15.                EXIT.
```

COBOLプログラム

```
000589  100185*-----------------------------------------------------------------
000590  100186 ACCOUNT  SECTION.
000591  100187*
000592  100188*     COUNT
000593  100189*
000594  100190 CUUNTR. ADD  1    TO   CNT.
000595  100191 CUUNTO. EXIT.
000596  100192*
000597  100193*     GCOUNT
000598  100194*
000599  100195 GCOUNT. ADD  1    TO   GCNTO.
000600  100196 GCOUNTO.     EXIT.
000601  100197*
000602  100198*     OUTPUT COUNT
000603  100199 GCOUNTP.  ADD 1 TO GCNT.
000604  100200 GCOUNTP-EXIT. EXIT.
000605  100201*-----------------------------------------------------------------
000606  100202*
000607  100203*  S-SRSLT          I    NODE NUMBER
000608  100204*                   REN  RECORD-TYPE
000609  100205*                   ITN  RESULT ATTRIBUTE NAME
000610  100206*                   J    POINTER TO THE VALUE
000611  100207*
000612  100208 S-SRSLT           SECTION.
000613  100209      ADD  1    TO   CSRSLT.
000614  100210      IF  CSRSLT  >  MAXSRSLT    GO TO SRSLT2.
000615  100211*
000616  100212          MOVE  I      TO  NODE-NO   IN SRSLT (CSRSLT).
000617  100213          MOVE  REN    TO  REC-TYPE  IN SRSLT (CSRSLT).
000618  100214          MOVE  ITN    TO  ATT-NAME  IN SRSLT (CSRSLT).
000619  100215          MOVE  J      TO  NPT       IN SRSLT (CSRSLT).
000620  100216          MOVE  0      TO  CVPT      IN SRSLT (CSRSLT).
000621  100217                       GO TO SRSLT3.
000622  100218 SRSLT2.   MOVE 608 TO ERR1.  MOVE  1 TO ERR2.  PERFORM ERRP.
000623  100219 SRSLT3. EXIT.
000624  100245 ERRP           SECTION.
000625  100246 ERRP1.
000626  100247      DISPLAY 'PATTERN NUMB. = '  ERR1  ', ERROR CODE = '  ERR2.
000627  100248      STOP RUN.
000628  100249 ERR-EXIT.    EXIT.
000629  100250*-----------------------------------------------------------------
000630  100251*
000631  100252* NEXTI      INPUT  II = THE NODE#
000632  100253*            OUTPUT  I = THE COLUMN OF SRSLT
000633  100254*                        IN WHICH II IS STORED
000634  100255 NEXTI          SECTION.
000635  100256 NEXTO.
000636  100257          ADD 1 TO I.
000637  100258          IF  NPT IN SRSLT (I)  NOT =  II
000638  100259                                GO TO NEXTO.
000639  100260 NEXT1.  EXIT.
000640  101220*-----------------------------------------------------------------
000641  101221*
000642  101222*              OUTPUT    PRINT
000643  101223*
000644  101224 OUTPT          SECTION.
000645  101225 OUTPTO.
000646  101226          MOVE 0  TO I J.
000647  101227 OUTPT1. ADD 1 TO J.
000648  102016**** DML-RESULT OUTPUT          (0159)
000649  102017          MOVE 0159 TO II.
000650  102018          PERFORM  NEXTI.
000651  102019*              I = THE COLMN. NUMB. OF SRSLT
000652  102020*                  WHICH 0159 IS STORED
000653  102021          IF  MARK IN STDBKEY (I, J)  =  OFFF    GO TO L0317.
000654  102022*             CLEAR MARK
000655  102023          MOVE OFFF TO MARK IN STDBKEY (I, J).
000656  102024***          FIND REPORTR  DB-KEY IS DBKEY IN STDBKEY (I, J).
000657  102025          MOVE DBKEY IN STDBKEY (I, J)  TO  PGCS OF FCOM.
000658  102026***          GET  REPORTR.
000659  102027          GET  CURRENT.
000660  102028              PERFORM  GCOUNTP.
000661  102029*     SET RESULT    TO   THE  RESULT    RELATION ( RSLT ).
000662  102030          MOVE TITLE IN REPORTR TO TITLE0159 IN RSLT.
000663  102031*
000664  102032 L0317.
000665  102033**** DML-RESULT OUTPUT            (0161)
000666  102034          MOVE 0161 TO II.
000667  102035          PERFORM  NEXTI.
000668  102036*              I = THE COLMN. NUMB. OF SRSLT
000669  102037*                  WHICH 0161 IS STORED
000670  102038          IF  MARK IN STDBKEY (I, J)  =  OFFF    GO TO L0317.
000671  102039*             CLEAR MARK
000672  102040          MOVE OFFF TO MARK IN STDBKEY (I, J).
000673  102041***          FIND EMPLOYEE  DB-KEY IS DBKEY IN STDBKEY (I, J).
000674  102042          MOVE DBKEY IN STDBKEY (I, J)  TO  PGCS OF FCOM.
000675  102043***          GET  EMPLOYEE.
000676  102044          GET  CURRENT.
000677  102045              PERFORM  GCOUNTP.
000678  102046*     SET RESULT    TO   THE  RESULT    RELATION ( RSLT ).
000679  102047          MOVE EFNAME IN EMPLOYEE TO EFNAME0161 IN RSLT.
000680  102048*
000681  102049 L0317.
000682  103228          DISPLAY  RSLT.
000683  103229          WRITE  RSLT  INVALID KEY
000684  103230                       DISPLAY  'RESULT FILE OVERFLOW'
000685  103231                       GO TO OUTPT10.
000686  103232          MOVE  SPACE  TO  RSLT.
```

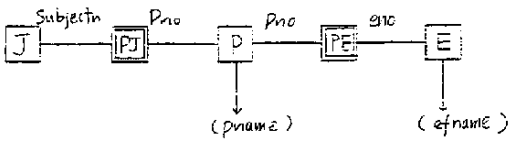
COBOLプログラム

```
000687  103233          IF  J  NOT <  CYPT IN SRSLT (I)
000688  103234                                  GO TO OUTPT10.
000689  103235          MOVE  0  TO  I.
000690  103236          GO  TO  OUTPT11.
000691  103237*           END OF OUTPUT
000692  103238 OUTPT10.   MOVE 0 TO I.
000693  103239 OUTPT11.   ADD  1 TO I.
000694  103240           IF  I  >  CSRSLT  GO TO OUTPT12.
000695  103241          MOVE  0  TO  CYPT_IN_SRSLT (I).
000696  103242           GO TO OUTPT11.
000697  103243 OUTPT12.
000698  103244          EXIT.
```

結果リスト

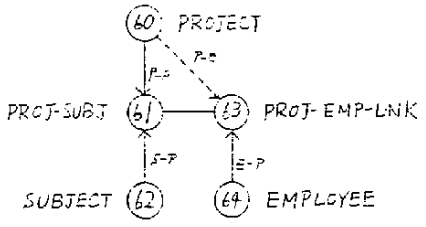
```
( ( TITLE ( EFNAME ( ) ) ) )
RESOURCE INTEGRATION SYSTEM ON JIPNET                                    EIJI

ANNUAL REPORT ON DISTRIBUTED PROCESSING OF RESUURCES                     MAKOTU
                                                                         EIJI
                                                                         SENICHI

                                                                         MAKOTO
ANNUAL REPORT ON DISTRIBUTED PROCESSING OF RESOURCES                     HIROYUKI
                                                                         EIJI
                                                                         SENICHI
                                                                         MAKOTO
                                                                         HIROYUKI
ANNUAL REPORT ON DISTRIBUTED PROCESSING OF RESOURCES (FINAL REPORT)-DISTRIBUTED DATABASES
                                                                         MAKOTO
                                                                         MAKOTO
                                                                         EIJI
RESOURCE INTEGRATION AND DATA SHARING ON HETEROGENEOUS RESOURCE SHARING  MAKOTO
                                                                         EIJI
THE FOUR-SCHEMA CONCEPT AS THE GROSS ARCHITECTURE OF DISTRIBUTED DATABASES AND HETERUGENEITY PROBLEMS
                                                                         MAKOTO
                                                                         EIJI
SCHEMA-LAYER DESIGN AND QUERY PROCESSING IN DISTRIBUTED DATABASE SYSTEM:JOOBS
                                                                         MAKOTO

THE NUMBER OF LOCATED  OCCURRENCES  = 00000141
THE NUMBER OF ACCESSED OCCURRENCES  = 00000063
THE NUMBER OF OUTPUT   OCCURRENCES  = 00000025
THE ELAPSE TIME FOR ACCESSING = 00000296
THE ELAPSE TIME FOR OUTPUT    = 00000000
THE TOTAL TIME                = 00000296
```

TEST 5&3




```
RANGE (P,PROJECT) (E, EMPLOYEE) (J, SUBJECT ) ;
RANGE (PJ, PROJ-SUBJ) (PE, PROJ-EMP-LNK) ;
RETRIEVE INTO TEST503 (P.PNAME, E.EFNAME)
WHERE P.PNO      = PE.PNO    AND PE.ENO      = E.ENO      AND
      P.PNO      = PJ.PNO    AND PJ.SUBJECTN = J.SUBJECTN ;
```

TEST 5&3



```
000002  010002 IDENTIFICATION                   DIVISION.
000003  010003 PROGRAM-ID.                      DBTGMDNL.
000004  010004 AUTHOR.                          JDDBS.
000005  010005*
000006  010006*
000007  010007 ENVIRONMENT                      DIVISION.
000008  010008 CONFIGURATION                    SECTION.
000009  010009 SUBSCHEMA-NAME.                  'PROJSUB1'.
000010  010010 SOURCE-COMPUTER.                 FACOM-M160.
000011  010011 OBJECT-COMPUTER.                 FACOM-M160.
000012  010012*
000013  010013 INPUT-OUTPUT                     SECTION.
000014  010014 FILE-CONTROL.
000015  010015     SELECT   RSLT-F              ASSIGN   TO   DA-S-UD1.
000016  020001**** DATA-FD1
000017  020002 DATA                    DIVISION.
000018  020003 FILE                    SECTION.
000019  020004*
000020  020005 FD  RSLT-F  LABEL RECORD  IS  STANDARD.
000021  020006 01  RSLT.
000022  020092**** DATA-FD2                     (0060)
000023  020093     03  PNAME0060   PIC  X(100).
000024  020112**** DATA-FD2                     (0064)
000025  020113     03  EFNAME0064   PIC  X(020).
000026  021007*-----------------------------------------------------*
000027  021008**** DATA-WS1
000028  021009 WORKING-STORAGE          SECTION.
000029  021010*
000030  021011 01  NDTBL.
000031  021012     02  NDTBL0          OCCURS   005 TIMES.
000032  021013         03  NPT         PIC      9(05).
000033  021014         03  NODE-NO     PIC      9(05).
000034  021015 01  CNDTBL              PIC      9(05)     VALUE    0.
000035  021016 01  MAXNDTBL            PIC      9(05)     VALUE    005.
000036  021017*
000037  021018*    SCHEME OF THE RESULT RELATION
000038  021019 01  SRSLT.
000039  021020     02  SRSLT0          OCCURS   30 TIMES.
000040  021021         03  NPT         PIC      9(05).
000041  021022         03  NODE-NO     PIC      9(05).
000042  021023             03  CVPT    PIC      9(05).
000043  021024         03  REC-TYPE    PIC      X(16).
000044  021025         03  ATT-NAME    PIC      X(16).
000045  021026 01  CSRSLT              PIC      9(05)     VALUE    0.
000046  021027 01  MAXSRSLT            PIC      9(05)     VALUE    30.
000047  021028 01  CSP                 PIC      9(05)     VALUE    1.
000048  021029*    TABLE FOR STORING DBKEYS OF SATISFIABLE OCCURRENCES.
000049  021030 01  STDBKEY.
000050  021031     02  STDBKEY1        OCCURS   30  TIMES.
000051  021032         03  STDBKEY2    OCCURS   00100  TIMES.
000052  021033             04  MARK    PIC      9(02).
000053  021034****         04  DBKEY   USAGE IS  DB-KEY.
000054  021035             04  DBKEY   PIC      X(08).
000055  021036*
000056  021037*    STATUS   VEHICLE FROM   PRECEDING NODE
000057  021038 01  LFOUND              PIC      9(02).
000058  021039*    THE NUMB. OF ACCESSED OCCURRENCES
000059  021040 01  CNT                 PIC      9(08)     VALUE    0.
000060  021041 01  GCNT                PIC      9(08)     VALUE    0.
000061  021042 01  GCNTO               PIC      9(08)     VALUE    0.
000062  021043*
000063  021044*    VARIABLES FOR STORING ELAPSE TIME.
000064  021045 01  T1                  PIC      9(08).
000065  021046 01  T2                  PIC      9(08).
000066  021047 01  T3                  PIC      9(08).
000067  021048 01  T4                  PIC      9(08).
000068  021049 01  T5                  PIC      9(08).
000069  021050*    CONSTANT
000070  021051 01  FALSE               PIC      9(02)     VALUE    0.
000071  021052 01  TRUE                PIC      9(02)     VALUE    1.
000072  021053 01  ENDD                PIC      9(02)     VALUE    0.
000073  021054 01  YES                 PIC      9(02)     VALUE    1.
000074  021055 01  NOO                 PIC      9(02)     VALUE    0.
000075  021056 01  ONN                 PIC      9(02)     VALUE    1.
000076  021057 01  OFFF                PIC      9(02)     VALUE    0.
000077  021058*
000078  021059*
000079  021060*    WORKING  VARIABLES
000080  021061 01  I                   PIC      S9(08).
000081  021062 01  J                   PIC      S9(08).
000082  021063 01  K                   PIC      S9(08).
000083  021064 01  L                   PIC      S9(08).
000084  021065 01  M                   PIC      S9(08).
000085  021066 01  N                   PIC      S9(08).
000086  021067 01  II                  PIC      S9(08).
000087  021068 01  JJ                  PIC      S9(08).
000088  021069 01  KK                  PIC      S9(08).
000089  021070 01  PT                  PIC      9(08).
000090  021071 01  PPT                 PIC      9(08).
000091  021072*77  DBK                 USAGE IS  DB-KEY.
000092  021073 01  DBK                 PIC      X(08).
000093  021074*    NAME OF RECORD-TYPE (REN) AND ITEM (ITN)
000094  021075 01  REN                 PIC      X(16).
000095  021076 01  ITN                 PIC      X(16).
000096  021077 01  EHR1                PIC      9(03).
000097  021078 01  EHR2                PIC      9(02).
000098  021079*
```

```
000099  021080*     SCHEME OF THE RESULT
000100  021081 01   PSCHEME                          PIC       X(025)    VALUE
000101  021082           '( PNAME ( ) ( ERNAME ) ) '.
000102  021083*
000103  021084*     FOR RSLT RELATION
000104  021085 01   CRSLT                            PIC       9(05)     VALUE 0.
000105  021086 01   MAXRSLT                          PIC       9(05)     VALUE 00100.
000106  021087**** DATA-WS3                          (0060)
000107  021088 01   L0120                 PIC  9(08).
000108  021089*                           USAGE IS  DBKEY.
000109  021090 01   L0130                 PIC  X(08).
000110  021091 01   L0140                 PIC  X(08).
000111  021094**** DATA-WS3                          (0061)
000112  021095 01   L0220                 PIC  9(08).
000113  021096 01   L0221                 PIC  9(08).
000114  021097*                           USAGE IS  DBKEY.
000115  021098 01   L0240                 PIC  X(08).
000116  021099**** DATA-WS3                          (0062)
000117  021100 01   L0320                 PIC  9(08).
000118  021101*                           USAGE IS  DBKEY.
000119  021102 01   L0340                 PIC  X(08).
000120  021103**** DATA-WS3                          (0063)
000121  021104 01   L0420                 PIC  9(08).
000122  021105 01   L0421                 PIC  9(08).
000123  021106*                           USAGE IS  DBKEY;
000124  021107 01   L0440                 PIC  X(08).
000125  021108**** DATA-WS3                          (0064)
000126  021109 01   L0520                 PIC  9(08).
000127  021110*                           USAGE IS  DBKEY.
000128  021111 01   L0540                 PIC  X(08).
000129  050001*-------------------------------------------------------------*
000130  050002**** DML-INIT
000131  050003 PROCEDURE                                DIVISION.
000132  050004 DML-M.
000133  050005           READY.
000134  050006           OPEN       OUTPUT          RSLT-F.
000135  050007*
000136  050008           DISPLAY    PSCHEME.
000137  050009           ACCEPT     T1     FROM  TIME.
000138  050010*
000139  050011*     INITIALIZATION
000140  050012*
000141  050013           MOVE 1     TO   TRUE.      MOVE 0    TO   FALSE.
000142  050014           MOVE 1     TO   UNK.       MOVE 0    TO   OFFF.
000143  050015           MOVE 1     TO   YES.       MOVE 0    TO   NRU.
000144  050016           MOVE 1     TO   ENDD.
000145  050017*
000146  050018           MOVE            FALSE    TO         LFOUND.
000147  050019*     SRSLT
000148  050020           MOVE       0    TO   CSRSLT.   MOVE 1    TO   CSR.
000149  050021           MOVE       30   TO   MAXSRSLT.
000150  050022*
000151  050023*
000152  050024*     RSLT
000153  050025           MOVE       00100  TO   MAXRSLT.
000154  050026*
000155  050027           MOVE  1  TO  I.
000156  050028 INITO.
000157  050029           MOVE  0  TO  CVPT IN SRSLT (I).
000158  050030           IF   I  <  MAXSRSLT
000159  050031                ADD 1 TO I      GO TO INITO.
000160  050032*     CLEAR    MARK
000161  050033           MOVE 0     TO   I.
000162  050034 INIT1.    IF   I =  MAXSRSLT        GO   TO   INIT3.
000163  050035           ADD  1     TO   I.
000164  050036                MOVE 0     TO   J.
000165  050037 INIT2.        IF   J =  MAXRSLT   GO   TO   INIT1.
000166  050038           ADD  1     TO   J.
000167  050039                   MOVE OFFF TO MARK IN STUBKEY (I, J).
000168  050040                                    GO   TO   INIT2.
000169  050041 INIT3.
000170  050042*     SET NODE-NO IN NDTBL.
000171  050043           MOVE 0     TO   I.
000172  050044 INIT4.    IF   I =  MAXNDTBL        GO   TO   INIT5.
000173  050045           ADD  1     TO   I.
000174  050046                   MOVE  I  TO  NODE-NO IN NDTBL (I).
000175  050047                                    GO   TO   INIT4.
000176  050048 INIT5.
000177  050049**** INIT-SUB                          (0060)
000178  050050           MOVE FALSE      TO   L0120.
000179  050051           ADD  1          TO   CNDTBL.
000180  050052           MOVE  0060  TO  RPT IN NDTBL (CNDTBL).
000181  050053           MOVE  NODE-NO IN NDTBL (CNDTBL)  TO  I.
000182  050054           MOVE 'PROJECT'      TO   REN.
000183  050055           MOVE 0060           TO   J.
000184  050056*
000185  050057           MOVE 'PNAME' TO ITN.  PERFORM   S-SRSLT.
000186  050058**** INIT-SUB                          (0061)
000187  050059           MOVE FALSE      TO   L0220.
000188  050060           ADD  1          TO   CNDTBL.
000189  050061           MOVE  0061  TO  RPT IN NDTBL (CNDTBL).
000190  050062**** INIT-SUB                          (0062)
000191  050063           MOVE FALSE      TO   L0320.
000192  050064           ADD  1          TO   CNDTBL.
000193  050065           MOVE  0062  TO  RPT IN NDTBL (CNDTBL).
000194  050066**** INIT-SUB                          (0063)
000195  050067           MOVE FALSE      TO   L0420.
000196  050068           ADD  1          TO   CNDTBL.
```

```
000197  050069          MOVE  0063  TO  PPT IN NDTBL (C-UTBL).
000198  050070**** INIT-SUB                          (0064)
000199  050071          MOVE FALSE      TO   L0520.
000200  050072          ADD  1          TO   CNDTBL.
000201  050073          MOVE  0064  TO  PPT IN NDTBL (CNDTBL).
000202  050074          MOVE  NODE-NO IN NDTBL (CNDTBL)  TO  I.
000203  050075          MOVE 'EMPLOYEE'      TO   REN.
000204  050076          MOVE 0064        TO   J.
000205  050077*
000206  050078          MOVE 'EFNAME' TO ITN,  PERFORM   S-SRSLT.
000207  060001*-------------------------------------------------------------------*
000208  060002**** FIRST                            (0060)
000209  060003****      FIND FIRST      PROJECT        WITHIN  RANGEA .
000210  060004          FIND FIRST      'PROJECT'      WITHIN 'RANGEA' RANGE.
000211  060005          IF   DB-EXCEPTION IS 11  GO TO TERM.
000212  060006          IF   DB-EXCEPTION IS 12  GO TO TERM.
000213  060007                                   GO TO L0102.
000214  060008 L0101.
000215  060009****      FIND NEXT       PROJECT        WITHIN  RANGEA.
000216  060010          FIND NEXT       'PROJECT'      WITHIN 'RANGEA' RANGE.
000217  060011          IF   DB-EXCEPTION IS 12  GO TO TERM.
000218  060012          IF   DB-EXCEPTION IS 11  GO TO TERM.
000219  060013 L0102.
000220  060014          MOVE 0060  TO   PT.
000221  060015                                   PERFORM   COUNTR.
000222  060016***       ACCEPT    L0140  FROM                        CURRENCY.
000223  060017          MOVE      PGCS OF FCOM   TO   L0140.
000224  060018          MOVE      L0140  TO   L0130.
000225  060019          MOVE      L0140  TO   DBK.
000226  060020                                   PERFORM   RESULT.
000227  060021*-------------------------------------------------------------------*
000228  060022**** NEXT                           (0061)
000229  060023 L0201.
000230  060024          MOVE      0061  TO PT.
000231  060025          IF   L0220 = TRUE      GO TO L0202.
000232  060026          MOVE      FALSE      TO   LFOUND.
000233  060027          MOVE TRUE TO L0220.   MOVE FALSE TO L0221.
000234  060028                                   GO   TO   L0203.
000235  060029 L0202.
000236  060030          IF   LFOUND = TRUE  MOVE TRUE TO L0221.
000237  060031 L0203.
000238  060032          FIND NEXT 'PROJ-SUBJ' WITHIN 'P-S'.
000239  060033          IF   DB-EXCEPTION = 11   GO   TO   L0204.
000240  060034          IF   DB-EXCEPTION = 12   GO   TO   L0204.
000241  060035                                   GO   TO   L0206.
000242  060036*    END OF   SET  OCCURENCE
000243  060037 L0204.
000244  060038          MOVE FALSE TO L0220.  MOVE L0221 TO LFOUND.
000245  060039          IF   L0221 = TRUE  GO TO L0205.
000246  060040*    FAILURE  RETURN
000247  060041          MOVE 00060 TO PPT. PERFORM FRTRN.
000248  060042                                   GO   TO   L0101.
000249  060043*    SUCCESS  RETURN
000250  060044 L0205.
000251  060045          MOVE 00063 TO PPT. PERFORM SRTRN.
000252  060046                                   GO   TO   L0401.
000253  060047*    OUT END  OF   SET
000254  060048 L0206.                              PERFORM   COUNTR.
000255  060049***       ACCEPT    L0240 FROM P-S CURRENCY.
000256  060050          MOVE      PGCS OF FCOM   TO L0240.
000257  060051***       SET  PROJ-SUBJ.
000258  060052*-------------------------------------------------------------------*
000259  060053**** LAST-OWNER                       (0062)
000260  060054 L0301. MOVE 0062 TO PT.
000261  060055          FIND OWNER      WITHIN    'S-P'.
000262  060056          IF   DB-EXCEPTION IS 12  GO   TO   L0302.
000263  060057                                   GO   TO   L0303.
000264  060058*    NOT FOUND.
000265  060059 L0302.
000266  060060          MOVE FALSE TO  LFOUND.
000267  060061*    FAILURE  RETURN.
000268  060062          MOVE 00061 TO PPT. PERFORM FRTRN.
000269  060063                                   GO   TO   L0201.
000270  060064*    FOUND
000271  060065 L0303.                          PERFORM COUNTR.
000272  060066***       ACCEPT    L0340  FROM S-P  CURRENCY.
000273  060067          MOVE      PGCS OF FCOM   TO   L0340.
000274  060068*    SATISFIED.
000275  060069          MOVE TRUE TO   LFOUND.
000276  060070*    SUCCESS  RETURN.
000277  060071          MOVE  00061  TO  PPT.      PERFORM  SRTRN.
000278  060072                                   GO   TO   L0201.
000279  060073*-------------------------------------------------------------------*
000280  060074**** NEXT                           (0063)
000281  060075 L0401.
000282  060076          MOVE      0063  TO PT.
000283  060077          IF   L0420 = TRUE      GO TO L0402.
000284  060078          MOVE      FALSE      TO   LFOUND.
000285  060079          MOVE TRUE TO L0420.   MOVE FALSE TO L0421.
000286  060080                                   GO   TO   L0403.
000287  060081 L0402.
000288  060082          IF   LFOUND = TRUE  MOVE TRUE TO L0421.
000289  060083 L0403.
000290  060084          FIND NEXT 'PROJ-EMP-LNK' WITHIN 'P-E'.
000291  060085          IF   DB-EXCEPTION = 11   GO   TO   L0404.
000292  060086          IF   DB-EXCEPTION = 12   GO   TO   L0404.
000293  060087                                   GO   TO   L0406.
000294  060088*    END OF   SET  OCCURENCE
```

```
000295  060080 L0404.
000296  060090          MOVE FALSE TO L0420.  MOVE L0421 TO LFOUND.
000297  060091          IF   L0421 = TRUE  GO TO L0405.
000298  060092*     FAILURE  RETURN
000299  060093          MOVE 00060 TO PPT. PERFORM FRTRN.
000300  060094                                    GO   TO   L0101.
000301  060095*     SUCCESS  RETURN
000302  060096 L0405.
000303  060097          MOVE 00060 TO PPT. PERFORM SRTRN.
000304  060098                                    GO   TO   L0101.
000305  060099*     NOT END  OF   SET
000306  060100 L0406.                              PERFORM   COUNTR.
000307  060101***       ACCEPT    L0440 FROM P-E CURRENCY.
000308  060102          MOVE      PGCS OF FCOM   TO L0440.
000309  060103***       GET  PROJ-EMP-LNK.
000310  060104*-------------------------------------------------------------------*
000311  060105***** LAST-OWNER                  (0064)
000312  060105 L0501. MOVE 0064 TO PT.
000313  060107          FIND OWNER     WITHIN    'E-P'.
000314  060108          IF   DB-EXCEPTION IS 12  GO   TO   L0502.
000315  060109                                   GO   TO   L0503.
000316  060110*     NOT FOUND.
000317  060111 L0502.
000318  060112          MOVE FALSE TO  LFOUND.
000319  060113*     FAILURE  RETURN.
000320  060114          MOVE 00063 TO PPT. PERFORM FRTRN.
000321  060115                                   GO   TO   L0401.
000322  060116*     FOUND
000323  060117 L0503.                        PERFORM COUNTR.
000324  060118***       ACCEPT    L0540  FROM E-P  CURRENCY.
000325  060119          MOVE      PGCS OF FCOM   TO   L0540.
000326  060120*     SATISFIED.
000327  060121          MOVE TRUE TO   LFOUND.
000328  060122          MOVE L0540 TO DBK. PERFORM RESULT.
000329  060123*     SUCCESS  RETURN.
000330  060124          MOVE  00063  TO  PPT.      PERFORM  SRTRN.
000331  060125                                   GO   TO   L0401.
000332  070001*-------------------------------------------------------------------*
000333  070002* TERMINATION
000334  070003 TERM.
000335  070004          ACCEPT    T2   FROM  TIME.
000336  070005*     GET THE RECORD OCCURRENCE USING THE DB-KEY
000337  070006*         IN THE RSLTO.
000338  070007*
000339  090001          ACCEPT    T3   FROM  TIME.
000340  090002          FINISH.
000341  090003          CLOSE  RSLT-F.
000342  090004*     PRINT ACCOUNTING.
000343  090005          DISPLAY 'THE NUMBER OF LOCATED  OCCURRENCES  = '  CNT.
000344  090006          DISPLAY 'THE NUMBER OF ACCESSED OCCURRENCES  = '  GCNTO.
000345  090007          DISPLAY 'THE NUMBER OF OUTPUT   OCCURRENCES  = '  GCNT.
000346  090008               COMPUTE  T4 = T2 - T1.
000347  090009               COMPUTE  T5 = T3 - T2.
000348  090010          DISPLAY 'THE ELAPSE TIME FOR ACCESSING = '  T4.
000349  090011          DISPLAY 'THE ELAPSE TIME FOR OUTPUT    = '  T5.
000350  090012               ADD  T4  TO  T5.
000351  090013          DISPLAY 'THE TOTAL TIME                = '  T5.
000352  090014          STOP RUN.
000353  100035*-------------------------------------------------------------------*
000354  100036*
000355  100037*     RESULT   PT   = THE POINTER  TO THE NODE
000356  100038*              DBK  = DB-KEY
000357  100039*
000358  100040 RESULT       SECTION.
000359  100041          MOVE 0    TO   K.
000360  100042 RESULT1.
000361  100043          IF   K    =      CSRSLT  GO   TO   RESULT2.
000362  100044          ADD  1    TO   K.
000363  100045          IF  RPT IN SHSLT (K)  =  PT   NEXT SENTENCE
000364  100046                                ELSE GO   TO   RESULT1.
000365  100047          COMPUTE  L  =  CVPT IN SHSLT (K)  +  1.
000366  100048          MOVE  ONE  TO  MARK   IN STDBKEY (K, L).
000367  100049          MOVE  DBK  TO  DBKEY  IN STDBKEY (K, L).
000368  100050          MOVE  L    TO  CVPT   IN SHSLT   (K).
000369  100051                                   GO   TO   RESULT3.
000370  100052 RESULT2.     MOVE 602 TO ERR1.  MOVE  1 TO ERR2.  PERFORM ERRP.
000371  100053*     RETURN   TO   CALLER.
000372  10005A RESULT3.     EXIT.
000373  100055*-------------------------------------------------------------------*
000374  100056*
000375  100057*     FRTRN    PT   THE POINTER TO THE CURRENT NODE
000376  100058*              PPT  THE POINTER TO THE FAILURE RETURNED NODE
000377  100059*
000378  100060 FRTRN        SECTION.
000379  100061*     SEARCH   NDTBL
000380  100062          MOVE     0    TO   I.
000381  100063 FRTRN1.
000382  100064          IF  I    = MAXNDTBL          GO   TO   FRTRN9.
000383  100065          ADD 1    TO   I.
000384  100066              IF  RPT IN NDTBL (I)  =  PPT  NEXT SENTENCE
000385  100067                                    ELSE GO   TO   FRTRN1.
000386  100068                  MOVE  NODE-NO IN NDTBL (I)  TO  K.
000387  100069*
000388  100070          COMPUTE  II = I + 1.    COMPUTE   J =  MAXNDTBL + 1.
000389  100071 FRTRN2.
000390  100072          IF  J    = II                GO   TO   FRTRN10.
000391  100073          COMPUTE  J    =    J - 1.
000392  100074              IF  RPT IN NDTBL (J)  =  PT   NEXT SENTENCE
```

```
000393  100075                              ELSE GO   TO   FRTRN12.
000394  100076                  MOVE  NODE-NO IN NDTBL (J)  TO  L.
000395  100077              IF  K > NODE-NO IN SRSLT (CSRSLT)  GO TO FRTRN15.
000396  100078*     K    = THE FIRST COLUMN NUMB. OF THE FAILURE RETURNED NODE.
000397  100079*     L    = THE LAST  COLUMN NUMB. OF THE CURRENT NODE.
000398  100080*
000399  100081      MOVE      0     TO   I.
000400  100082 FRTRN3.
000401  100083      IF  I     =     CSRSLT          GO   TO   FRTRN11.
000402  100084      ADD 1     TO    I.
000403  100085              IF  NODE-NO IN SRSLT (I)  <  K      GO TO FRTRN3.
000404  100086              COMPUTE  M = CVPT IN SRSLT (I)  -  1.
000405  100087*****
000406  100088              COMPUTE  N  =  I - 1.
000407  100089 FRTRN4.  IF    N     = CSRSLT          GO   TO   FRTRN15.
000408  100090              ADD  1    TO   N.
000409  100091              IF  NODE-NO IN SRSLT (N)  >  L      GO TO FRTRN15.
000410  100092*
000411  100093                  COMPUTE  II = CVPT IN SRSLT (N) + 1.
000412  100094                   COMPUTE   JJ = M + 1.
000413  100095              IF  JJ = 0     GO TO FRTRN15.
000414  100096*             CLEAR      MARK.
000415  100097 FRTRN5.           IF  II  =  JJ          GO TO FRTRN6.
000416  100098                   COMPUTE   II = II - 1.
000417  100099                  MOVE  OFFF  TO  MARK IN STDBKEY (N, II).
000418  100100                                         GO   TO   FRTRN5.
000419  100101 FRTRN6.          MOVE  N  TO  CVPT IN SRSLT (I).
000420  100102                                         GO   TO   FRTRN4.
000421  100103*****
000422  100104*     ERROR     STOP
000423  100105 FRTRN9.          MOVE 603 TO ERR1.  MOVE  9 TO ERR2.  PERFORM ERRP.
000424  100106 FRTRN10.         MOVE 603 TO ERR1.  MOVE 10 TO ERR2.  PERFORM ERRP.
000425  100107 FRTRN11.         MOVE 603 TO ERR1.  MOVE 11 TO ERR2.  PERFORM ERRP.
000426  100108 FRTRN12.         MOVE 603 TO ERR1.  MOVE 12 TO ERR2.  PERFORM ERRP.
000427  100109*     RETURN    TO   CALLER
000428  100110 FRTRN15.                 EXIT.
000429  100111*-------------------------------------------------------------------
000430  100112*
000431  100113*     SRTRN     PT : THE POINTER TO THE CURRENT NODE.
000432  100114*                PPT: THE POINTER TO THE SUCCESS RETURNED NODE
000433  100115*
000434  100116 SRTRN            SECTION.
000435  100117          MOVE 0      TO   I.
000436  100118 SRTRN1.  IF   I  =  MAXNDTBL        GO   TO   SRTRN9.
000437  100119          ADD  1      TO   I.
000438  100120          IF  PPT IN NDTBL (I)  =  PPT     NEXT SENTENCE
000439  100121                                   ELSE GO   TO   SRTRN1.
000440  100122              MOVE  NODE-NO IN NDTBL (I)  TO  K.
000441  100123*
000442  100124          COMPUTE   J     =     MAXNDTBL  +     1.
000443  100125 SRTRN2.  IF    J     =     1           GO   TO   SRTRN10.
000444  100126          COMPUTE   J     =     J     - 1.
000445  100127          IF  PPT IN NDTBL (J)  =  PT      NEXT SENTENCE
000446  100128                                   ELSE GO   TO   SRTRN2.
000447  100129              MOVE  NODE-NO IN NDTBL (J)  TO  L.
000448  100130*
000449  100131          IF   K     >     L           GO   TO   SRTRN15.
000450  100132*
000451  100133*     K    THE FIRST COLUMN NUMB. OF THE SUCCESS RETURNED NODE.
000452  100134*     L    THE LAST  COLUMN NUMB. OF THE CURRENT NODE.
000453  100135*
000454  100136          COMPUTE   I     =     CSRSLT    + 1.
000455  100137 SRTRN3.  IF    I     = 1               GO   TO   SRTRN15.
000456  100138          COMPUTE   I     =     I     - 1.
000457  100139          IF  NODE-NO IN SRSLT (I)  >  L      GO TO SRTRN3.
000458  100140              MOVE  CVPT IN SRSLT (I)  TO  M.
000459  100141*             SAVE  I
000460  100142          MOVE  I TO II.
000461  100143*
000462  100144          ADD 1 TO I.     MOVE 0 TO KK.
000463  100145 SRTRN4.
000464  100146          SUBTRACT 1 FROM I.
000465  100147          IF  NODE-NO IN SRSLT (I)  <  K    GO  TO  SRTRN6.
000466  100148          IF  CVPT IN SRSLT (I)  >  M
000467  100149                  MOVE CVPT IN SRSLT (I) TO M.
000468  100150          IF   I  >  1   GO  TO  SRTRN4.
000469  100151          MOVE  1  TO  KK.
000470  100152*
000471  100153 SRTRN6.
000472  100154*         IF   I  >  II        GO TO SRTRN7.
000473  100155              MOVE  0  TO  CVPT IN SRSLT (I).
000474  100156             ADD 1 TO I.   GO TO SRTRN6.
000475  100157*              SUCCESS RETURN TO ROOT NODE
000476  100158*                 OUTPUT
000477  100159 SRTRN7.
000478  100160          IF  KK  NOT =  1   GO TO SRTRN15.
000479  100161*     SET THE LOWEST TUPLE NUMB. TO EVERY CVPT.
000480  100162          MOVE CSRSLT     TO   I.
000481  100163 SRTRN80.       MOVE CVPT IN SRSLT (I)   TO   M.
000482  100164          COMPUTE   I     =     CSRSLT  - 1.
000483  100165 SRTRN82.       IF    I = 1               GO   TO   SRTRN86.
000484  100166          SUBTRACT  1     FROM I.
000485  100167          IF  CVPT IN SRSLT (I)  >     M     GO TO SRTRN80.
000486  100168              MOVE M      TO   CVPT IN SRSLT (I).
000487  100169                                         GO   TO   SRTRN82.
000488  100170*     OUTPUT
000489  100171 SRTRN85.
000490  100172          PERFORM  OUTPT.
```

```
000491  100173              GO TO SRTR015.
000492  100174*     ERROR     STOP
000493  100175 SRTR09.        MOVE 604 TO ERR1.  MOVE  9 TO ERR2.  PERFORM ERRP.
000494  100176 SRTR010.       MOVE 604 TO ERR1.  MOVE 10 TO ERR2.  PERFORM ERRP.
000495  100177 SRTR011.       MOVE 604 TO ERR1.  MOVE 11 TO ERR2.  PERFORM ERRP.
000496  100178 SRTR012.       MOVE 604 TO ERR1.  MOVE 12 TO ERR2.  PERFORM ERRP.
000497  100179*     RETURN    TO   CALLER.
000498  100180 SRTR015.             EXIT.
000499  100181*-------------------------------------------------------------------*
000500  100182 ACCOUNT  SECTION.
000501  100183*
000502  100184*     COUNT
000503  100185*
000504  100186 COUNTR. ADD  1     TO   CNT.
000505  100187 COUNTO. EXIT.
000506  100188*
000507  100189*     GCOUNT
000508  100190*
000509  100191 GCOUNT. ADD  1     TO   GCNTO.
000510  100192 GCOUNTO.      EXIT.
000511  100193*
000512  100194*     OUTPUT COUNT
000513  100195 GCOUNTP.  ADD 1 TO GCNT.
000514  100196 GCOUNTP-EXIT. EXIT.
000515  100197*-------------------------------------------------------------------*
000516  100198*
000517  100199*     S-SRSLT         I    NODE NUMBER
000518  100200*                     REN  RECORD-TYPE
000519  100201*                     ITN  RESULT ATTRIBUTE NAME
000520  100202*                     J    POINTER TO THE VALUE
000521  100203*
000522  100204 S-SRSLT            SECTION.
000523  100205          ADD  1    TO   CSRSLT.
000524  100206          IF  CSRSLT  >  MAXSRSLT     GO TO SRSLT2.
000525  100207*
000526  100208          MOVE  I       TO  NODE-NO   IN SRSLT (CSRSLT).
000527  100209          MOVE  REN     TO  REC-TYPE  IN SRSLT (CSRSLT).
000528  100210          MOVE  ITN     TO  ATT-NAME  IN SRSLT (CSRSLT).
000529  100211          MOVE  J       TO  NPT       IN SRSLT (CSRSLT).
000530  100212          MOVE  0       TO  CVPT      IN SRSLT (CSRSLT).
000531  100213                        GO TO SRSLT3.
000532  100214 SRSLT2.   MOVE 608 TO ERR1.  MOVE  1 TO ERR2.  PERFORM ERRP.
000533  100215 SRSLT3. EXIT.
000534  100241 ERRP             SECTION.
000535  100242 ERRP1.
000536  100243      DISPLAY 'PATTERN NUMB. = '  ERR1  ', ERROR CODE = '  ERR2.
000537  100244      STOP RUN.
000538  100245 ERR-EXIT.     EXIT.
000539  100246*-------------------------------------------------------------------*
000540  100247*
000541  100248*     NEXT1     INPUT  I = THE NODE#
000542  100249*               OUTPUT  I = THE COLUMN OF SRSLT
000543  100250*                                 IN WHICH I IS STORED
000544  100251 NEXT1            SECTION.
000545  100252 NEXT0.
000546  100253          ADD 1 TO I.
000547  100254          IF  NPT IN SRSLT (I)  NOT =  II
000548  100255                                    GO TO NEXT0.
000549  100256 NEXT1.  EXIT.
000550  100257*-------------------------------------------------------------------*
000551  101217*
000552  101218*               OUTPUT    PRINT
000553  101219*
000554  101220 OUTPT            SECTION.
000555  101221 OUTPT0.
000556  101222          MOVE 0  TO I J.
000557  101223 OUTPT1. ADD  1 TO J.
000558  102001**** DML-RESULT OUTPUT              (0060)
000559  102002          MOVE 0060 TO II.
000560  102003          PERFORM  NEXT1.
000561  102004*              I = THE COLMN. NUMB. OF SRSLT
000562  102005*                  WHICH 0060 IS STORED
000563  102006          IF  MARK IN STDBKEY (I, J)  =  OFFF     GO TO L0117.
000564  102007*              CLEAR MARK
000565  102008          MOVE OFFF TO MARK IN STDBKEY (I, J).
000566  102009***           FIND PROJECT  DB-KEY IS DBKEY IN STDBKEY (I, J).
000567  102010          MOVE DBKEY IN STDBKEY (I, J)  TO  PGCS OF FCOM.
000568  102011***           GET  PROJECT.
000569  102012          GET  CURRENT.
000570  102013              PERFORM  GCOUNTP.
000571  102014*     SET RESULT     TO   THE  RESULT     RELATION ( RSLT ).
000572  102015          MOVE PNAME IN PROJECT TO PNAME0060 IN RSLT.
000573  102016*
000574  102017 L0117.
000575  102018**** DML-RESULT OUTPUT              (0064)
000576  102019          MOVE 0064 TO II.
000577  102020          PERFORM  NEXT1.
000578  102021*              I = THE COLMN. NUMB. OF SRSLT
000579  102022*                  WHICH 0064 IS STORED
000580  102023          IF  MARK IN STDBKEY (I, J)  =  OFFF     GO TO L0517.
000581  102024*              CLEAR MARK
000582  102025          MOVE OFFF TO MARK IN STDBKEY (I, J).
000583  102026***           FIND EMPLOYEE  DB-KEY IS DBKEY IN STDBKEY (I, J).
000584  102027          MOVE DBKEY IN STDBKEY (I, J)  TO  PGCS OF FCOM.
000585  102028***           GET  EMPLOYEE.
000586  102029          GET  CURRENT.
000587  102030              PERFORM  GCOUNTP.
000588  102031*     SET RESULT     TO   THE  RESULT     RELATION ( RSLT ).
```

```
000589  102032          MOVE EFNAME IN EMPLOYEE TO EFNAMEO64 IN RSLT.
000590  102033*
000591  102034* L0517.
000592  102224          DISPLAY  RSLT.
000593  102225          WRITE  RSLT  INVALID KEY
000594  102226                    DISPLAY 'RESULT FILE OVERFLOW'
000595  102227                    GO TO OUTPT10.
000596  102228          MOVE  SPACE  TO  RSLT.
000597  102229          IF  J  NOT <  CVPT IN SRSLT (I)
000598  102230                                    GO TO OUTPT10.
000599  102231          MOVE  0  TO  I.
000600  102232          GO  TO  OUTPT1.
000601  102233*            END OF OUTPUT
000602  102234* OUTPT10.    MOVE 0 TO I.
000603  102235 OUTPT11.    ADD  1 TO I.
000604  102236             IF  I  >  CSRSLT  GO TO OUTPT12.
000605  102237          MOVE  0  TO  CVPT IN SRSLT (I).
000606  102238             GO TO OUTPT11.
000607  102239 OUTPT12.
000608  102240          EXIT.
```

(PNAME ( ) (EFNAME))
MAN MACHINE USER INTERFACE

RSS

DISTRIBUTED DATABASE SYSTEMS

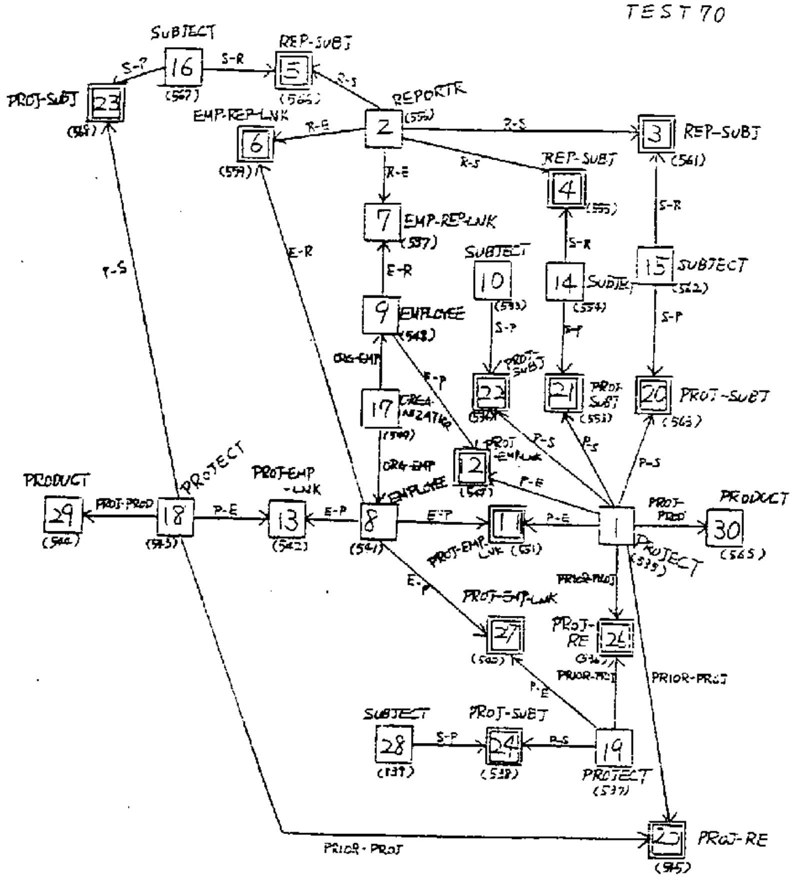
JIPNET

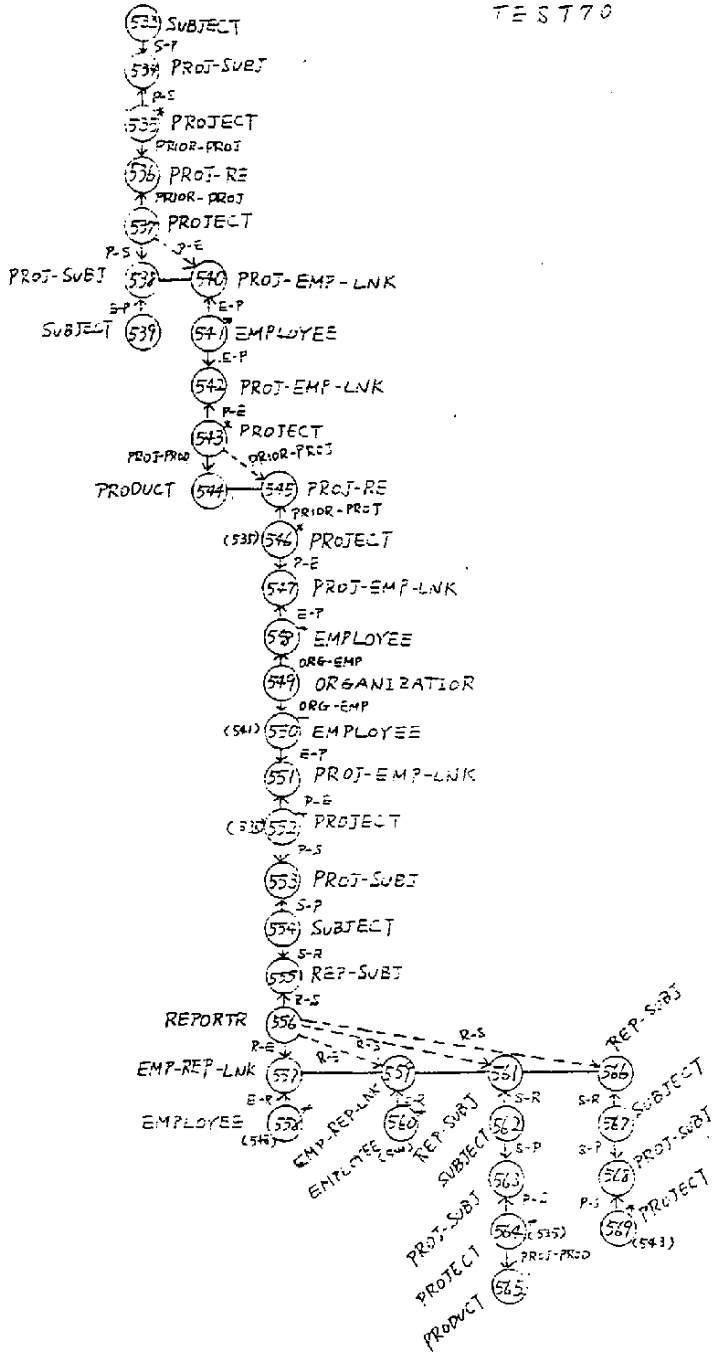
```
THE NUMBER OF LOCATED  OCCURRENCES  = 00000104
THE NUMBER OF ACCESSED OCCURRENCES  = 00000000
THE NUMBER OF OUTPUT   OCCURRENCES  = 00000023
THE ELAPSE TIME FOR ACCESSING = 00006154
THE ELAPSE TIME FOR OUTPUT    = 00030000
THE TOTAL TIME                = 0000015*
```

TEST 70

```
RANGE (P1, PROJECT) (R, REPORTR) ;
RANGE (RJ1, REP-SUBJ) (RJ2, REP-SUBJ) (RJ3, REP-SUBJ) ;
RANGE (ER1, EMP-REP-LNK) (ER2, EMP-REP-LNK) ;
RANGE (E1, EMPLOYEE) (E2, EMPLOYEE) ;
RANGE (PJ, PROJ-SUBJ) ;
RANGE (PE1, PROJ-EMP-LNK) (PE2, PROJ-EMP-LNK) (PE3, PROJ-EMP-LNK) ;
RANGE (OE1, ORG-EMP) (OE2, ORG-EMP) ;
RANGE (PO1, PROJ-PROD) (PO2, PROJ-PROD) ;
RANGE (P2, PROJECT) (P3, PROJECT) ;
RANGE (PJ1, PROJ-SUBJ) (PJ2, PROJ-SUBJ) (PJ3, PROJ-SUBJ) ;
RANGE (PJ4, PROJ-SUBJ) (PJ5, PROJ-SUBJ) ;
RANGE (PRE1, PROJ-RE) ;
RANGE (PRE2, PROJ-RE) (PE4, PROJ-EMP-LNK) ;
RETRIEVE INTO TEST070(PRE2.PNO, P1.PNAME,PRE1.PRIOR-PR-PNO, P2.PNAME,
        E1.EFNAME, E2.EFNAME)
WHERE P1.PNO        = PO1.PNO            AND P1.PNO          = PRE1.PNO           AND
      P1.PNO        = PJ1.PNO            AND P1.PNO          = PJ2.PNO            AND
      P1.PNO        = PE1.PNO            AND P1.PNO          = PE3.PNO            AND
      P1.PNO        = PRE2.PRIOR-PR-PNO  AND PJ4.SUBJECTN = "COMPUTER NETWORKS" AND
      PJ4.SUBJECTN  = RJ3.SUBJECTN       AND
      PJ2.SUBJECTN  = RJ2.SUBJECTN       AND
      RJ2.SUBJECTN  = "DISTRIBUTED DATABASE SYSTEMS"       AND
      PJ1.SUBJECTN  = RJ1.SUBJECTN       AND RJ1.SUBJECTN = "DATABASE SYSTEMS" AND
      R.RNO         = RJ3.RNO            AND R.RNO           = RJ2.RNO            AND
      R.RNO         = RJ1.RNO            AND R.RNO           = ER1.RNO            AND
      R.RNO         = ER2.RNO            AND
      ER1.AUTH-NO   = 1                  AND ER2.AUTH-NO     = 2                  AND
      E1.ENO        = ER1.ENO            AND E1.ENO          = PE1.ENO            AND
      E1.ENO        = OE1.ENO            AND E1.ENO          = PE4.ENO            AND
      E2.ENO        = ER2.ENO            AND E2.ENO          = PE2.ENO            AND
      E2.ENO        = OE2.ENO            AND
      OE2.ORG-NAME  = OE1.ORG-NAME       AND
      (OE2.ORG-NAME = "JIPDEC"           OR  OE2.ORG-NAME    = "SYSTEM")          AND
      P2.PNO        = PJ4.PNO            AND P2.PNO          =PRE1.PRIOR-PR-PNO   AND
      P2.PNO        = PE3.PNO            AND PE3.ENO         = E1.ENO             AND
      P2.PNO        = PO2.PNO            AND
      P3.PNO        = PRE2.PNO           AND P3.PNO          = PE4.PNO            AND
      P3.PNO        = PJ5.PNO            AND
      P1.PNO        = PJ3.PNO            AND
      PJ3.SUBJECTN  = "FOUR-SCHEMA STRUCTURE"                                     AND
      (PJ5.SUBJECTN = "DATABASE SYSTEMS" OR PJ5.SUBJECTN = "MAN-MACHINE
      USER INTERFACE") ;
```

TEST 70




TEST70
SUBJECT
S-P
PROJ-SUBJ
P-S
PROJECT
PRIOR-PROJ
PROJ-RE
PRIOR-PROJ
PROJECT
P-S
P-E
PROJ-SUBJ
PROJ-EMP-LNK
S-P
E-P
SUBJECT
EMPLOYEE
E-P
PROJ-EMP-LNK
P-E
PROJECT
PROJ-PROD
PRIOR-PROJ
PRODUCT
PROJ-RE
PRIOR-PROJ
PROJECT
P-E
PROJ-EMP-LNK
E-P
EMPLOYEE
ORG-EMP
ORGANIZATION
ORG-EMP
EMPLOYEE
E-P
PROJ-EMP-LNK
P-E
PROJECT
P-S
PROJ-SUBJ
S-P
SUBJECT
S-R
REP-SUBJ
R-S
REPORTR
R-S
R-S
R-S
REP-SUBJ
EMP-REP-LNK
E-R
EMPLOYEE
EMP-REP-LNK
EMPLOYEE
REP-SUBJ
SUBJECT
S-R
S-P
PROJ-SUBJ
P-S
PROJECT
PROJECT
PROJ-PROD
PRODUCT
S-R
SUBJECT
S-P
PROJ-SUBJ
P-S
PROJECT

実 行 結 果

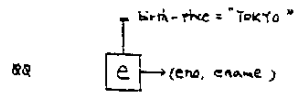
TEST ﾌﾟﾛｸﾞ 3/18

```
( ( PNO PNAME : ( ) ( EENAME ( PNO PNAME ( ) ( ( EENAME ( ( ( ( ( ) ( ) ( ( ( ) ) ( ( ( ) ) ) ) ( ( ( ) ) ) ) ) ) ) ) ) ) ) ) ; )
DISTRIBUTED DATABASE SYSTEMS                                                    DDBS80  LIST
        DISTRIBUTED DATABASE SYSTEMS                                                    DDBS80

THE NUMBER OF LOCATED  OCCURRENCES  = 00002654
THE NUMBER OF ACCESSED OCCURRENCES  = 00001169
THE NUMBER OF OUTPUT   OCCURRENCES  = 00000003
THE ELAPSE TIME FOR ACCESSING = 00002546
THE ELAPSE TIME FOR OUTPUT    = 00000000
THE TOTAL TIME                = 00002546
```

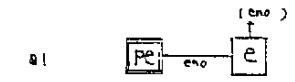
2. テスト問合せ

テスト#

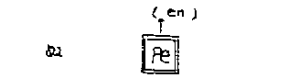
```
∞   RANGE    ( S, SUBJECT ) ( P, PROJECT ) ( E, EMPLOYEE ) ;
    RANGE  (PS, PROJ-SUBJ) ( PE, PROJ-EMP-LNK);
    RETRIEVE INTO TESTOOO ( E.ENO, E.ENAME )
    WHERE
         E.BIRTH-PLACE = "TOKYO" ;
```



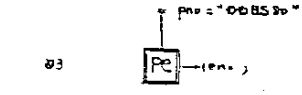

```
01  RANGE    ( S, SUBJECT ) ( P, PROJECT ) ( E, EMPLOYEE ) ;
    RANGE  (PS, PROJ-SUBJ) ( PE, PROJ-EMP-LNK);
    RETRIEVE INTO TESTOO1 ( E.ENO )
    WHERE
         P.PNO          = PE.PNO              AND  PE.ENO        = E.ENO ;
```



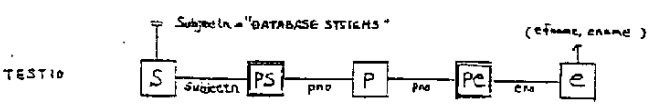

```
02  RANGE    ( S, SUBJECT ) ( P, PROJECT ) ( E, EMPLOYEE ) ;
    RANGE  (PS, PROJ-SUBJ) ( PE, PROJ-EMP-LNK);
    RETRIEVE INTO TESTOO2 ( PE.ENO ) ;
```




```
03  RANGE    ( S, SUBJECT ) ( P, PROJECT ) ( E, EMPLOYEE ) ;
    RANGE  (PS, PROJ-SUBJ) ( PE, PROJ-EMP-LNK);
    RETRIEVE INTO TESTOO3 ( PE.ENO )
    WHERE
         PE.PNO         = "DDBS80" ;
```



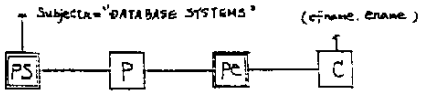

```
10  RANGE    ( S, SUBJECT ) ( P, PROJECT ) ( E, EMPLOYEE ) ;
    RANGE  (PS, PROJ-SUBJ) ( PE, PROJ-EMP-LNK);
    RETRIEVE INTO TESTO10 ( E.ENAME, E.ENAME )
    WHERE
         E.ENO          = PE.ENO              AND  PE.PNO        = P.PNO          AND
         P.PNO          = PS.PNO              AND
         PS.SUBJECTN    = S.SUBJECTN          AND
         S.SUBJECTN     = "DATABASE SYSTEMS" ;
```



11

```
RANGE    ( S, SUBJECT ) ( P, PROJECT ) ( E, EMPLOYEE ) ;
RANGE  (PS, PROJ-SUBJ) ( PE, PROJ-EMP-LNK);
RETRIEVE INTO TEST011 ( E.EFNAME, E.ENAME )
WHERE
     E.ENO          = PE.ENO              AND  PE.PNO        = P.PNO          AND
     P.PNO          = PS.PNO              AND
     PS.SUBJECTN    = "DATABASE SYSTEMS" ;
```

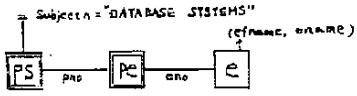
TEST 11



12

```
RANGE    ( S, SUBJECT ) ( P, PROJECT ) ( E, EMPLOYEE ) ;
RANGE  (PS, PROJ-SUBJ) ( PE, PROJ-EMP-LNK);
RETRIEVE INTO TEST012 ( E.EFNAME, E.ENAME )
WHERE
     E.ENO          = PE.ENO              AND  PE.PNO        = PS.PNO         AND
     PS.SUBJECTN    = "DATABASE SYSTEMS" ;
```

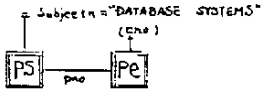
TEST 12



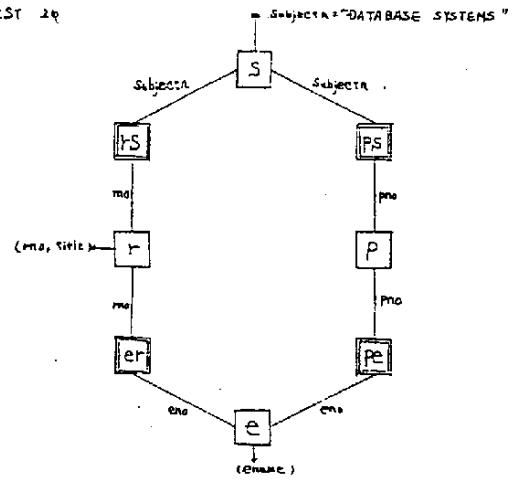
13

```
RANGE    ( S, SUBJECT ) ( P, PROJECT ) ( E, EMPLOYEE ) ;
RANGE  (PS, PROJ-SUBJ) ( PE, PROJ-EMP-LNK);
RETRIEVE INTO TEST013 ( PE.ENO )
WHERE
     PS.SUBJECTN    = "DATABASE SYSTEMS" AND
     PS.PNO         = PE.PNO ;
```

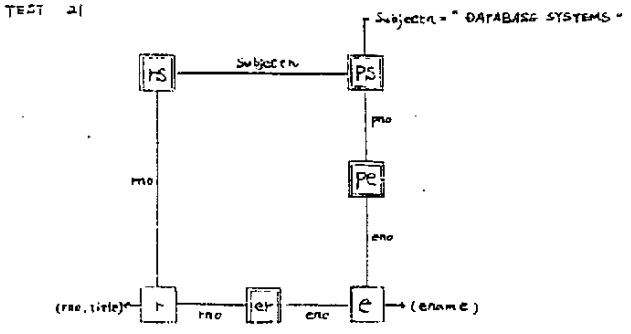
TEST 13



20
```
RANGE    ( P, PROJECT ) ( E, EMPLOYEE ) ( S, SUBJECT ) ;
RANGE    ( R, REPORTR) ;
RANGE  ( PE, PROJ-EMP-LNK) ( ER, EMP-REP-LNK);
RANGE    ( RS, REP-SUBJ ) ( PS, PROJ-SUBJ ) ;
RETRIEVE INTO TEST020 ( E.ENAME, R.RNO, R.TITLE )
WHERE
     S.SUBJECTN     = "DATABASE SYSTEMS" AND
     S.SUBJECTN     = PS.SUBJECTN        AND  PS.PNO     = P.PNO     AND
     P.PNO          = PE.PNO             AND  PE.ENO     = E.ENO     AND
     E.ENO          = ER.ENO             AND  ER.RNO     = R.RNO     AND
     R.RNO          = RS.RNO             AND
     RS.SUBJECTN    = S.SUBJECTN ;
```



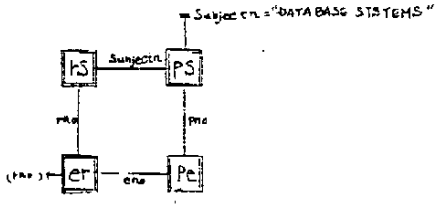
21
```
RANGE    ( P, PROJECT ) ( E, EMPLOYEE ) ( S, SUBJECT ) ;
RANGE    ( R, REPORTR) ;
RANGE  ( PE, PROJ-EMP-LNK) ( ER, EMP-REP-LNK);
RANGE    ( RS, REP-SUBJ ) ( PS, PROJ-SUBJ ) ;
RETRIEVE INTO TEST021 ( E.ENAME, R.RNO, R.TITLE )
WHERE
     PS.SUBJECTN    = "DATABASE SYSTEMS" AND
     PS.PNO         = PE.PNO             AND  PE.ENO     = E.ENO
     E.ENO          = ER.ENO             AND  ER.RNO     = R.RNO
     R.RNO          = RS.RNO             AND
     RS.SUBJECTN    = PS.SUBJECTN ;
```



22

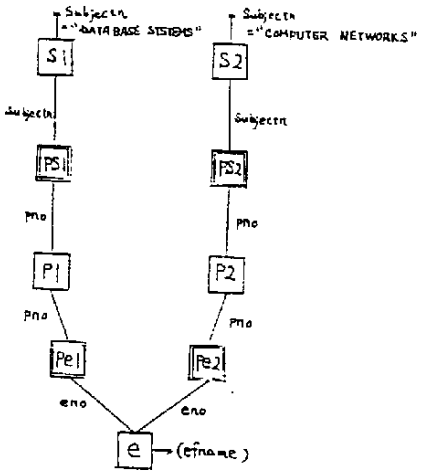
```
RANGE     ( P, PROJECT ) ( E, EMPLOYEE ) ( S, SUBJECT ) ;
RANGE     ( R, REPORTR) ;
RANGE   ( PE, PROJ-EMP-LNK) ( ER, EMP-REP-LNK);
RANGE     ( RS, REP-SUBJ ) ( PS, PROJ-SUBJ ) ;
RETRIEVE INTO TEST022 ( ER.RNO )
WHERE
     PS.SUBJECTN    = "DATABASE SYSTEMS" AND
     PS.PNO         = PE.PNO             AND  PE.ENO      = ER.ENO      AND
     ER.RNO         = RS.RNO             AND
     RS.SUBJECTN    = PS.SUBJECTN ;
```

TEST 22



30

TEST30

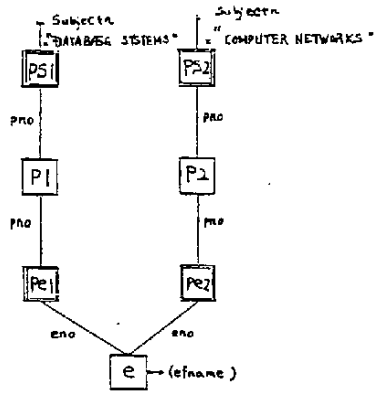



```
RANGE     ( S1, SUBJECT ) ( S2, SUBJECT ) ;
RANGE     ( P1, PROJECT ) ( P2, PROJECT ) ;
RANGE     ( PS1, PROJ-SUBJ ) ( PS2, PROJ-SUBJ ) ;
RANGE     ( PE1, PROJ-EMP-LNK  ) ( PE2, PROJ-EMP-LNK  );
RANGE     ( E, EMPLOYEE ):
RETRIEVE INTO TEST030 ( E.ENAME )
WHERE
     S1.SUBJECTN    = "DATABASE SYSTEMS" AND
     S2.SUBJECTN    = "COMPUTER NETWORKS" AND
     S1.SUBJECTN    = PS1.SUBJECTN       AND  PS1.PNO     = P1.PNO      AND
     P1.PNO         = PE1.PNO            AND
     S2.SUBJECTN    = PS2.SUBJECTN       AND  PS2.PNO     = P2.PNO      AND
     P2.PNO         = PE2.PNO            AND
     PE1.ENO        = E.ENO              AND  PE2.ENO     = E.ENO ;
```

31

```
RANGE    ( S1, SUBJECT ) ( S2, SUBJECT ) ;
RANGE    ( P1, PROJECT ) ( P2, PROJECT ) ;
RANGE    ( PS1, PROJ-SUBJ ) ( PS2, PROJ-SUBJ ) ;
RANGE    ( PE1, PROJ-EMP-LNK  ) ( PE2, PROJ-EMP-LNK  );
RANGE    ( E, EMPLOYEE );
RETRIEVE INTO TEST031 ( E.EFNAME )
WHERE
     PS1.SUBJECTN  = "DATABASE SYSTEMS"   AND
     PS2.SUBJECTN  = "COMPUTER NETWORKS"  AND
     PS1.PNO       = P1.PNO               AND  PS2.PNO     = P2.PNO      AND
     P1.PNO        = PE1.PNO              AND  P2.PNO      = PE2.PNO     AND
     PE1.ENO       = E.ENO                AND  PE2.ENO     = E.ENO ;
```

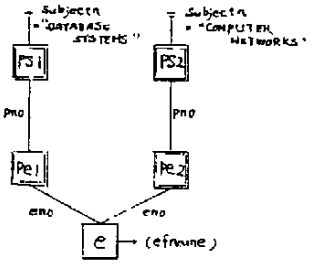
TEST 31



32

```
RANGE    ( S1, SUBJECT ) ( S2, SUBJECT ) ;
RANGE    ( P1, PROJECT ) ( P2, PROJECT ) ;
RANGE    ( PS1, PROJ-SUBJ ) ( PS2, PROJ-SUBJ ) ;
RANGE    ( PE1, PROJ-EMP-LNK  ) ( PE2, PROJ-EMP-LNK  );
RANGE    ( E, EMPLOYEE );
RETRIEVE INTO TEST032 ( E.EFNAME )
WHERE
     PS1.SUBJECTN  = "DATABASE SYSTEMS"   AND
     PS2.SUBJECTN  = "COMPUTER NETWORKS"  AND
     PS1.PNO       = PE1.PNO              AND  PS2.PNO     = PE2.PNO     AND
     PE1.ENO       = E.ENO                AND  PE2.ENO     = E.ENO ;
```

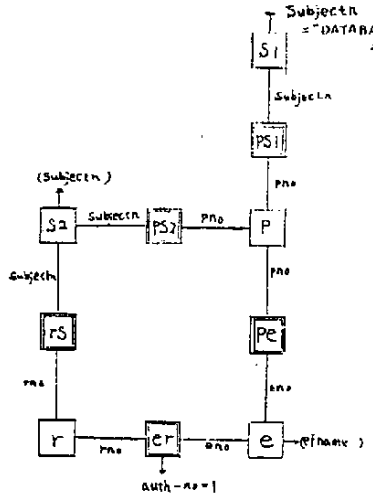
TEST 32



40

```
RANGE    ( P, PROJECT ) ( E, EMPLOYEE ) ( R, REPORTR) ;
RANGE    ( S1, SUBJECT ) ( S2, SUBJECT ) ;
RANGE    ( PS1, PROJ-SUBJ ) ( PE, PROJ-EMP-LNK  ) ( PS2, PROJ-SUBJ ) ;
RANGE    ( ER, EMP-REP-LNK  ) ( RS, REP-SUBJ ) ;
RETRIEVE INTO TEST040 ( E.ENO, E.EFNAME, S2.SUBJECTN )
WHERE
     S1.SUBJECTN      = "DATABASE SYSTEMS" AND
     S1.SUBJECTN      = PS1.SUBJECTN       AND  PS1.PNO    = P.PNO      AND
     P.PNO            = PE.PNO             AND  PE.ENO     = E.ENO      AND
     E.ENO            = ER.ENO             AND  ER.RNO     = R.RNO      AND
     ER.AUTH-NO       = 1                  AND
     R.RNO            = RS.RNO             AND
     RS.SUBJECTN      = S2.SUBJECTN        AND  S2.SUBJECTN = PS2.SUBJECTN  AND
     PS2.PNO          = P.PNO ;
```

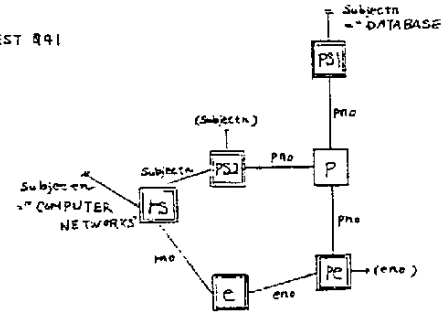
TEST 040



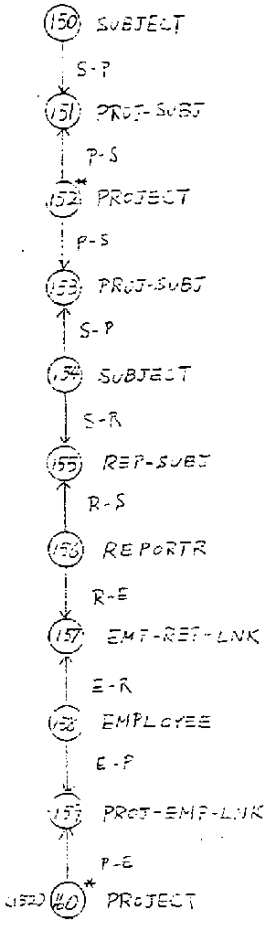
41

```
RANGE    ( P, PROJECT ) ( E, EMPLOYEE ) ( R, REPORTR) ;
RANGE    ( S1, SUBJECT ) ( S2, SUBJECT ) ;
RANGE    ( PS1, PROJ-SUBJ ) ( PE, PROJ-EMP-LNK  ) ( PS2, PROJ-SUBJ ) ;
RANGE    ( ER, EMP-REP-LNK  ) ( RS, REP-SUBJ ) ;
RETRIEVE INTO TEST041 (PE.ENO, PS2.SUBJECTN )
WHERE
     PS1.SUBJECTN      = "DATABASE SYSTEMS" AND
     PS1.PNO           = P.PNO              AND  P.PNO      = PE.PNO     AND
     PE.ENO            = ER.ENO             AND  ER.AUTH-NO = 1          AND
     ER.RNO            = RS.RNO             AND  RS.SUBJECTN = PS2.SUBJECTN  AND
      RS.SUBJECTN = "COMPUTER NETWORKS"          AND
     PS2.PNO           = P.PNO ;
```

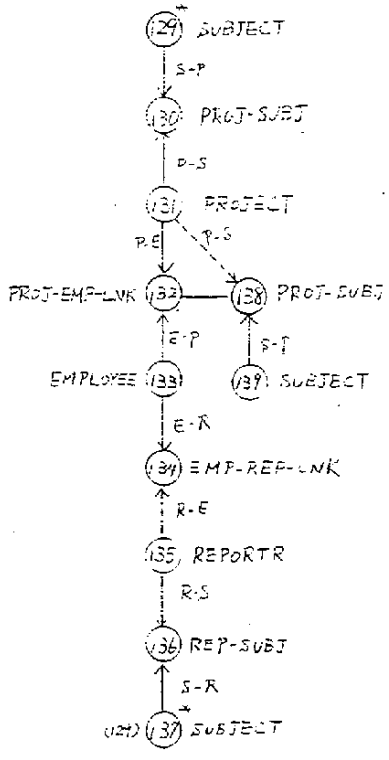
TEST 041



TEST 4A

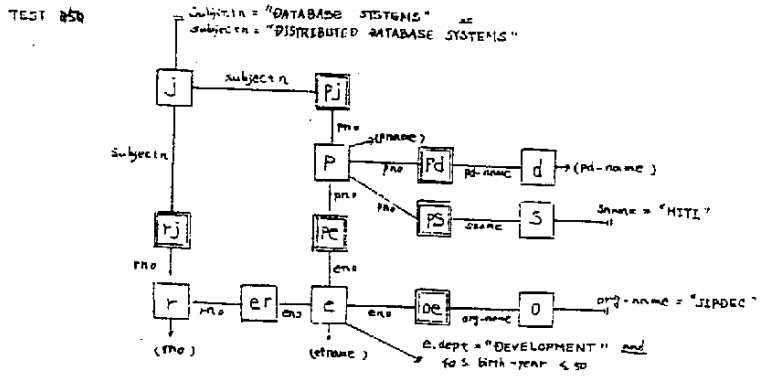


TEST 4I

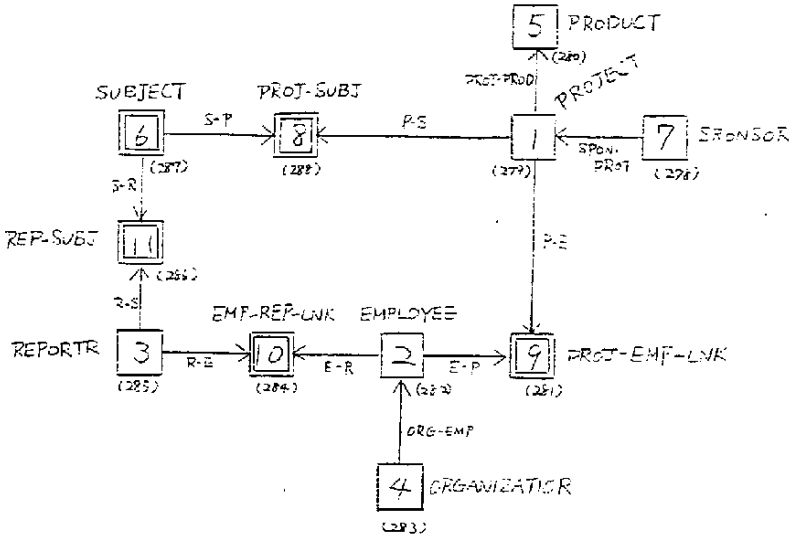


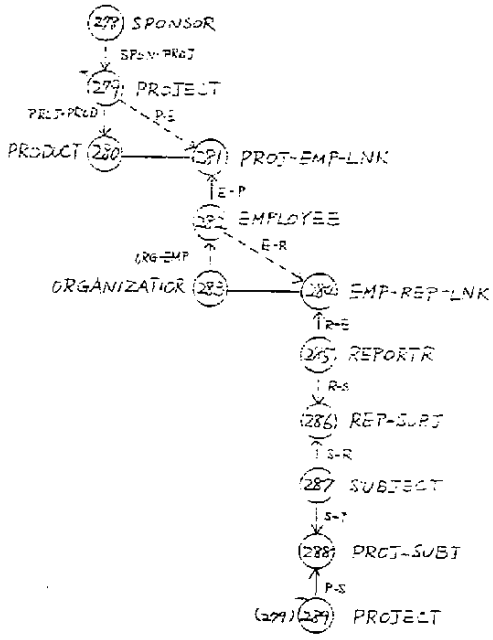
50

```
RANGE
     ( P, PROJECT ) ( E, EMPLOYEE ) ( R, REPORTR )
     ( O, ORGANIZATION ) ( D, PRODUCT ) ;
RANGE ( J, SUBJECT ) ( S, SPONSOR ) ;
RANGE ( PJ, PROJ-SUBJ ) ( PE, PROJ-EMP-LNK)
      ( ER, EMP-REP-LNK ) ( RJ, REP-SUBJ )
      ( OE, ORG-EMP )   ( PD, PROJ-PROD )
      ( SP, SPON-PROJ ) ;
RETRIEVE INTO TEST050 ( E.ENAME, P.PNAME, D.PD-NAME, R.RNO )
WHERE
     P.PNO          = PJ.PNO          AND  PJ.SUBJECTN  = J.SUBJECTN   AND
     J.SUBJECTN     = RJ.SUBJECTN     AND  RJ.RNO       = R.RNO        AND
     R.RNO          = ER.RNO          AND  ER.ENO       = E.ENO        AND
     E.ENO          = PE.ENO          AND
     P.PNO          = SP.PNO          AND  SP.SNAME     = S.SNAME      AND
     E.ENO          = OE.ORG-NAME     AND  OE.ORG-NAME  = O.ORG-NAME   AND
     P.PNO            = PE.PNO          AND
     O.ORG-NAME     = "JIPDEC"        AND
     E.BIRTH-YEAR   LE 50             AND  E.BIRTH-YEAR GE 40          AND
     S.SNAME        = "MITI"          AND
     E.DEPT         = "DEVELOPMENT"   AND
     ( J.SUBJECTN   = "DATABASE SYSTEMS" OR
       J.SUBJECTN   = "DISTRIBUTED DATABASE SYSTEMS" )                 AND
     P.PNO          = PD.PNO          AND  PD.PD-NO     = D.PD-NO    ;
```



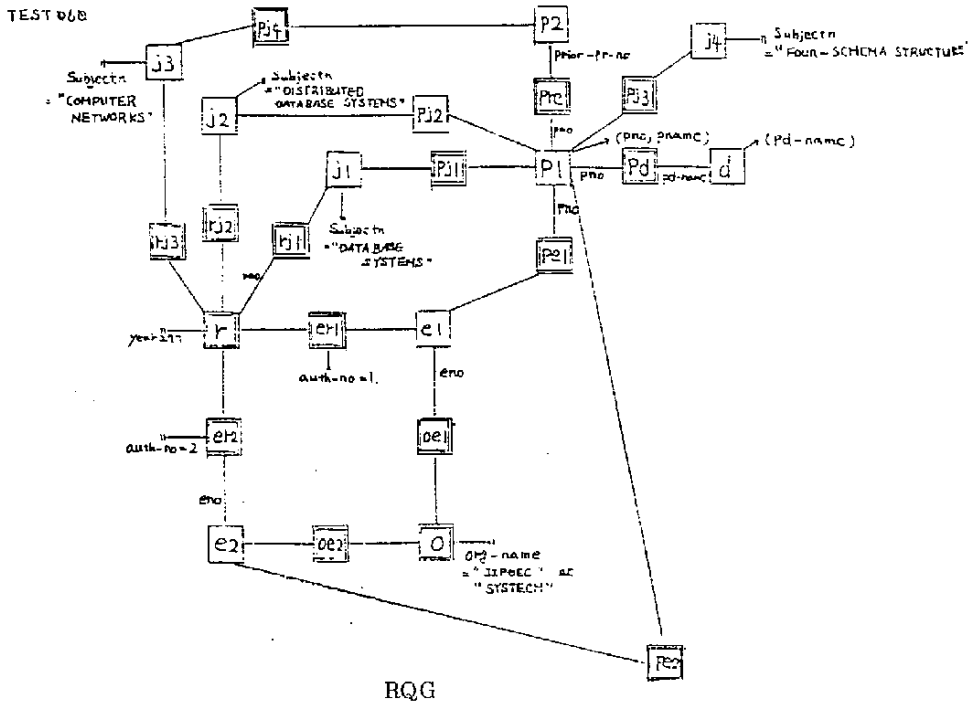
TEST 5 0


5 PRODUCT
(280)
PROJECT
PROJ-PROD
SUBJECT
PROJ-SUBJ
6
S-P
8
P-S
1
7 SPONSOR
SPON-PROJ
(287)
(288)
(279)
(278)
S-R
REP-SUBJ
11
(286)
P-E
R-S
EMP-REP-LNK
EMPLOYEE
REPORTR
3
R-E
10
E-R
2
E-P
9
PROJ-EMP-LNK
(285)
(284)
(282)
(281)
ORG-EMP
4 ORGANIZATION
(283)

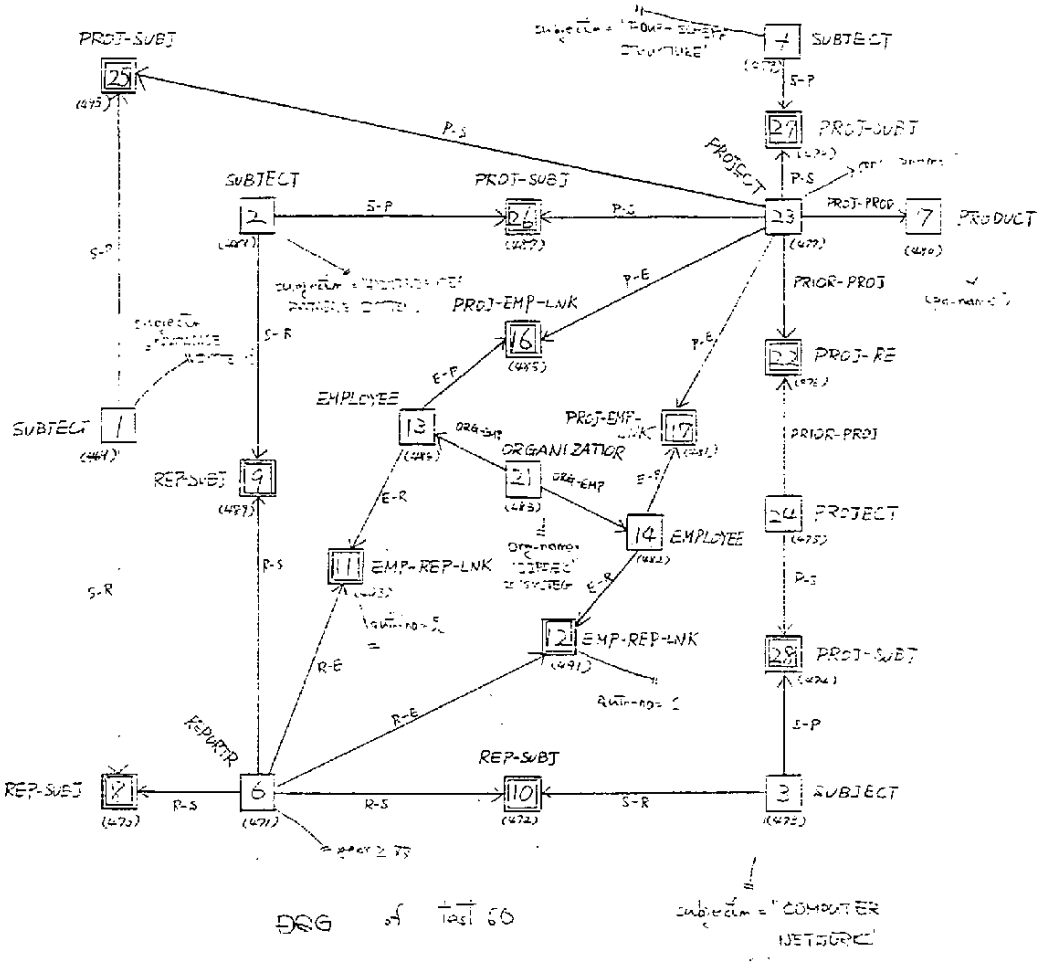

278 SPONSOR
SPON-PROJ
279 PROJECT
PROJ-PROD
P-S
PRODUCT 280
281 PROJ-EMP-LNK
E-P
282 EMPLOYEE
E-R
ORG-EMP
ORGANIZATION 283
284 EMP-REP-LNK
R-E
285 REPORTR
R-S
286 REP-SUBJ
S-R
287 SUBJECT
S-P
288 PROJ-SUBJ
P-S
279 289 PROJECT

60

```
#00020 RANGE (J1, SUBJECT) (J2, SUBJECT) (J3, SUBJECT) (J4, SUBJECT) ;
#00030 RANGE (P, PROJECT) (R, REPORTR) ;
#00040 RANGE (D, PRODUCT) ;
#00050 RANGE (RJ1, REP-SUBJ) (RJ2, REP-SUBJ) (RJ3, REP-SUBJ) ;
#00060 RANGE (ER1, EMP-REP-LNK) (ER2, EMP-REP-LNK) ;
#00070 RANGE (E1, EMPLOYEE) (E2, EMPLOYEE) ;
#00080 RANGE (PJ, PROJ-SUBJ) ;
#00090 RANGE (PE1, PROJ-EMP-LNK) (PE2, PROJ-EMP-LNK) ;
#00100 RANGE (PD, PROJ-PROD) ;
#00110 RANGE (DE1, ORG-EMP) (DE2, ORG-EMP) (O, ORGANIZATION) ;
#00120 RANGE (PRE, PROJ-RE) ;
#00130 RANGE (P1, PROJECT) (P2, PROJECT) ;
#00140 RANGE (PJ1, PROJ-SUBJ) (PJ2, PROJ-SUBJ) (PJ3, PROJ-SUBJ) ;
#00150 RANGE (PJ4, PROJ-SUBJ) ;
#00160 RETRIEVE INTO TEST060 (P1.PNO, P1.PNAME, D.PD-NAME)
#00170 WHERE J1.SUBJECTN = "DATABASE SYSTEMS" AND
#00180       J1.SUBJECTN = PJ1.SUBJECTN        AND PJ1.PNO = P1.PNO AND
#00190       J2.SUBJECTN = "DISTRIBUTED DATABASE SYSTEMS"          AND
#00200       J2.SUBJECTN = PJ2.SUBJECTN        AND PJ2.PNO = P1.PNO AND
#00210       J1.SUBJECTN = RJ1.SUBJECTN        AND J2.SUBJECTN     = RJ2.SUBJECTN   AND
#00220       RJ1.RNO     = R.RNO               AND RJ2.RNO = R.RNO  AND
#00230       R.RNO       = ER1.RNO             AND ER1.AUTH-NO     = 1              AND
#00240       R.RNO       = ER2.RNO             AND ER2.AUTH-NO     = 2              AND
#00250       ER1.ENO     = E1.ENO              AND E1.ENO          = PE1.ENO        AND
#00260       PE1.PNO     = P1.PNO              AND
#00270       ER2.ENO     = E2.ENO              AND E2.ENO          = PE2.ENO        AND
#00280       PE2.PNO     = P1.PNO              AND
#00290       E1.ENO      = DE1.ENO             AND DE1.ORG-NAME    = O.ORG-NAME     AND
#00300       E2.ENO      = DE2.ENO             AND DE2.ORG-NAME    = O.ORG-NAME     AND
#00310       O.ORG-NAME  = "MIT"               AND
#00320       P1.PNO      = PD.PNO              AND PD.PD-NO        = D.PD-NO        AND
#00330       P1.PNO      = PJ3.PNO             AND PJ3.SUBJECTN    = J4.SUBJECTN    AND
#00340       J4.SUBJECTN = "FOUR-SCHEMA STRUCTURE"                  AND
#00350       P1.PNO      = PRE.PNO             AND PRE.PRIOR-PR-PNO = P2.PNO         AND
#00360       P2.PNO      = PJ4.PNO             AND PJ4.SUBJECTN    = J3.SUBJECTN    AND
#00370       J3.SUBJECTN = "COMPUTER NETWORKS" AND
#00380       J3.SUBJECTN = RJ3.SUBJECTN        AND RJ3.RNO         = R.RNO          AND
#00390       R.YEAR      GE 77                 AND
#00400     (D.ORG-NAME   = "JIPDEC"            OR  D.ORG-NAME      = "SYSTECH") ;
```

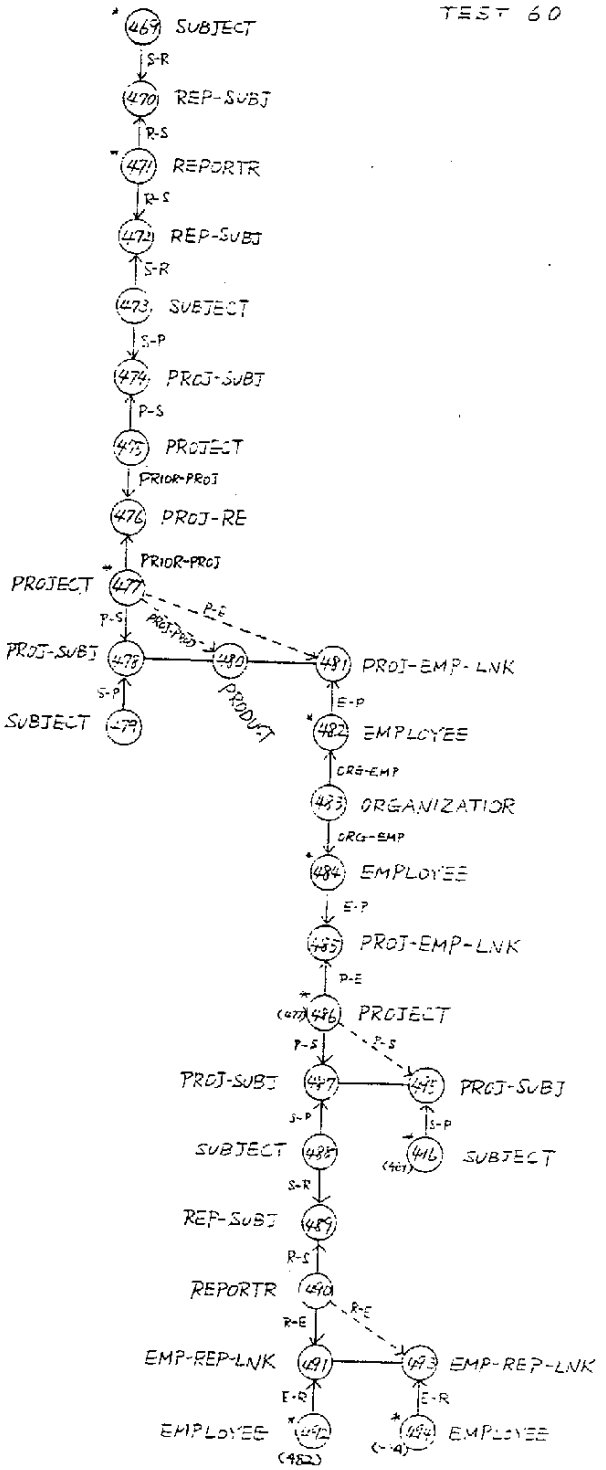


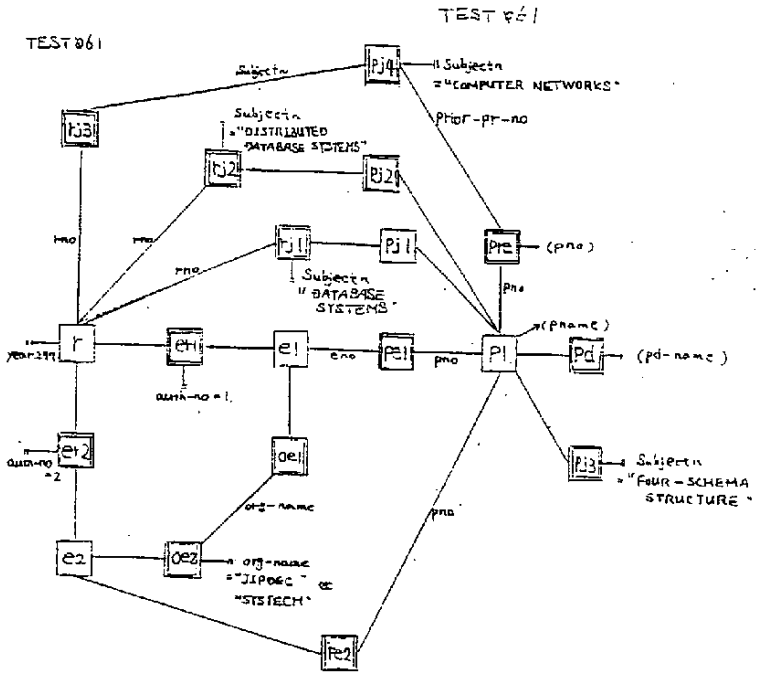
RQG



DBG of Test 60

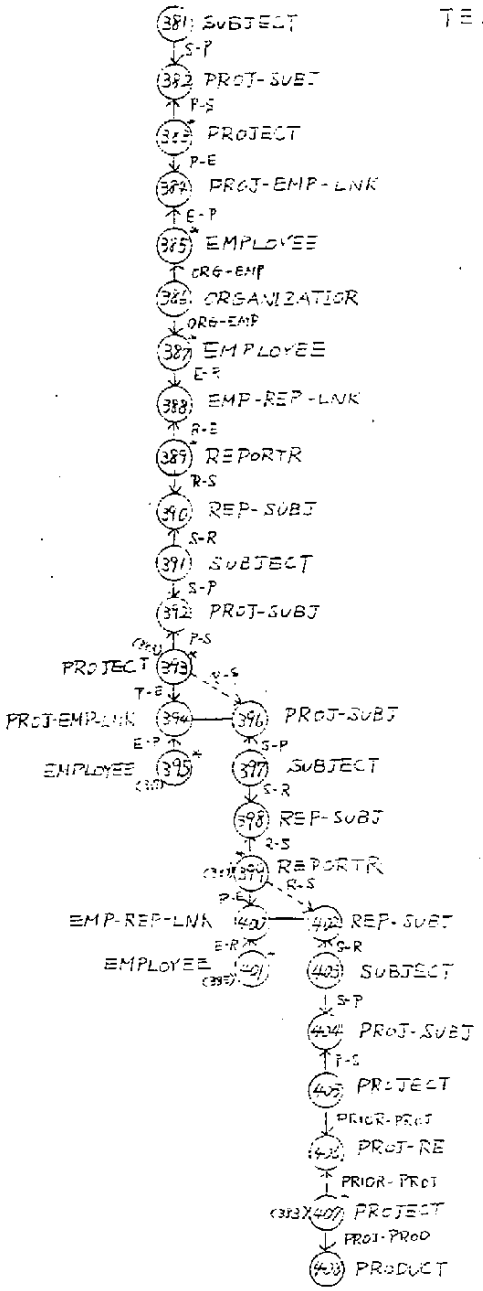
TEST 60


469 SUBJECT
S-R
470 REP-SUBJ
R-S
471 REPORTR
R-S
472 REP-SUBJ
S-R
473 SUBJECT
S-P
474 PROJ-SUBJ
P-S
475 PROJECT
PRIOR-PROJ
476 PROJ-RE
PRIOR-PROJ
PROJECT 477
P-S
P-E
PROJ-PROJ
PROJ-SUBJ 478
480
481 PROJ-EMP-LNK
S-P
PROJECT
SUBJECT 479
E-P
482 EMPLOYEE
ORG-EMP
483 ORGANIZATION
ORG-EMP
484 EMPLOYEE
E-P
485 PROJ-EMP-LNK
P-E
(477) 486 PROJECT
P-S
P-S
PROJ-SUBJ 487
495 PROJ-SUBJ
S-P
S-P
SUBJECT 488
(461) 416 SUBJECT
S-R
REP-SUBJ 489
R-S
REPORTR 490
R-E
R-E
EMP-REP-LNK 491
493 EMP-REP-LNK
E-R
E-R
EMPLOYEE (482) 492
(-4) 494 EMPLOYEE

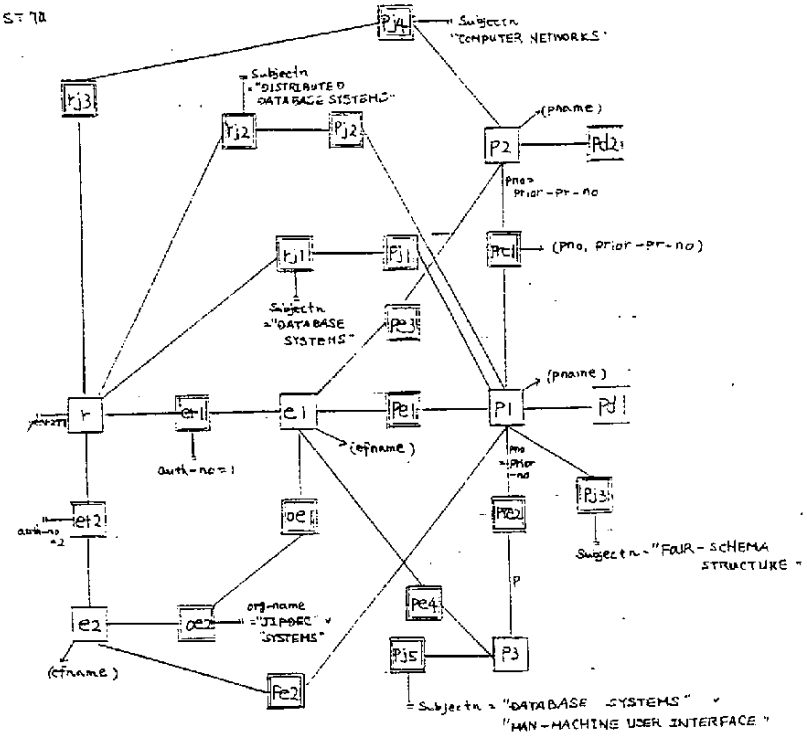



```
#00020 RANGE (J1, SUBJECT) (J2, SUBJECT) (J3, SUBJECT) (J4, SUBJECT) ;
#00030 RANGE (P, PROJECT) (R, REPORTR) ;
#00040 RANGE (D, PRODUCT) ;
#00050 RANGE (RJ1, REP-SUBJ) (RJ2, REP-SUBJ) (RJ3, REP-SUBJ) ;
#00060 RANGE (ER1, EMP-REP-LNK) (ER2, EMP-REP-LNK) ;
#00070 RANGE (E1, EMPLOYEE) (E2, EMPLOYEE) ;
#00080 RANGE (PJ, PROJ-SUBJ) ;
#00090 RANGE (PE1, PROJ-EMP-LNK) (PE2, PROJ-EMP-LNK) ;
#00100 RANGE (PD, PROJ-PROD) ;
#00110 RANGE (OE1, ORG-EMP) (OE2, ORG-EMP) (O, ORGANIZATION) ;
#00120 RANGE (PRE, PROJ-RE) ;
#00130 RANGE (P1, PROJECT) (P2, PROJECT) ;
#00140 RANGE (PJ1, PROJ-SUBJ) (PJ2, PROJ-SUBJ) (PJ3, PROJ-SUBJ) ;
#00150 RANGE (PJ4, PROJ-SUBJ) ;
#00160 RETRIEVE INTO TEST06 (PRE.PNO, P1.PNAME, PD.PD-NO)
#00170 WHERE P1.PNO = PD.PNO AND P1.PNO = PRE.PNO AND
#00180 P1.PNO = PJ1.PNO AND P1.PNO = PJ2.PNO AND
#00190 P1.PNO = PE1.PNO AND P1.PNO = PE2.PNO AND
#00200 PRE.PRIOR-PR-PNO= PJ4.PNO AND PJ4.SUBJECTN = "COMPUTER NETWORKS"AND
#00210 PJ4.SUBJECTN = RJ3.SUBJECTN AND
#00220 PJ2.SUBJECTN = RJ2.SUBJECTN AND
#00230 RJ2.SUBJECTN = "DISTRIBUTED DATABASE SYSTEMS" AND
#00240 PJ1.SUBJECTN = RJ1.SUBJECTN AND RJ1.SUBJECTN = "DATABASE SYSTEMS" AND
#00250 R.RNO = RJ3.RNO AND R.RNO = RJ2.RNO AND
#00260 R.RNO = RJ1.RNO AND R.RNO = ER1.RNO AND
#00270 R.RNO = ER2.RNO AND
#00280 ER1.AUTH-NO = 1 AND ER2.AUTH-NO = 2 AND
#00290 E1.ENO = ER1.ENO AND E1.ENO = PE1.ENO AND
#00300 E1.ENO = OE1.ENO AND
#00310 E2.ENO = ER2.ENO AND E2.ENO = PE2.ENO AND
#00320 E2.ENO = OE2.ENO AND
#00330 OE2.ORG-NAME = OE1.ORG-NAME AND
#00340 (OE2.ORG-NAME = "JIPDEC" OR OE2.ORG-NAME = "SYSTECH") AND
#00350 P1.PNO = PJ3.PNO AND
#00360 PJ3.SUBJECTN = "FOUR-SCHEMA STRUCTURE" ;
```

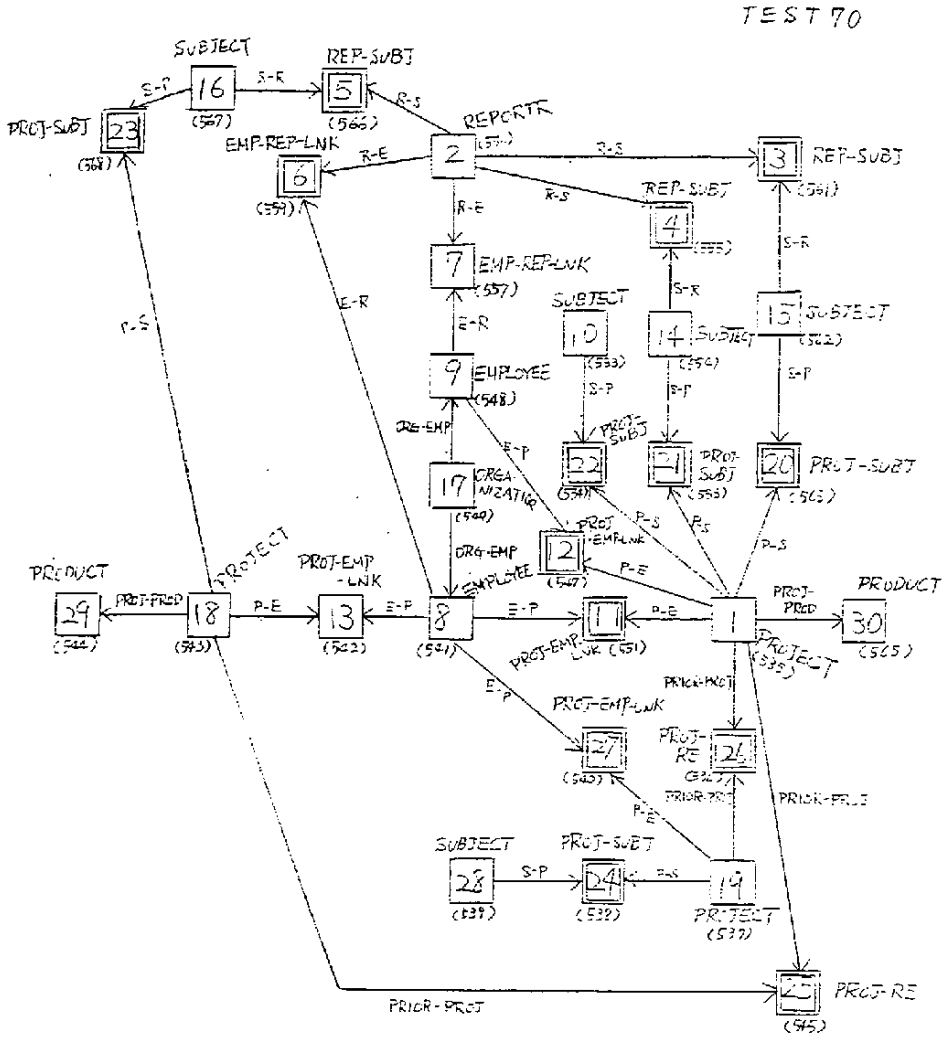
TEST 61


381 SUBJECT
S-P
382 PROJ-SUBJ
P-S
383 PROJECT
P-E
384 PROJ-EMP-LNK
E-P
385 EMPLOYEE
ORG-EMP
386 ORGANIZATION
ORG-EMP
387 EMPLOYEE
E-R
388 EMP-REP-LNK
R-E
389 REPORTR
R-S
390 REP-SUBJ
S-R
391 SUBJECT
S-P
392 PROJ-SUBJ
P-S
PROJECT 393
T-E
PROJ-EMP-LNK 394
E-P
EMPLOYEE 395
396 PROJ-SUBJ
S-P
397 SUBJECT
S-R
398 REP-SUBJ
R-S
REPORTR
P-E
EMP-REP-LNK
E-R
EMPLOYEE 401
402 REP-SUBJ
S-R
403 SUBJECT
S-P
404 PROJ-SUBJ
P-S
405 PROJECT
PRIOR-PROJ
406 PROJ-RE
PRIOR-PROJ
407 PROJECT
PROJ-PROD
408 PRODUCT

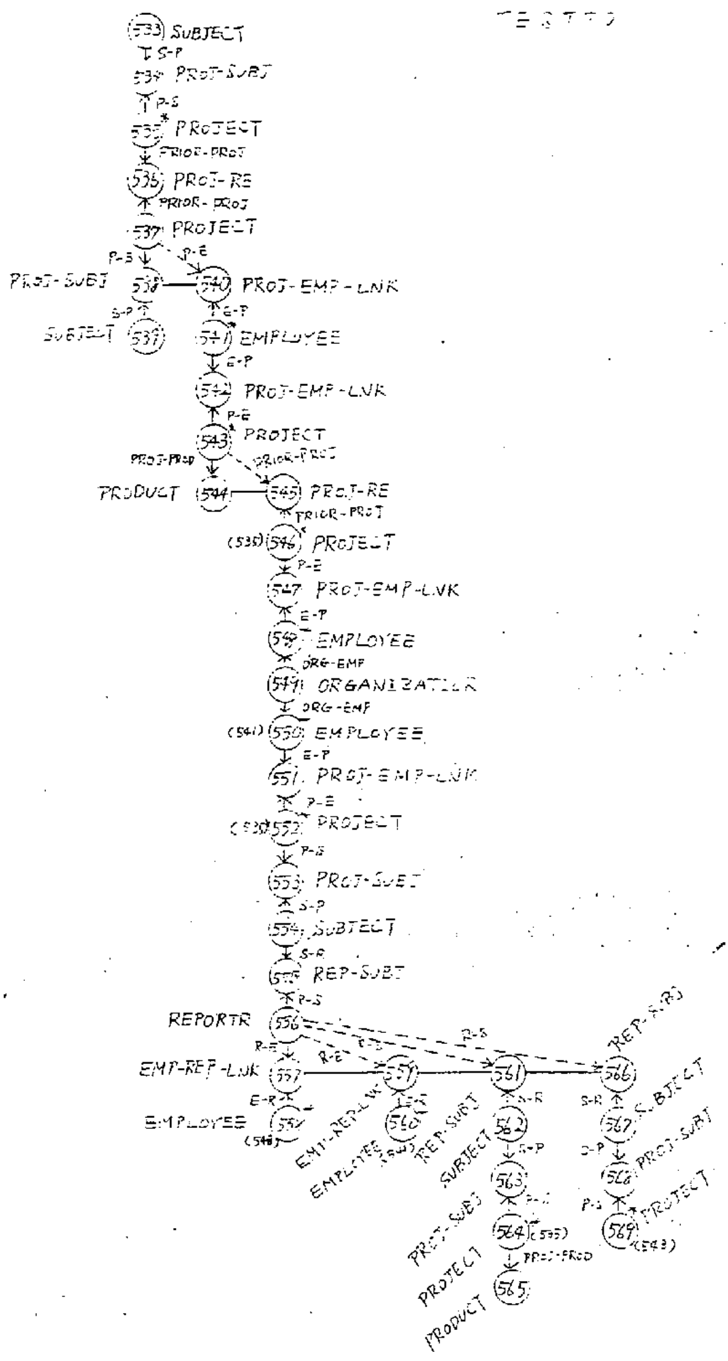
```
#00020 RANGE (P1, PROJECT) (R, REPORTR) ;
#00030 RANGE (RJ1, REP-SUBJ) (RJ2, REP-SUBJ) (RJ3, REP-SUBJ) ;
#00040 RANGE (ER1, EMP-REP-LNK) (ER2, EMP-REP-LNK) ;
#00050 RANGE (E1, EMPLOYEE) (E2, EMPLOYEE) ;
#00060 RANGE (PJ, PROJ-SUBJ) ;
#00070 RANGE (PE1, PROJ-EMP-LNK) (PE2, PROJ-EMP-LNK) (PE3, PROJ-EMP-LNK) ;
#00080 RANGE (OE1, ORG-EMP) (OE2, ORG-EMP) ;
#00090 RANGE (PD1, PROJ-PROD) (PD2, PROJ-PROD) ;
#00100 RANGE (P2, PROJECT) (PJ, PROJECT) ;
#00110 RANGE (PJ1, PROJ-SUBJ) (PJ2, PROJ-SUBJ) (PJ3, PROJ-SUBJ) ;
#00120 RANGE (PJ4, PROJ-SUBJ) (PJ5, PROJ-SUBJ) ;
#00130 RANGE (PRE1, PROJ-RE) ;
#00140 RANGE (PRE2, PROJ-RE) (PE4, PROJ-EMP-LNK) ;
#00150 RETRIEVE INTO TEST070(PRE2.PNO, P1.PNAME,PRE1.PRIOR-PR-PNO), P2.PNAME,
#00160        E1.EFNAME, E2.EFNAME)
#00170 WHERE P1.PNO          = PD1.PNO          AND P1.PNO          = PRE1.PNO    AND
#00180       P1.PNO          = PJ1.PNO          AND P1.PNO          = PJ2.PNO     AND
#00190       P1.PNO          = PE1.PNO          AND P1.PNO          = PE2.PNO     AND
#00200       P1.PNO          = PRE2.PRIOR-PR-PNO AND PJ4.SUBJECTN = "COMPUTER NETWORKS" AND
#00210       PJ4.SUBJECTN    = RJ3.SUBJECTN     AND
#00220       PJ2.SUBJECTN    = RJ2.SUBJECTN     AND
#00230       RJ2.SUBJECTN    = "DISTRIBUTED DATABASE SYSTEMS"          AND
#00240       PJ1.SUBJECTN    = RJ1.SUBJECTN     AND RJ1.SUBJECTN = "DATABASE SYSTEMS" AND
#00250       R.RNO           = RJ3.RNO          AND R.RNO           = RJ2.RNO     AND
#00260       R.RNO           = RJ1.RNO          AND R.RNO           = ER1.RNO     AND
#00270       R.RNO           = ER2.RNO          AND
#00280       ER1.AUTH-NO     = 1                AND ER2.AUTH-NO     = 2           AND
#00290       E1.ENO          = ER1.ENO          AND E1.ENO          = PE1.ENO     AND
#00300       E1.ENO          = OE1.ENO          AND E1.ENO          = PE4.ENO     AND
#00310       E2.ENO          = ER2.ENO          AND E2.ENO          = PE2.ENO     AND
#00320       E2.ENO          = OE2.ENO          AND
#00330       OE2.ORG-NAME    = OE1.ORG-NAME     AND
#00340       (OE2.ORG-NAME   = "JIPDEC"         OR  OE2.ORG-NAME = "SYSTEM")     AND
#00350       P2.PNO          = PJ4.PNO          AND P2.PNO          =PRE1.PRIOR-PR-PNO AND
#00360       P2.PNO          = PE3.PNO          AND PE3.ENO         = E1.ENO      AND
#00370       P2.PNO          = PD2.PNO          AND
#00380       P3.PNO          = PRE2.PNO         AND P3.PNO          = PE4.PNO     AND
#00390       P3.PNO          = PJ5.PNO          AND
#00400       P1.PNO          = PJ3.PNO          AND                              AND
#00410       PJ3.SUBJECTN    = "FOUR-SCHEMA STRUCTURE"
#00420       (PJ5.SUBJECTN   = "DATABASE SYSTEMS" OR PJ5.SUBJECTN = "MAN-MACHINE
#00430       USER INTERFACE") ;
```



RQG

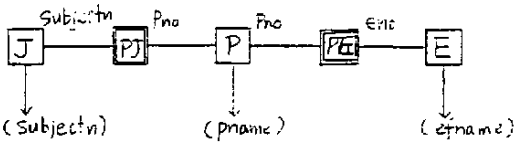

TEST 70
SUBJECT
REP-SUBJ
REPORTR
EMP-REP-LNK
EMPLOYEE
ORGA-NIZATION
PROJECT
PROJ-EMP-LNK
PRODUCT
PROJ-PROD
PROJ-SUBJ
PROJ-RE
PRIOR-PROJ
ORG-EMP

DQG

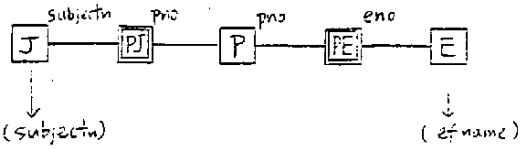


533 SUBJECT
S-P
PROJ-SUBJ
P-S
535 PROJECT
PRIOR-PROJ
536 PROJ-RE
PRIOR-PROJ
537 PROJECT
P-S
P-E
PROJ-SUBJ
540 PROJ-EMP-LNK
S-P
E-P
SUBJECT
539
541 EMPLOYEE
E-P
542 PROJ-EMP-LNK
P-E
543 PROJECT
PROJ-PROD
PRIOR-PROJ
PRODUCT
544
545 PROJ-RE
PRIOR-PROJ
(535) 546 PROJECT
P-E
547 PROJ-EMP-LNK
E-P
548 EMPLOYEE
ORG-EMP
549 ORGANIZATION
ORG-EMP
(541) 550 EMPLOYEE
E-P
551 PROJ-EMP-LNK
P-E
552 PROJECT
P-S
553 PROJ-SUBJ
S-P
554 SUBJECT
S-R
REP-SUBJ
P-S
REPORTR
556
R-E
R-S
EMP-REP-LNK
557
561
566
E-R
EMPLOYEE
(541)
EMP-REP-LNK
EMPLOYEE
REP-SUBJ
SUBJECT
562
S-R
S-P
563
PROJ-SUBJ
564
(535)
PROJECT
PROJ-PROD
565
PRODUCT
REP-SUBJ
SUBJECT
567
PROJ-SUBJ
568
PROJECT
569
(543)

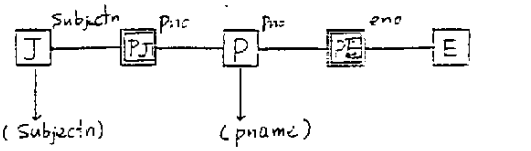
```
RANGE (P,PROJECT) (E, EMPLOYEE) (J, SUBJECT ) ;
RANGE (PJ, PROJ-SUBJ) (PE, PROJ-EMP-LNK) ;
RETRIEVE INTO TEST500 (J.SUBJECTN, E.EFNAME, P.PNAME)
WHERE P.PNO          = PE.PNO    AND PE.ENO          = E.ENO                    AND
      P.PNO          = PJ.PNO    AND PJ.SUBJECTN     = J.SUBJECTN ;
```



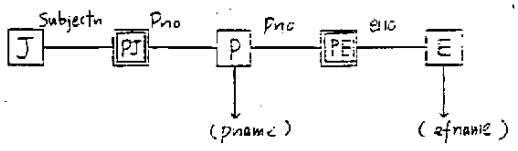

```
RANGE (P,PROJECT) (E, EMPLOYEE) (J, SUBJECT ) ;
RANGE (PJ, PROJ-SUBJ) (PE, PROJ-EMP-LNK) ;
RETRIEVE INTO TEST501 (J.SUBJECTN, E.EFNAME)
WHERE P.PNO          = PE.PNO    AND PE.ENO          = E.ENO                    AND
      P.PNO          = PJ.PNO    AND PJ.SUBJECTN     = J.SUBJECTN ;
```



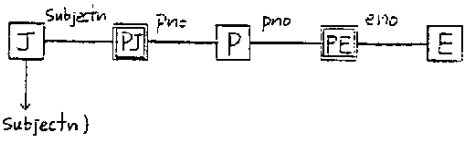

```
RANGE (P,PROJECT) (E, EMPLOYEE) (J, SUBJECT ) ;
RANGE (PJ, PROJ-SUBJ) (PE, PROJ-EMP-LNK) ;
RETRIEVE INTO TEST502 (P.PNAME, J.SUBJECTN)
WHERE P.PNO          = PE.PNO    AND PE.ENO          = E.ENO                    AND
      P.PNO          = PJ.PNO    AND PJ.SUBJECTN     = J.SUBJECTN ;
```




```
RANGE (P,PROJECT) (E, EMPLOYEE) (J, SUBJECT ) ;
RANGE (PJ, PROJ-SUBJ) (PE, PROJ-EMP-LNK) ;
RETRIEVE INTO TEST503 (P.PNAME, E.EFNAME)
WHERE P.PNO          = PE.PNO    AND PE.ENO          = E.ENO                    AND
      P.PNO          = PJ.PNO    AND PJ.SUBJECTN     = J.SUBJECTN ;
```




```
RANGE (P,PROJECT) (E, EMPLOYEE) (J, SUBJECT ) ;
RANGE (PJ, PROJ-SUBJ) (PE, PROJ-EMP-LNK) ;
RETRIEVE INTO TEST504 (J.SUBJECTN)
WHERE P.PNO          = PE.PNO    AND PE.ENO          = E.ENO                    AND
      P.PNO          = PJ.PNO    AND PJ.SUBJECTN     = J.SUBJECTN ;
```



昭和５６年３月発行

発行所　財団法人　日本情報処理開発協会
東京都港区芝公園３－５－８
機械振興会館内
TEL　(434)8211(代表)

印刷所　株式会社　タ　ケ　ミ　印　刷
東京都千代田区神田司町２－１６
TEL　(254)5840(代表)

55—S002