54-S 001

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)

昭和 55 年 3 月

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５４年度に実施した「分散型リソース処理技術の研究開発」の成果をとりまとめたものであります。

序

当協会は情報処理技術の研究開発の一環として、昭和５２年度より「分散型リソース処理技術の研究開発」に着手いたしました。

コンピュータ利用の高度化および複雑化にともない、ハードウェア、ソフトウェア、データベース等のリソースの集中処理はほぼ限界に達してまいりました。

分散処理システムの出現によって大量のデータおよび複雑な処理が可能になった反面、新たに発生した異機種に分散しているデータおよびプログラムをどのようにして効率よく、かつ容易に利用するかという問題の解決に迫られております。このような問題を解決するためには、分散処理システムのリソース全体を総合的にとらえる必要があります。

また、わが国の情報処理の特徴の一つである日本語情報処理について、低コストで操作性にすぐれた日本語端末の開発が必要であります。

当協会は、コンピュータネットワークＪＩＰＮＥＴを有し、過去において日本語情報処理の研究開発をおこなっており、このような問題解決のための研究開発を実施しうる環境をそなえていると判断いたし、昭和５２年度より３ケ年計画で、分散処理におけるリソースの統合に関する研究開発を実施致しました。

本年度は最終年度にあたり、データベース統合のモデルの作成、評価方式の検討、実験・評価、日本語端末モデルの作成をおこないました。

本報告書は分散型データベースシステム編と日本語情報処理編の２分冊から成り、分散型データベースシステムに関する研究開発の総合報告としてそ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ある方々に広く利用され、わが国情報処理技術向上の一助として寄与できることを念願いたす次第であります。

昭和５５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

1. 概　論

１９７０年代に確立されたＡＲＰＡＮＥＴ〔ＲＯＢＥＬ７０〕を始めとするコンピュータネットワーク技術によって地理的に離れたコンピュータ相互の基本的な通信が可能となった。これによって、地理的距離にかかわりなく、ハードウェア、ソフトウェア、データというコンピュータリソースの共有を可能ならしめてきた。この成果は、ＩＳＯ／ＴＣ９７／ＳＣ１６〔ＢＡＣＨＣ７８〕における様な、通信プロトコル階層（図1.1）の明確化による国際標準化としてまとめられてきている。こうしたコンピュータネットワーク技術の発展の背景には、通信回線コストの減少に比してＬＳＩ技術を中心とするハードウェア技術の進歩によるコンピュータハードウェアの大幅なコスト低下とこれによるパケット交換技術の進歩とをあげることができる。更に最近のＶＬＳＩを中心としたデバイス技術のめざましい進歩は、かっての大型機並みのメモリと処理能力を持つロジックとを１つのチップ内に埋め込むことを可能としてきている。１９８５年頃には、１つのチップ内に１０万ゲート（ＩＢＭ　３７０／１５８程度）の集積が可能とさえ言われている。このことは、かって高価であったプロセッサとメモリとを、時分割システム（ＴＳＳ）又はコンピュータネットワークを介して多くのユーザの間で共有してきた時代から、これらの処理能力を１人のユーザが占有出来る時代をむかえようとしていることを示唆している。この典型的な例として、現在の大型又は超大型機程度の処理能力を備え、電卓並みのコンパクトさを持ったパーソナルコンピュータがある。今まで時分割システム又はネットワークを介して行なっていた処理をこれによって占有的に行なえる。リソースを共有することによって必要となっていたスケジューリング等の複雑な制御機能は不要となり、ソフトウェア危機の典型的な例としての巨大オペレーティングシステムもこの多くが不要となろう。この様なコンピュータソースの占有化の動向のなかで、最後まで共用されるリソースとは何であろう。それはデータであると我々は考えている。

１９７０年代初頭に、ＩＢＭのＥ.Ｆ.Ｃoddは、データモデルとして著名なリレーショナルモデルの提案〔ＣＯＤＤＥ７０〕を行なった。このリレーショナルモデルは、従来のデータベースシステムに対して、高度のデータ独立性と非手続的問合せ言語とを提供できることを特徴としている。又、これらの新しい概念を裏づけるものとして、強力な数学的基礎を持っている。この発表と同時に、Bachman を始めとする従来のデータベース研究者との間に激しいデータモデル論争〔ＳＩＧＭＤ７４〕を起こした。Bachman 等の提唱するネットワークモデルは、レコード型間に１：ｎの関係性を表わすセット型のネットワークをつくることによってデータを表わし、このネットワークをたどることによって必要なデータを得さようとするものである。ネットワークモデルはリレーショナ

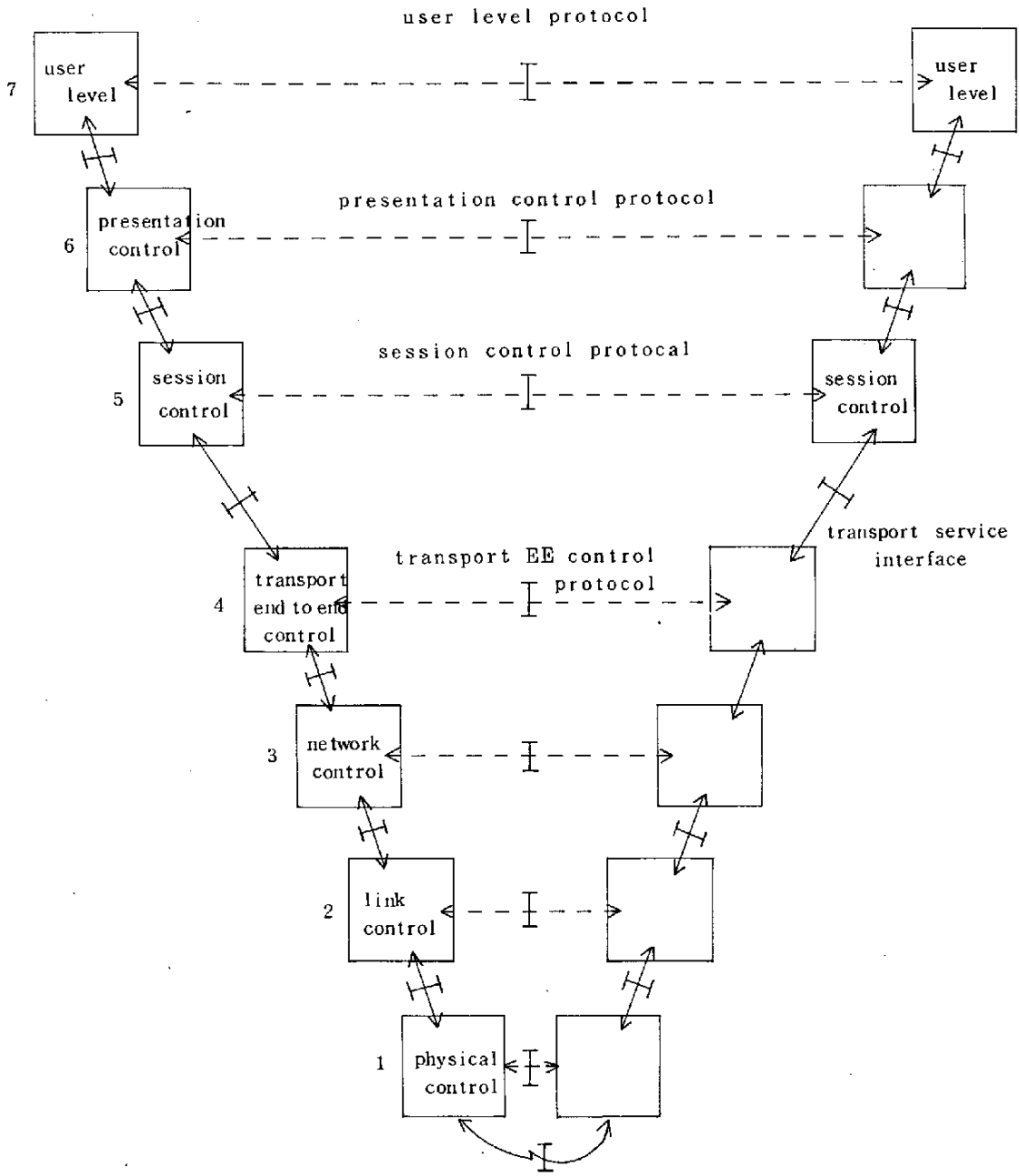


図1.1　ISO／TC 97／SC 16／N 34〔BACHC 78〕の protocol 階層

ルモデルと比較して、データ要素間の関係性が明確で固定的でありアクセス言語が手続的である。加えて、多くの物理構造が反映されている。しかし、彼のネットワークモデルは、その後、多くの商用システムで実現され伝統的なデータ処理、即ち定形業務処理に広く用いられるとともに、国際的な標準化作業が CODASYL DBTG 〔CODAS 74、78〕によって、ＤＢＴＧモデルとして進められている。

一方、リレーショナルモデルも、良好なパフォーマンスを得ないことについての批判も、Ｂ－tree〔ＢＡＹＥＲ ７２，ＨＡＥＲＴ ７８〕等のインプリメンテーション技術の進歩によって克服され、他の商用データベースシステムの手続的処理と同程度のパフォーマンスを得れる〔ＡＳＴＲＭ ７９〕までになっている。この例としてＳＴＳＴＥＭ Ｒ〔ＡＳＴＲＭ ７６〕、ＩＮＧＲＥＳ〔ＳＴＯＮＭ ７６〕，ＱＢＥ〔ＺＬＯＯＭ ７５〕、富士通のＲＤＢ等の商用システムがある。リレーショナルデータベースシステムは、ＤＢＴＧモデルと比して、新たな非定形的な業務に多く用いられると共に、自然言語のような容易なユーザインタフェースへの完全な（complete）核システムとして用い〔ＷＡＬＴＡ ７０，ＳＡＧＡＤ ７７，ＣＯＤＤＥ ７８，ＭＹＬＯＪ ７６〕られようとしている。又、リレーショナルモデルの簡潔性はＶＬＳＩを中心とするデバイス技術進歩とともにこれを直接に実行させようとするデータベースマシン〔ＤＥＷＩＤ ７９，ＢＡＮＥＴ ７９，ＯＺＫＡＥ ７７，ＳＵｎＳ ７９〕の実現を可能とさせてきている。

今後、伝統的な定形業務中心の既存大型データベースシステムでは、ＤＢＴＧモデルが用いられ、新たな非定形業務、管理業務、及びより一般のユーザー向けのデータベースシステムではリレーショナルモデルが用いられていこう。

これら２つのモデルの共存そして競合の時代は今後４～５年続くであろう。しかしリレーショナルモデルによって確立されたデータ独立性の概念、即ち物理的なアクセス方法及び格納方法と、論理的なデータ記述及びアクセス手段との間の明確な分離は、ＤＢＴＧモデルへも大きな影響を与えてきている。１つの例は〔ＣＯＤＡＳ ７８〕によるスキーマＤＤＬとＤＳＤＬとの分離である。この考え方を、更に進めたものは、ＡＮＳＩ／Ｘ３／ ＳＰＡＲＣ〔ＡＮＳＩＸ ７５，ＴＳＩＣＤ ７８〕のレポートに発表されたデータベースシステムの３つの階層化〔図1.2〕である。この階層化の重要な点は、各階層が各々のレベルに対応したあるデータモデルとデータ言語とを備えるオートマトンと考えられるということである。このモデルと言語とは、ユーザ側の層（外部スキーマ）では彼のアプリケーションに適したものであり、最下の物理構造を表わす層（内部スキーマ）では物理的アクセスと格納とに有効なものである。更に、この２つの層のマッピングの核となる層（概念スキーマ）は、格納されたデータのセマンティクスの一意な記述に関している。

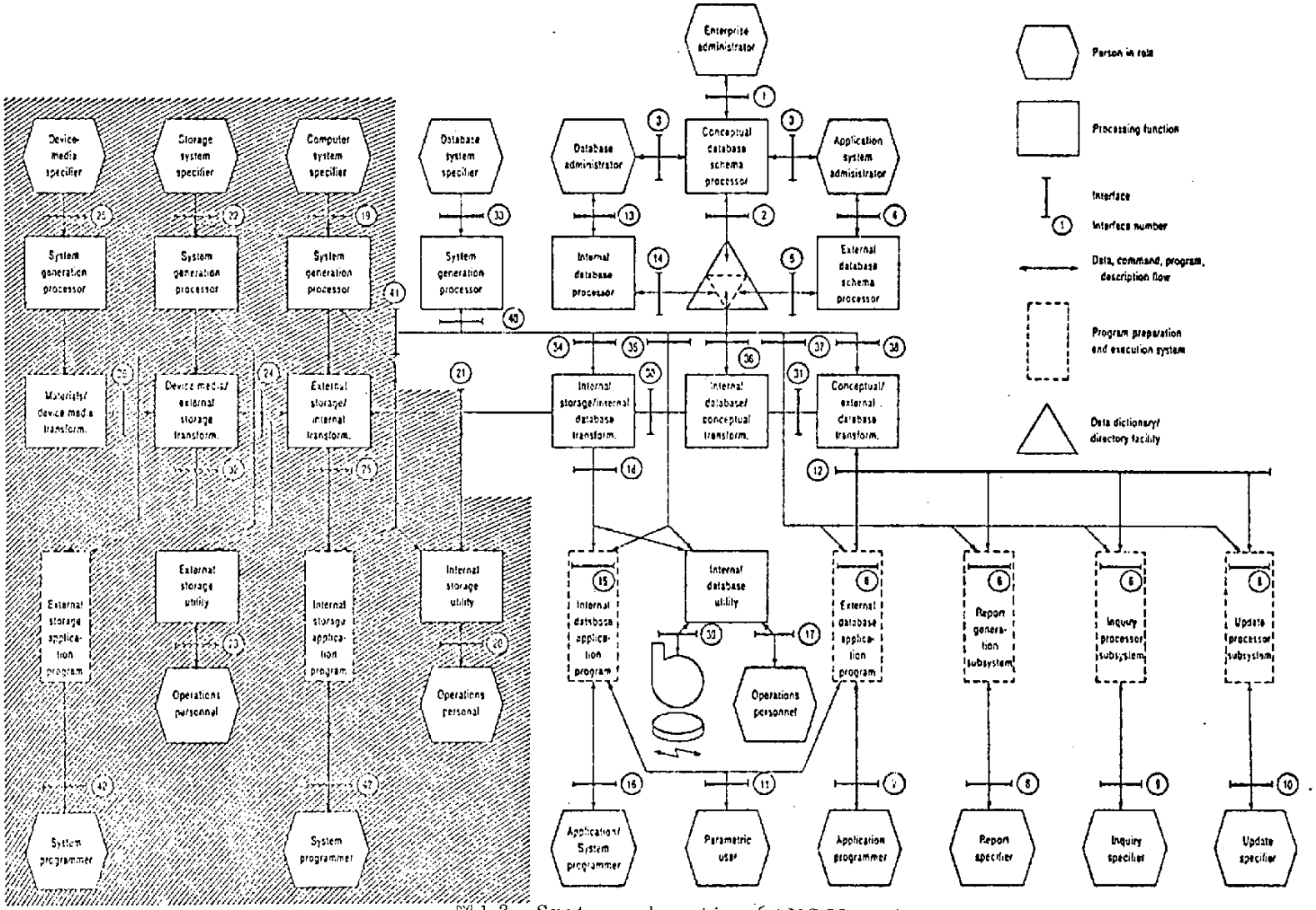


図 1.2　System schematic〔ANSIX 75〕

上述してきたデータベースシステム技術の進歩は、データを物理的構成とは独立に、各々の目的に適した形式で、かつデータの論理的な意味に基づいてアクセスすることを可能としてきた。

本章でみてきたデータベースシステム技術の進歩とネットワーク技術の進歩とは、分散型データベースシステム（distributed database system）として近年結集してきている。分散型データベースシステム研究は、１９７０年代後半より米国、ヨーロッパを中心に本格的に開始された。例として、ＳＤＤ－１〔ＲＯＴＨＪ ７７〕，ＩＮＧＲＥＳ〔ＳＴＯＮＭ ７７〕，ＬＡＤＤＥＲ〔ＳＡＧＡＤ ７７〕，ＸＮＤＭ〔ＫＩＭＢＳ ７９〕，ＩＤＥＲ〔ＬＩＥＮＹ ７８〕，ＡＤＤ〔ＭＡＨＭＳ ７８〕，ＰＯＬＹＰＨＥＭＥ〔ＡＤＩＢＭ ７８〕．ＳＴＡＲ〔ＧＡＲＤＧ ７９〕，ＰＯＲＥＬ〔ＮＥＵＨＥ ７７〕，ＶＤＮ〔ＭＵＮＺＲ ７９〕等がある。１９８０年代における分散型データベースシステム技術の研究開発が重要である理由は、本章の始めで述べた様に、メモリとプロセッサの占有化のなかで、データが共用される最後で最重要なリソースであるからである。現在のデバイス技術は、メモリを高速にかつ安価にしてきているが、社会の必要とするデータ量の増大と多様化は、これをはるかにしのいでいる。このことは、１つのサイトにデータを集中して管理しておくことが物理的に困難であることを示している。よってデータを、これらを最も必要とするサイト又はデータの発生源に分散して保持しておこうということが、分散型データベースシステムへの最初の発想であった。更に、これには、信頼性とセキュリティ等の問題から、集中して管理するものよりも分散して管理することの方が望ましいということもある。

更に考えを進めたものとして、論理的に異なった意味を持つデータベースから新たなデータベースシステムをつくろうという統合化の概念が、分散型データベースシステムの新しい柱と成ってきている。統合が可能となってきた背景には、データベースシステム技術の成果で述べた様に、各データベースシステムが、物理構成とは独立なよりセマンティカルなデータ記述系としてのデータモデルと、これに基づいたより高度なアクセス言語とを提供してきていることがある。今後、データベースシステムは、より以上にデータのセマンティクスに着目していくだろう。データアクセス言語として関数型言語〔ＢＡＣＫＪ ７８〕、論理プログラミング言語〔ＫＯＷＡＲ ７４〕等のプログラミング言語と統合した統一言語（unified language）化〔ＥＬＭＡＲ ７９，ＦＵＴＯＩ ７８〕が進むとともに、これらを核にした自然言語インタフェースの開発も盛んになっていくと思われる。我々の目指す分散型データベースシステムも最終的には、この様な異なった意味構造を持つデータベースシステムから、新たな論理的データベースシステムをつくろうとするものである。

本報告書は、当協会で昭和５２年度より開始された「分散型リソース処理技術の研究開発」の一環として行なわれた分散型データベースシステムＪＩＰＮＥＴ－ＤＤＢＳ（ＪＤＤＢＳと呼ぶ）の

3年間の研究開発の総合報告書である。本報告書の中で、既存データベースシステムとして既存の3つのデータベースモデル、i.e.、リレーショナルモデル、DBTGモデル、IMSモデルを設定し、これらの分散型データベースシステムへの統合とアクセス問題について論じた。特に、今後、既存大型データベースシステムとして重要となっていくと考えられるCODASYL DBTGモデルの統合とアクセスについて論じた。第2章では、分散型データベースシステムとして何が問題になるかを明らかにした。第3章では、分散型データベースシステムの全体アーキテクチャ、即ち、システムのスキーマ層として四層スキーマ構造（FSS）概念〔TAKIM 78, 79a, HAMAE 78〕を提案した。第4章と5章とでは、既存データベースシステムの異種性としてデータモデルと言語との克服について論じた。特に第4章では、異種性除去過程（同種化と呼ぶ）〔TAKIM 79a〕について論じ、第5章では共通アクセス言語からDBTGDMLへの変換問題（問合せ変換と呼ぶ）〔TAKIM 79b, 80a〕について論じた。第6章と7章とでは、各データベースシステムが地理的に分散しかつ論理的に異なった意味を持つことによって生じる分散問題について論じた。特に6章では、統合化〔TAKIM 80b〕を7章では各サイトでの問合せの分散処理（問合せ分割と呼ぶ）〔TAKIM 80b, c〕について論じた。第8章では我々のインプリメントした異種性情報処理システム（HIPS）問合せ変換システム（QTP）、分散情報処理システム（DIPS）、問合せ分割システム（QDP）の各々について論じた。

2. 分散型データベース問題

前章〔第１章〕で述べた様に、分散型データベースシステム（distributed database system or DDBS）は、次の２つの技術の結合のもとに可能となってきた。

1） ARPANET, CYCLADES, JIPNET に代表されるコンピュータネットワーク技術

2） ANSI/X3/SPARCのレポート〔ANSIX 75〕、リレーショナルモデル〔CODE 70〕等のデータベース管理システム（DBMS）技術

1）のネットワーク技術は、距離の克服、即ち地理的に離れたコンピュータ間の通信を可能とさせた。その成果は、〔BACHC 78〕に示される様な通信プロトコルの階層構成としてまとめられてきている。

DBMS技術の進歩は、まず第１にリレーショナルモデルを始めとするデータモデルと、これに基づいたデータアクセス／記述言語の成果に依っている。これによってデータベースシステム（DBS）を、あるデータモデルとあるデータ言語とを提供するブラックボックスとすることが可能となってきた。このことは又、ANSI／X3／SPARCのレポート〔TSICD 78, ANSIX 75〕で述べられている様なデータベースシステムの三階層化へ導いていった。これは、データベースシステムを、コンピュータのファイル、アクセス方法等の物理構造に依存した部分（内部スキーマ）、ユーザのアプリケーションへの適応を考えた部分（外部スキーマ）へ明確に分離するとともに、これらの２つの部分の変換の核の部分（概念スキーマ）を設けようとするものである。これらの３つの階層の各々は、レベルの差はあるが、あるデータモデルと言語とによって特徴づけられている。以上のことより、DBMS技術の進歩は、データベースシステムの階層を明らかにし、あるデータモデルと、これに基づいたデータ言語によってある階層を通してアクセスできることを可能としてきていると言える。更に、これらのデータモデルと言語とは、リレーショナルモデルに代表される様な高水準なものとなりつつあることも言える。

上述したコンピュータネットワーク技術とDBMS技術との結合点としての分散型データベースシステムは、地理的に分散したデータベースシステムを、ネットワークを介して距離を意識することなく、各データベースシステムのある階層を通して通信させ、共同して作業を行なわせようとするものである。よって、我々は分散型データベース問題を論じるうえで、次の様な仮定を設けることができる。

1） 各データベースシステムはコンピュータネットワークによって互いに結合されている。

コンピュータネットワークは、コンピュータ間の基本的通信機能、即ち、あるサイトで発

せられたメッセージを、目的のサイトへ届ける機能（ホスト・ホスト・プロトコル）を提供している。

2） 各データベースシステムは、分散型データベースシステムからみた時、あるデータモデルとあるデータ言語とを提供するオートマトン（ブラックボックス）と考えることができる。

以降、この仮定の基に、分散型データベースシステム（ＤＤＢＳ）問題として何を解決し、何を論じねばならないかを明らかにする。

分散型データベースシステムを考えるうえで、次の点を論じる必要がある。

1） 設計アプローチ

2） 全体アーキテクチャ（gross architecture）

3） スキーマ層の設計方法

4） アクセス機能

5） 制御機能

6） アプリケーション

7） システムアーキテクチャ

A．設計アプローチ

まず、分散型データベースシステムを考えるうえで、次の２つのアプローチのうちどちらをとるかが重要な問題となる。

1） 分割型（top-down）アプローチ

2） 統合型（bottom-up）アプローチ

1）では、まず分散型データベースシステム上に仮想的なデータベースを考える。次に、データベースの外延（extension）を、ある目的関数のもとで最適となる様に外データベースシステムサイトへ分配する〔図2.1〕。このアプローチにおいて、ユーザの利用要求に会うように、システムの信頼性、データの獲得性（availability）を高めるために、各サイトのデータの冗長性（redundancy）を高めることもなされる。このアプローチは、新しくデータベースをつくるために、top-down的に目的に合う様にシステムをつくっていこうとするものである。よって多くの場合、各サイトのデータベースシステムは、仮想データベースシステムと同種のものであり、いかに最適にデータを各サイトへ分配するかという最適配置問題が主要問題となる。

このアプローチの例として、ＣＣＡ社のＳＤＤ－１〔ＲＯＴＨＪ　７７〕、UC. Berkeleyの分散型ＩＮＧＲＥＳ〔ＳＴＯＮＭ　７７〕等が有名である。

これに対して、第２の統合型（bottom-up）アプローチは、既にデータベースが各サイトに

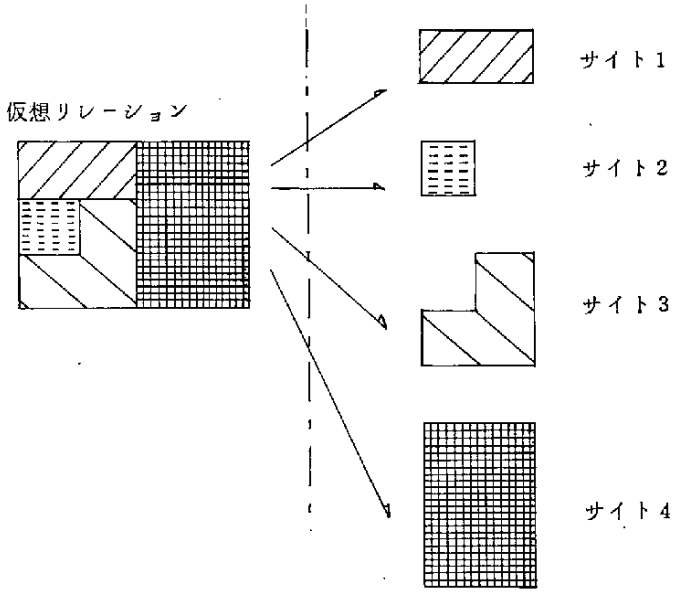


図2.1　分割型設計アプローチ

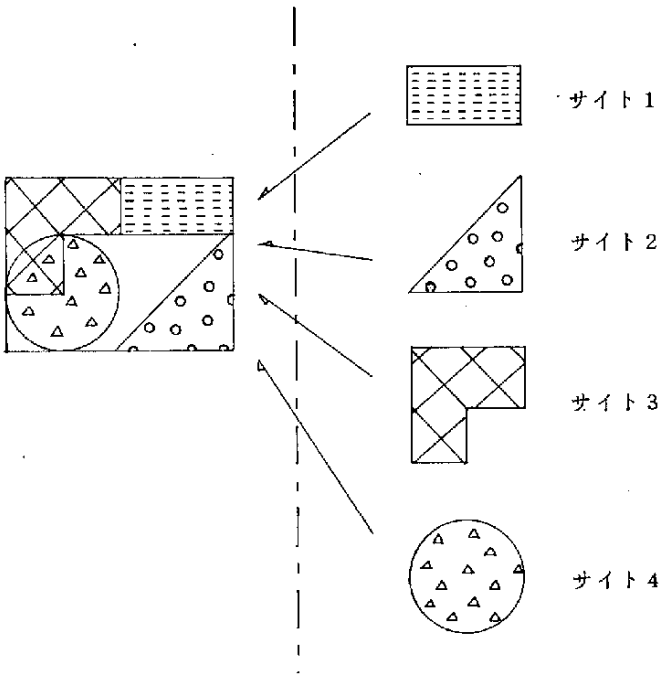


図2.2　統合型設計アプローチ

独立に存在している時に、これらを統合して1つのシステムをつくるために用いられる〔図2.2〕。各サイトのデータベースシステムは、他のデータベースシステムとは独立に、それ自身の固有のアプリケーションに適するようなデータ構造を持っている。従って統合型アプローチにおける問題は、既に存在している異なったデータモデルと言語とを持ち、異なった意味構造を備えたデータベースをいかに仮想的な1つの論理的データベースへ統合するかである。このアプローチは、一般に先述した分割型に比して困難であり、全ての場合について統合が可能とは言えない。又、このアプローチでは、一般に、各データベースシステムは、異なったデータモデルとデータ言語とを備えた異種（heterogeneous）データベースシステムである。この例として、グ

ルノーブル大学のＰＯＬＹＰＨＥＭＥ〔ＡＤＩＢＭ　７８〕がある。

分散型データベースシステムへの２つの設計アプローチの選択は、後述する全体アーキテクチャ、設計方法、アクセス機能、制御方法、等に大きな影響を与える。

B．全体アーキテクチャ

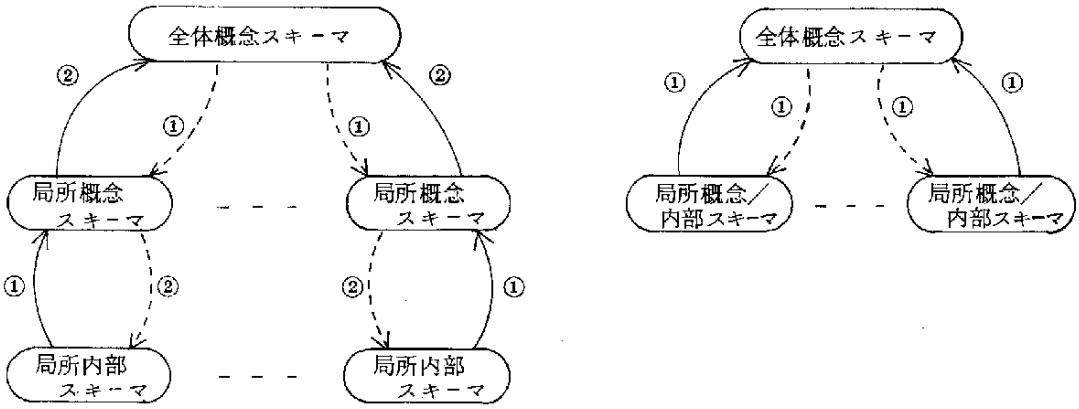
全体アーキテクチャ（gross　architecture）とは、分散型データベースシステム全体のスキーマ階層をどのように設定するかに関している。このことは又、各スキーマ階層にどの様なデータモデルとデータ言語とを設けるかを決定することでもある。このスキーマ階層の最下の層は、各データベースシステムの通信レベルに対応している。これより上位の階層には、分散型データベースシステムとして仮想化された１つの巨大システムレベルに対応するスキーマ階層がある。このスキーマ階層は、分散型データベースシステムの核となるものである。分割型設計アプローチでは、まずこのスキーマ階層に対応するスキーマの定義がなされ、ある目的関数に基づいて各サイト、即ち最下の層のスキーマが決定される。一方統合型設計アプローチでは、この逆に、まず最下の層の各サイトのスキーマが存在しており、これから分散型データベースシステムの核となる層のスキーマが決定されることになる。全体アーキテクチャに対する我々のアプローチは、統合型設計を目指す四層スキーマ構造（ＦＳＳ）と呼ばれるもので、４つのスキーマ階層から成っている。ＦＳＳの詳細については、第３章において論じる。統合型設計アプローチにおいて、主要な問題点として、

１）　構成要素たるデータベースシステムの異種性を解決することと

２）　地理的に分散し異なった意味構造を持つ各データベースシステムを１つの論理的データベースシステムへ統合すること

の２点を指摘できる。我々のＦＳＳアプローチは、まず第１の問題を解決するための層（これを局所概念スキーマと呼ぶ）と、これの１つ上位に第２の問題を解決するための層（これを全体概念スキーマと呼ぶ）とを設けたことが大きな特徴となっている。〔図２３ａ）〕同種データベースシステムの統合においては前者の層は、最下の層となる〔図2.3ｂ）〕。ここで、データベースシステムのスキーマを局所内部スキーマと呼んでいる。

我々のＦＳＳアプローチでは最上位のスキーマ層として、ユーザのアプリケーションに適したデータの記述である外部スキーマが存在している。第３章で述べるように外部スキーマ問題は分散型データベースシステム固有のものでないので、ここでは触れないことにする。



a） 異種データベースシステムから成る
分散型データベースシステム

b） 同種データベースシステムから成る
分散型データベースシステム

——→ ： 統合型設計

- - - - - -→ ： 分割型設計

図 2.3 分散型データベースシステム階層と設計

C．スキーマ層設計方法

図2.3において、実線は統合型（bottom-up）設計アプローチにおける分散型データベースシステムの各スキーマ層がどのように生成されていくかを示している。一方点線は、分割型(top-down)設計アプローチにおける場合を示している。丸で囲まれた数字は、スキーマ層の生成順序を示している。この様な各スキーマ層の生成方法を設計方法と呼ぶ。図2.3から解かる様に、分割型アプローチにおいては、分散型データベースシステム全体のスキーマ、即ち全体概念スキーマ、が決定され、ある目的関数のもとで最適となる様に各サイトのスキーマとその外延が決定される。よって分割型設計方法においては、目的関数の設計が主要テーマとなる。データの

各サイトへの分配は、次の目標を達成するように決定される。

1） ＧＣＳレベルの問合せの応答時間と全処理時間（即ち通信コスト）の最少化。即ちデータの獲得性（availability）を高めること。

2） 各サイトでの格納コストの最少化。

3） 分散型データベースシステム全体の信頼性の最大化。

1）を達成するためには、多くの問合せが必要とするデータがなるべく問合せを発したサイト又はその近傍にあること，即ち、多くの問合せをなるべくサイト間の処理を必要とすることなく局所的に処理できるようにすることが求められる。このためにはデータの冗長性（redundancy）を高めることが必要となる。又、この冗長性は、あるサイトが障害を起こしても、他のサイトに冗長コピーが存在するために、分散型データベースシステム全体の信頼性を高めることになる。しかし、一方では、冗長コピーの存在は、システム全体の格納コストを増大させるとともに、信頼性問題、即ち深刻なインテグリティと一致性（consistency）制御問題〔ＢＥＲＮＰ７８〕をもたらすことになる。特に、動的なデータ、即ち更新頻度の高いデータが分散型データベースシステム内に冗長に存在すれば、各冗長コピー間のインテグリティと一致性を保つために、膨大な通信オーバヘッドが生じてしまう。以上の点を考慮して、分散型データベースシステムへの利用要求を満たすようにデータの配置が決定される。各データベースシステムが異種のものであるならば、局所概念スキーマを局所内部スキーマへマッピングすることが必要となる。

これに対して、統合型（bottom-up）設計法では、何よりも、各データベースシステムは他のデータベースシステムとは独立に自立して既に存在してしまっていることが大きな特徴となっている。一般に各データベースシステムは異種である。よって、まず異なったデータモデルと言語とをもった各データベースシステムの局所内部スキーマ層から、分散型データベースシステム全体で共通なデータモデルと言語とを備えた局所概念スキーマ層を生成せねばならない。この過程を、我々は同種化（homogenization）〔第４章を参照のこと〕と呼ぶ。同種化では各データベースシステムに固有のデータモデルから、分散型データベースシステム全体で共通のデータモデルへの変換を行なうことになる。ついで、局所概念スキーマ層から、全体概念スキーマの生成が行なわれる。この過程は困難なものである。何故なら、一般に、互いに異なった意味（論理）構造を持つデータベースの集合から、新たな論理構造を持ったデータベースをつくらねばならないからである。さらに、この過程では、分散型データベースシステムを利用しようとする種々のアプリケーションからの要求も考慮されねばならない。我々は、この過程を統合化（integration）と呼ぶ。分割型設計においては、応答時間、通信コスト、信頼性、格納コスト等を最適化することが主たる問題であった。しかし、統合型（bottom-up）設計においては、異なった

意味を独立に持つデータを、いかに新たなデータの意味へ統合し得るが主要問題となる。統合（integration）は全ての場合において可能とは限らない。現状では、まず統合し得る可能性を示すことがまず必要と考えられる。

D．アクセス機能

これまで述べてきた様な分散型データベースシステムのスキーマ階層が設計されたならば、次に、これらのスキーマ層に対して発せられた問合せをいかに処理するかというアクセス問題を解決せねばならない。アクセス問題は、スキーマ階層に対応して大きく2つに分けて考えられる。第1に、全体概念スキーマ層に対して発せられた問合せを、それが必要とするデータを保持しているデータベースシステムサイトへ分割する必要がある。我々はこれを問合せ分割（query decompostion）と呼ぶ。問合せ分割では、全体概念スキーマ問合せを、それが参照するデータを保持しているサイトの局所概念スキーマ問合せへ分割するとともに、サイト間にまたがった問合せの処理を行なわねばならない。前者が全体概念スキーマ要素から局所概念スキーマ要素への表現の変換であるのに対して、後者ではサイト間でのデータの転送を行なう必要がある。分散型データベースシステムにおいてデータの転送のための通信コストがそのコスト／パフォーマンス決定に支配的役割を持つことから、データ転送方法の最適化は主たる問題になる。この転送方法の決定アルゴリズムも、分散型データベースシステムの設計アプローチと密接に結びついている。即ち、分割型アプローチにおいては、データは、問合せの要求に合うように配置されており、多くの問合せ処理をあるサイト又はその近傍に局所化できる。しかし、統合型では一般に、その様に全体概念スキーマを設計していないので、より多くのサイト間処理が必

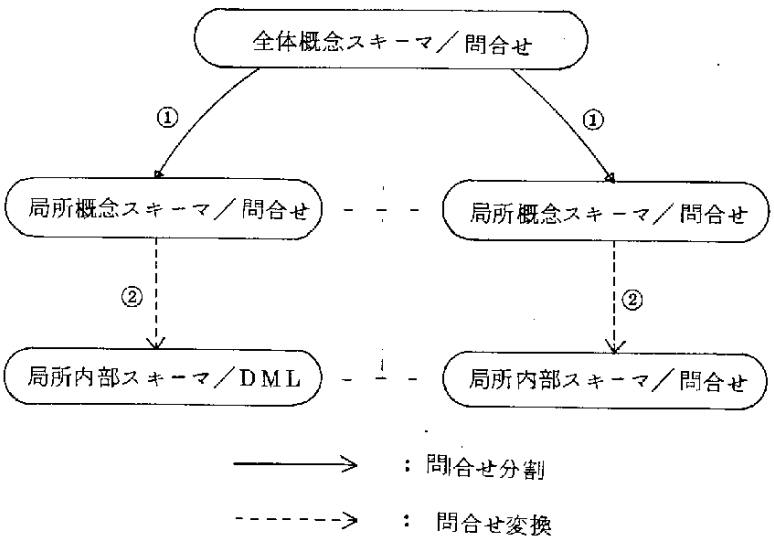


図2.4　分散型データベースシステムのアクセス問題

要となるかもしれない。

第２のアクセス問題は、問合せ分割によって生成された各データベースシステム単位の局所概念スキーマ問合せを、そのデータベースシステムで実行可能な局所内部スキーマＤＭＬへ変換する問題である。我々は、これを問合せ変換（query translation）と呼ぶ。局所概念スキーマ問合せは、分散型データベースシステム全体で共通なデータモデルに基づいているとともに、非手続的なものである。このため問合せ変換では、問合せの参照する局所概念スキーマ要素を、対応する局所内部スキーマ要素へ変換するとともに、そのデータベースシステムが最適に実行できる様なアクセス手続を生成せねばならない。最適化は、局所内部スキーマ層レベルにおけるものであって、実際のデータベースシステムの物理レベル、即ちＡＮＳＩ／Ｘ３／ＳＰＡＲＣで言うところの内部スキーマレベル、まで考慮したものではない。局所内部スキーマＤＭＬから、データベースシステムの内部スキーマ層のアクセス言語への変換と最適化は、そのデータベースシステムの責任であると考える。問合せ変換機能は、各データベースシステムの局所概念スキーマ層も設けるためのインタフェイスとも考えられる。

E．制御機能

上述してきたアクセス問題では、問合せの検索機能についてのみ考えており、追加、消去、置換といった更新機能を考えていない。更新問題は、特に問合せ分割において問題となる。全体概念スキーマレベルでの更新要求をいかに矛盾なく局所概念スキーマ層へ伝えるかは重要な問題である。これは、リレーショナルモデルにおける視野（view）更新〔ＤＡＴＡＵ　７８，ＣＨＡＭＤ　７６〕に類似している。更に問合せ分割では、冗長コピーに対する更新要求も考えねばならない。先述した様に、あるデータ要素が複数の冗長コピーを分散型データベースシステム内に持つ時、これに対する更新は、全コピーに対してなされねばならない。これらのコピーに対する更新の同期も管理されねばならない。この冗長コピーに対する更新問題は、一般に冗長コピーへの同時実行（concurrency）及び一致性（consistency）制御問題（我々はＣＣ問題と呼ぶ）と呼ばれるものである。現在、コピーへのロックをかける手法がとられ、いかにロック制御のための通信コストを少なくし、パフォーマンスを改良するかに多くの研究〔ＢＥＲＮＰ　７８〕がなされている。したがって、デッドロックの検出、復旧、回避問題も、このなかにはいっている。異種データベースシステムから成る分散型データベースシステムにおいて、ＣＣ制御は、各データベースシステムの局所概念スキーマ層間でなされねばならない。よって、これは局所概念スキーマ層で用いられるデータモデルが何であり、このモデルにおけるロック（lock）の単位は何かに依存している。我々は異種分散型データベースシステムにおいて、ロックは局所概念スキーマレベルで物理的ではなく論理的に処理されるべきと考えている。

他の重要な問題は、障害をいかに復旧させるかである。問合せの検索においても、サイト間にまたがった問合せの処理中に、必要などれかのデータベースシステムサイトが障害を起こしたならば、これ以上の処理は出来なくなる。この時サイト間処理のために生成された中間結果をどのように扱うかも問題である。この場合、更新問題は、更に複雑となる。又、冗長コピーが存在する場合には、より問題を複雑なものとしてしまう。

更に、セキュリティ（security）問題も今後重要となってくるだろう。

我々は上述した同時実行及び一致性制御（ＣＣ制御）、障害管理、セキュリティ管理を合わせて分散型データベースシステムの制御問題と呼ぶ。

F．システムアーキテクチャ

分散型データベースシステムの全体アーキテクチャとしてのスキーマ階層、これら各階層の設計、アクセス、及び制御問題の検討は、分散型データベースシステムをどのようにつくるかというシステムアーキテクチャの検討を必要とさせる。システムアーキテクチャ問題とは、これまで述べられてきた分散型データベースシステムの階層と機能とをどのように実現するかを明らかにすることである。各機能モヂュールを明確にし、各モヂュール間のインタフェースを明らかにし、各モヂュールの実現形態を明らかにすることである。例えば、問合せの変換機能は、データベースシステムをサポートする大型ホストコンピュータ内にあるべきか、又は、ホストとネットワークとの間のフロントエンドとしてのミニコンピュータ内にあるべきかということである。

G．アプリケーション

最後に重要な問題として、この様につくられてきた分散型データベースシステムをいったい何に使うのかという問題、即ち、アプリケーションの明確化の問題がある。特に、統合型データベースシステムでは、それを構成するデータベースシステムのライフサイクルも考えなければならない。近い将来にデータベースシステムの再構成が行なわれるのならば、統合型として設計する必要はないであろう。又、分散型データベースシステムは、ＶＬＳＩ等の進歩のなかで集中型データベースシステムに対して本当にメリットを持ち得るのかという問題もある。ＶＬＳＩの進歩は、ＴＳＳに代表されるようなプロセッサの共有から、各ユーザがプロセッサを占有する方向をもたらしている。その典型的ものとして、従来の大型コンピュータ並みのプロセッサとメモリとを持ったパーソナルコンピュータがある。この様なコンピュータリソースの占有化の流れのなかで、最後まで共有され得るものは何か。我々は、それはデータベースであると考える。共有される理由は、データの量がその格納媒体と比して大きいことと、データのセキュリティ等の管理上の問題によっている。こうしたデータベースを、種々の視野のもとで共有する問題は、分散型デ

ータベースシステム、とりわけ統合型システムの問題である。

又、オフィスオートメーションとＣＡＤ／ＣＡＭに代表される新しいコンピュータ利用形態は、これまでの文字数字中心でレコードアクセス中心の従来のスタティックな大型データベースに変わって、種々の形態のデータが動的に各サイトを流れていくことをもたらすだろう。これは、従来の蓄積が主たる役目であったデータベースシステムに加えて、流れる動的（流れながら形態、内容を変えていく）なデータの中継点としての小さなデータベースが大きな位置を占めてこよう。この小さなデータベースは例えば個人用、研究者用のデータベースである。流れるデータは、必然的にデータベースを通信ネットワークによって結合し、統合的にアクセスすることを要求する。これはまさに分散型データベースである。我々はこのような新形態のデータを扱う分散型データベースシステムに対する基本機能としての問合せ処理問題、スキーマ階層の設計問題等に取り組んでいこうと考えている。

3. 分散型データベースシステムの全体アーキテクチャー四層スキーマ構造（ＦＳＳ）

分散型データベースシステム（ＤＤＢＳ）において、次の２つの問題を解決せねばならない。

1） その構成要素たるデータベースシステムの異種性を解決する問題（異種性問題）

2） 地理的に分散した論理的に異なった意味を持つデータベースシステムを１つの論理的（仮想的）データベースシステムへ統合する問題（分散問題）

ここで言うデータベースシステム（ＤＢＳ）とは、分散型データベースシステムにおける各データベースシステム間の通信レベルに対応したスキーマ層を意味している。従って、データベースシステム全体を意味していない。

データベースシステム（ＤＢＳ）の異種性は、次の２つの面を持っている。１つはＤＢＳの構文的側面（syntactic aspect）である。これはデータベースシステムのデータモデル、アクセス言語が何であるかに関している。もう１つは、ＤＢＳがどのような意味を持ったデータを格納しているかというデータベースシステムの意味論的側面（semantic aspect）である。２つのデータベースシステムがあった場合、これらの相違とは、その構文的及び意味論的側面の各々についての相違である。これらの相違は、図3.1に示す様に組合せ的に4通りが考えられる。図3.1で1の場合、２つのデータベースシステムの統合は容易であり、簡単な所在情報を示すディレクトリがあればよい。

	構文的側面	意味論的側面
1	同	同
2	異	同
3	同	異
4	異	異

図3.1 ２つのＤＢＳの異種性

２の場合、１に加えて、データモデル言語の変換機能が必要になる。即ち異種性問題を克服する必要がある。３の場合は、少し問題は困難である。何故なら格納されたデータベースの意味が互いに異なっているからである。この意味の相違を克服することが分散問題である。４の場合は、３に加えて、データモデルと言語の変換が必要になる。この様に、データベースシステムの意味論的側面は、その表現、アクセス手段としての構文的側面とは独立であるので、分散型データベースシステムでは異種性問題と分散問題とを独立して考えることができる。又、分散型データベースシステムにおいて、既存データベースシステム（ＤＢＳ）を統合しようとする時、データベースシステムの意味論的側面の相違をいかに克服するかが最も主要なポイントになる。このために、まず各データベースシステムの意味論的側面の表現とアクセス系であるデ

ータモデルと言語とを各々ある共通モデルと共通言語とへ同種化し、しかる後に各データベースシステムの意味論的側面の統合を考えることが自然である。

我々の分散型データベースシステム設計アプローチ（我々は四層スキーマ構造（ＦＳＳ）アプローチと呼ぶ）は、上記の考察のもとに、まず第１に各データベースシステムのスキーマを共通モデルによって共通記述し、かつ共通言語を設定し、（異種性問題の克服）ついであるネットワーク共同体にとって意味のある仮想データベースシステムへ統合するもの（分散問題の克服）である。こうすることによって、各データベースシステムの物理構造（ｅ．ｇ．ホストコンピュータ、２次記憶）や、その構文構造（データモデル、言語）の変化に対する長寿命性（survivability）を獲得できる。更に、システムの２つへの階層化は、システムへの理解性を高めるとともに、処理上の機能の明確化、種々の変化への適応性を高めることができる。

分散型データベースシステム（ＤＤＢＳ）を考えるうえで、まず第１番目に明らかにされねばならないものは、全体アーキテクチャ（gross architecture）である。全体アーキテクチャとは、第２章で論じた様に、分散型データベースシステムのスキーマ階層をどのように設けるかを示している。このスキーマ階層は、分散型データベースシステムの設計、アクセス、制御、システムアーキテクチャを考える上で最も基本となるものである。我々の全体アーキテクチャ、四層スキーマ構造（four-schema structure）〔ＴＡＫＩＭ ７８，７９ａ〕は、上述してきたデータベースシステムの異種性に対する考察に基づき４つのスキーマ階層から成っている。即ち、４つのスキーマ層内に各データベースシステムの構文的相違を解決するためのスキーマ層（これは局所概念スキーマと呼ばれる）を設け、この上位にこれらの意味論的側面を解決するためのスキーマ層（これは全体概念スキーマと呼ばれる）を設けている。本章では、我々の分散型データベースシステムの全体アーキテクチャとしての四層スキーマ構造について論じる。

3.1では、四層スキーマ構造の概要を論じる。3.2では、これに基づいた分散型データベースシステムの処理上必要となる情報としてのネットワークデータディレクトリ（network data directory）について論じる。3.3では、同じく四層スキーマ構造に基づいたシステムアーキテクチャについて論じる。

3.1 四層スキーマ構造（ＦＳＳ）〔TAKIM 78, 79b〕

我々の分散型データベースシステム（ＤＤＢＳ）の全体アーキテクチャは、四層スキーマ構造（four－schema structure or FSS）と呼ばれる。これは図3.2に示すように、４層のスキーマ階層と各スキーマ層間のマッピングとそのマッピングに必要な情報とから成っている。

４層のスキーマは、局所内部（local internal）スキーマ、局所概念スキーマ（local

conceptual)スキーマ、全体概念（global conceptual)スキーマ、外部（external）スキーマとの４つである。局所内部スキーマ（ＬＩＳ）は既存データベースシステムのスキーマ又はサブスキーマに対応している。又、これはＡＮＳＩ／Ｘ３／ＳＰＡＲＣ〔ＴＳＩＣＤ　７８〕で言うところの概念スキーマ又は外部スキーマに対応している。データベースシステムのＬＩＳは、このデータベースシステムのデータのうち分散型データベースシステムとして提供し得るものの記述である。局所内部スキーマはあるデータモデル（リレーショナルモデル、ＤＢＴＧモデル又はＩＭＳモデル）で記述され、対応したアクセス言語を備えていると仮定できる。ＬＩＳの異種性とは、このデータモデルと言語との相違である。

局所概念スキーマ（ＬＣＳ）は、各データベースシステムの局所内部スキーマ（ＬＩＳ）を分散型データベースシステム（ＤＤＢＳ）全体で共通なデータモデルによって記述したスキーマである。この記述は各サイトの局所管理者（ＬＡ）によってなされる。又、局所概念スキーマ（ＬＣＳ）層には、この共通モデルに基づいた共通アクセス言語が設定されている。我々は、共通モデルとしては、Chen.P.P.SのＥ－Ｒモデル〔CHENP ７６〕を、共通アクセス言語としてはＱＵＥＬ〔ＨＥＬＤＧ　７５，ＹＯＵＳＫ　７７〕を採用した。これらについては第４章で詳論する。この様に局所概念スキーマ層は、データベースシステムの構文的側面、即ちデータモデルと言語とが同種化されたレベルに対応している。我々は局所内部スキーマ層から局所概念スキーマ層の生成過程を同種化（homogenization）と呼ぶ。この過程で各局所内部スキーマ層の異種性は、異種性情報（heterogeneity information or ＨＩ）として除去される。異種性情報は局所内部スキーマ層と、局所概念スキーマ層との対応情報である。逆に、局所概念スキーマレベルを発せられた問合せ（guery）を局所内部スキーマレベルのアクセス言語へ変換することを問合せ変換（query translation or ＱＴ）と呼ぶ。問合せ変換は同種化の逆過程と考えられ、同種化で除去された異種性情報を導入（利用）する過程でもある。問合せ変換については、第５章において詳論される。

全体概念スキーマ（ＧＣＳ）は、分散型データベースシステムの全体管理者（ＧＡ）によって、同種化で生成された各サイトの局所概念スキーマを基にして、あるネットワーク共同体（network community）にとって意味のあるデータの記述である。この記述は局所概念スキーマと同じくＥ－Ｒモデルを用いて行なわれ、全体概念スキーマ層のアクセス言語も同じくＱＵＥＬが設定される。この局所概念スキーマの集合から１つの全体概念スキーマを生成する過程を統合化（integration）と呼ぶ。統合化は、共通構文構造を持つ様に同種化された各データベースシステムの局所概念スキーマが表わすその異なった意味論的構造からある仮想的な１つの意味論的構造を表わす全体概念スキーマを生成（creation）する過程である。この過程は、要求工学における設

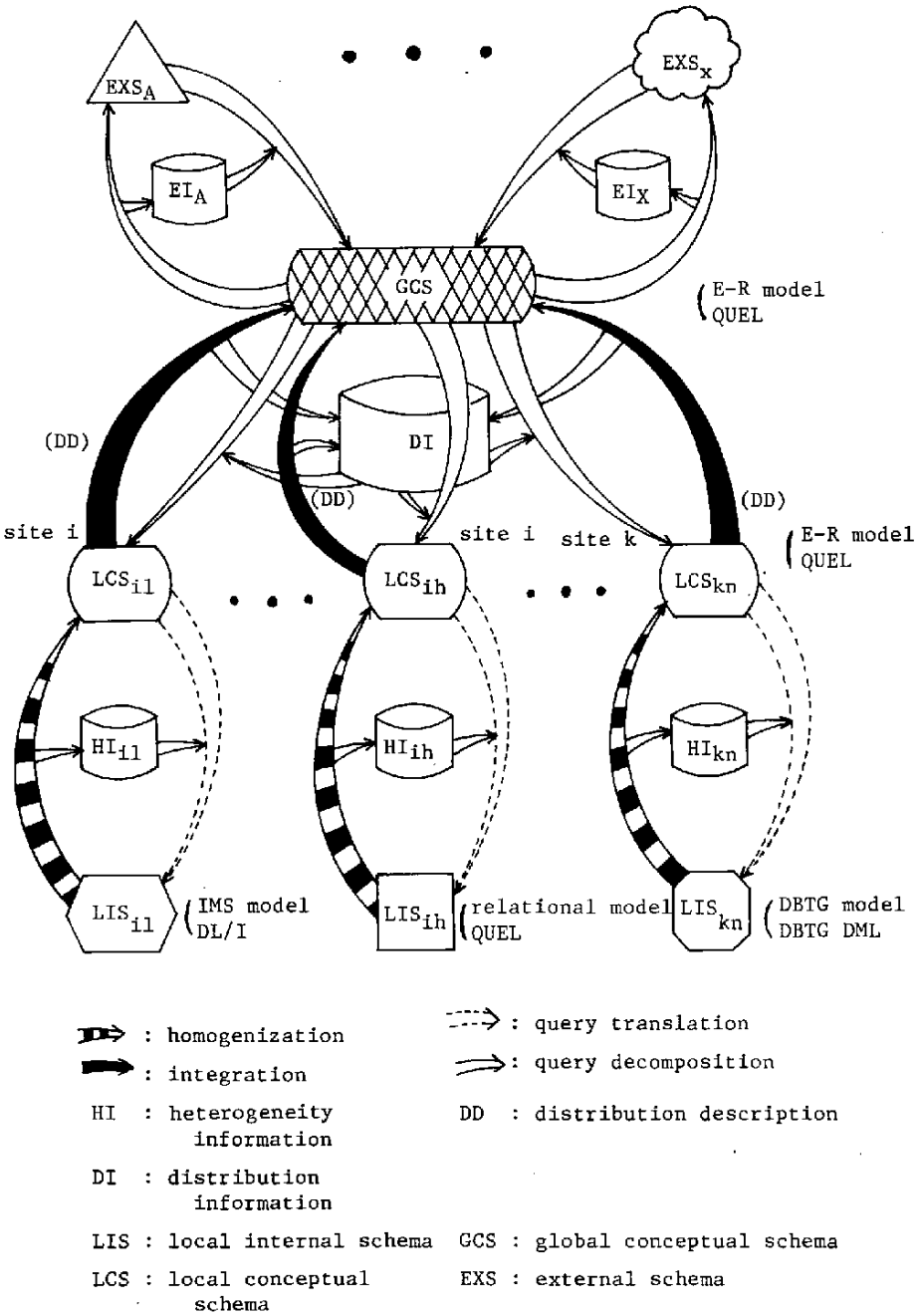


図 3.2 Four-Schema structure (FSS)

計過程に類似している。全体管理者は、各サイトの局所管理者との協議を通して、かつまたネットワーク共同体の要求を満足させながら、１つの全体概念スキーマをつくっていくわけである。全体管理者によってつくられる局所概念スキーマの集合からの全体概念スキーマの定義文の集合を分散記述（distribution description or DD）と呼ぶ。統合化において、各サイトのデータベースシステムの意味論的相違は分散情報（DI）として、それらの所在情報とともに除去される。これは又、全体概念スキーマ層と局所概念スキーマ層との対応情報である。全体概念スキーマ層では必要な情報がどのデータベースシステムにあり、そのデータベースシステムがどのようなタイプかを全く関知する必要がない。即ち、この層において、分散型データベースシステムは、１つのスキーマとしての全体概念スキーマと１つのアクセス言語としてのQUELとを備えた１つの巨大データベースシステムへと仮想化されたことになる。逆に全体概念スキーマ（GCS）に基づいたGCS問合せ（GCS guery or GQ）は、各サイトの局所概念スキーマ（LCS）に基づいたLCS問合せ（LCS guery or LQ）へ分割されねばならない。この過程を問合せ分割（guery decomposition or QD）と呼ぶ。これは又、統合化の逆過程であると考えられる。問合せ分割では統合化で除去された分散情報（DI）を逆に導入（利用）される。問合せ分割については、第７章において詳論する。

外部スキーマ（EXS）は、ネットワーク共同体内のあるアプリケーションにとって必要な全体概念スキーマ層のデータの部分集合にあたるデータを、このアプリケーションに適したデータモデルで記述したものである。外部スキーマは、全体概念スキーマからアプリケーション管理者（AA）によって記述される。外部スキーマ層は、アプリケーションに適したデータアクセス言語が、この外部スキーマを記述したデータモデルに基づいて設定される。アプリケーションはアプリケーションの利用性、セキュリティ等を考慮して外部スキーマを記述する。外部スキーマは、ANSI／X3／SPARCで言うところの〝外部スキーマ〟（external schema）と等価である。何故なら全体概念スキーマ層で分散型データベースシステム（DDBS）は１つの巨大システムへと仮想化されているために、この層のユーザはどのデータベースシステムが何のデータを持ち、そのデータベースシステムがどの様なタイプかを全く意識する必要がないからである。この意味で外部スキーマ（EXS）の問題は、分散型データベースシステムとは独立に議論できるものである。よって、我々は外部スキーマ問題は検討しなかった。これまでと同様に全体概念スキーマ層から外部スキーマ層の生成過程を、特殊化（specialization）、この逆過程を一般化（generalization）と呼ぶ。特殊化で除去され、一般化で導入される情報、即ち全体概念スキーマと外部スキーマとの対応情報を外部情報（external information or EI）と呼ぶ。

LISとLCS，LCSとGCS，GCSとEXSの各々のスキーマ間のマッピングにおける必

要情報は、各々異種性情報（ＨＩ）、分散情報（ＤＩ）、外部情報（ＥＩ）と呼ばれた。これらは、対応する２つのスキーマ層間の対応情報である。これらの情報は総称してネットワークデータディレクトリ（network data directory or NDD)と呼ばれる。ネットワークデータディレクトリについては次節で論じる。

同種化と異種性情報については４章、問合せ変換（ＱＴ）は５章、統合化と分散情報は６章、問合せ分割（ＱＤ）は７章で詳論する。

3.2. ネットワークデータディレクトリ（ＮＤＤ）

ネットワークデータディレクトリ（network data directory 以降 ＮＤＤと記す）は、分散型データベースシステムの処理が必要とする情報とその管理形態の総称である。ネットワークデータディレクトリ（ＮＤＤ）として管理される情報としては次のような種類がある。

1） 異種性情報（heterogencity information ）

2） 分散情報（distribution information ）

3） 外部情報（external information ）

異種性情報（ＨＩ）は、局所内部スキーマ（ＬＩＳ）層と局所概念スキーマ（ＬＣＳ）層との対応情報である。分散情報（ＤＩ）は、全体概念スキーマ（ＧＣＳ）層と局所概念スキーマ（ＬＣＳ）層との対応情報である。外部情報（ＥＩ）は、外部スキーマ（ＥＸＳ）層と全体概念スキーマ（ＧＣＳ）層との対応情報である。これらの３種類の情報は、ＮＤＤを考える上での基本単位となる。

ＮＤＤは、次の２点について論じられる必要がある。

1） 情報内容と種類

2） 各種類ごとの管理形態

情報の種類としては、前述した様に異種性情報（ＨＩ）、分散情報（ＤＩ）、外部情報（ＥＩ）の３種のものがある。これらの情報は、各々同種化、統合化、特殊化において除去されたもので、対応する２つのスキーマ層間の対応情報である。これらは又、各々問合せ変換（ＱＴ）、問合せ分割（ＱＤ）、一般化によって用いられる。各情報は、次の様な情報が成り立っている。

a） スキーマ要素間の対応情報

b） パフォーマンス情報

c） アクセス言語情報

a）は、２つのスキーマ層間のモデル要素の対応を示している。この対応関係は、スキーマ層設計時に対応する管理者によって定義される。即ち、データ表現方法の変換情報である。

b）は、上位のスキーマ層における問合せから、下位のスキーマ層における最適な問合せを生成す

るために必要な情報である。例えば、局所概念スキーマ問合せから局所内部スキーマＤＭＬ（問合せ）への変換の最適化には、局所内部スキーマ層でのスキーマ要素の持つカーディナリティ（cardinality），選択度（selectivity）等〔第５章〕が必要となる。

最後の情報は、対応するスキーマ層の持つアクセス言語の構文規則等についての情報である。異種性情報とその使われ方については、各々4.5節と５章において論じられており、分散情報とその使われ方については、各々6.3節と第７章において論じられているので参照されたい。外部情報（ＥＩ）については、本報告書では触れられていない。

２）の管理形態としては、次の２点を考える必要がある。

ａ）　分散型データベースシステムのどのスキーマ層で管理されるか。即ち、どのようなデータモデルで表わされ、どのような言語でアクセスされるか。

ｂ）　ＮＤＤ情報の分散形態

ＮＤＤ情報を考える時、ｂ）の情報の分散形態即ち、ＮＤＤ情報とその所在を常に考えねばならない。従って、我々は、ＮＤＤ情報は分散型データベースシステムの局所概念スキーマ（ＬＣＳ）層で管理されるべきであると考えている。このスキーマ層は、分散型データベースシステム全体で共通な記述系（データモデル）とアクセス系（問合せ言語）を提供しているとともに、情報の所在するデータベースシステムのサイト情報を視ることができるからである。従がって、ＮＤＤ情報、即ち分散情報、異種性情報、外部情報は、局所概念スキーマ層のデータモデル、即ち（Ｅ－Ｒモデルの）リレーショナルモデル記述〔Ｅ－Ｒモデルとリレーショナルモデルの関係性については4.2節を参照されたい〕によって表わされる。

次にＮＤＤ情報の分散形態を考えよう。上述した３種の情報は、対応する３つの問合せ処理において必要とされるものである。問合せ処理効率が、分散型データベースシステム全体のパフォーマンスを決定することから、処理とそれの必要とする情報は同一サイトにあることが必要である。もし、異なったサイトにあれば、問合せ処理は必要な情報を得るためにその処理時間のほとんどを費やしてしまうだろう。このことは次の点を導く。

ａ）　あるデータベースシステムの異種性情報は、問合せ変換（ＱＴ）を行なうサイト、即ちこの情報に対応するデータベースサイトと同じサイトに存在する。

ｂ）　分散情報は、問合せ分割を行なうサイトにその完全コピーが存在する。もし各サイトが問合せ分割を行なうのであれば、全サイトに分散情報の完全コピーが冗長に置かれることになる。

分散形態を考える上でもう１つの重要な点は、異種性情報と分散情報とは共有情報を互いに保有する点である。分散情報内に存在する異種性情報としては次のものをあげることができる。

a） 局所概念スキーマ要素（スキーマ情報）

b） 局所概念スキーマ要素のカーディナリティ，サイズ（a）についてのパフォーマンス情報）

我々は前者をスキーマ情報、後者をパフォーマンス情報と呼ぶ。これらの２つの情報は、互いに異なった性質を持っている。前者のスキーマ情報は、比較的長いライフサイクルを持っているのに対して、後者は短かいライフサイクル、即ち動的な性質を持っている。前者の静的な情報が、異種性情報と分散情報間に冗長に存在することは、格納コストが問題となるが管理上さして問題とはならない。しかし、後者のような動的情報の冗長性は問題である。これは第２章で述べたように複雑なＣＣ制御（concurrency and consistency control）のための膨大な通信オーバヘッドをもたらしてしまう。又、分散情報を各サイトが保有することは、さらに冗長度を高め、ＣＣ制御をより複雑化させる。このことは、ＮＤＤ情報の管理という視点からみると、分散情報は、パフォーマンス情報を保持しないことを指示している。もし問合せ分割（ＱＤ）が、このようなパフォーマンス情報を用いずに処理を行なえるならば、分散情報はパフォーマンス情報を保持しない方が望ましい。我々は、第７章で論じる問合せ分割では、このような観点から、パフォーマンス情報を不要とするアルゴリズムを提案を行なう。

3.3. 四層スキーマ構造に基づいたシステムアーキテクチャ

これまでに、分散型データベースシステムの全体アーキテクチャとしての四層スキーマ構造（以降ＦＳＳと略す）と、これが必要とする情報、即ちネットワークデータディレクトリ（以降ＮＤＤと略す）の管理形態について論じてきた。本節では、まず四層スキーマ構造（ＦＳＳ）で必要となる処理機能と各機能間のインタフェースを明らかにする。ついで、分散型データベースシステムのシステムアーキテクチャについて述べる。

四層スキーマ構造（ＦＳＳ）に基づいた分散型データベースシステムの処理機能と機能間のインターフェースを図4.3に示す〔ＴＡＫＩＭ ８０ａ〕。この図中の記述法は、ＡＮＳＩ／Ｘ３／ＳＰＡＲＣ〔ＡＮＳＩＸ ７５，ＴＳＩＣＤ ７８〕に基づいている。図3.3のなかで、四角の箱は処理機能を表わし、六角形は、人間（ユーザと管理者）を示し、三角形はこれらが必要とする情報、即ちネットワークデータディレクトリ（ＮＤＤ）を表わしている。両方向アークを持った線とこの線内の棒とは（  ）処理機能（人間も含む）間のインタフェースを示している。又、これに付加されている数字は、インタフェースの番号を表わしている。この図のＮＤＤより上半分は分散型データベースシステムの設計を、下半分はアクセスを表わしている。まず処理機能の各々について簡単に述べる。同種化システム（homogenizer）は、各データベースシステムサイトにあって、そのサイトの局所管理者（local administrator）とともにそのデータベースシステム

の局所内部スキーマ（ＬＩＳ）層から局所概念スキーマ（ＬＩＳ）層の生成、即ち同種化を行なう。これとともに、そのデータベースシステムに関する異種性情報（ＨＩ）を生成し、ネットワークデータディレクトリ（ＮＤＤ）に格納する。局所管理者は、局所内部スキーマから局所概念スキーマを定義言語を用いて記述する。同種化システムは、これを入力として、異種性情報を生成し、ネットワークデータディレクトリ（ＮＤＤ）に格納する。

統合化システム（integrator）は、全体管理者（global administrator）とともに、局所概念スキーマ（ＬＣＳ）層から全体概念スキーマ（ＧＣＳ）層の生成、即ち統合化を行なう。これとともに分散情報（ＤＩ）を生成し、ＮＤＤに格納する。全体管理者は、各サイトの局所概念スキーマを視、かつネットワーク共同体の要求に合う様な全体概念スキーマを定義用言語を用いて定義し、これを統合化システムにわたす。統合化システムは、この定義記述から、分散情報を生成し、ＮＤＤに格納する。

外部スキーマ（ＥＸＳ）生成システム（ＥＸＳ processor）とアプリケーション管理者（application administrator）とは、全体概念スキーマ（ＧＣＳ）層から外部スキーマ（ＥＸＳ）層の生成を行ない、外部情報（ＥＩ）をネットワークデータディレクトリ（ＮＤＤ）に格納する。アプリケーション管理は、全体概念スキーマを視ながら、そのアプリケーションの要求に合う様な外部スキーマを定義用言語を用いて記述し、外部スキーマ生成システムにわたす。外部スキーマ生成システムは、定義記述から外部情報を生成し、ＮＤＤに格納する。

局所内部スキーマ（ＬＩＳ）生成システムは、そのデータベースシステム内のデータの中で、分散型データベースシステムとして利用可能なものの記述、即ち局所内部スキーマを生成する。

以上は、統合型（bottom － up）分散型データベースシステムの設計過程に対応している。これに対して、分割型（top － down）設計の処理機能と機能間の関係を図 3.4 に示す。図3.3との最大の相違点は、分割システム（decomposer）と異種化システム（heterogenizer）である。まず全体管理者は、全体概念スキーマを定義する。分割システムは、ある目的関数のもとで最適となる様に、各サイトの局所概念スキーマを決定する。これとともに、分散情報をＮＤＤに格納する。異種化システムは、この局所概念スキーマを入力として、自分のデータベースシステムに対応する局所内部スキーマを生成する。分散型データベースシステムのアクセス部分は、分割型も統合型も同一である。以降アクセス部分について考える。

ＥＸＳ／ＧＣＳ変換システム（ＥＸＳ／ＧＣＳ transform）は、外部スキーマ層の問合せを、全体概念スキーマ層の問合せへの変換を、ＮＤＤ内の外部情報（ＥＩ）を用いて行なう。

問合せ分割（ＱＤ）システムは、全体概念スキーマ層の問合せから、局所概念スキーマ層の問合せへの変換（分割）を行なう。この時必要となる情報は、ＮＤＤ内の分散情報（ＤＩ）である。

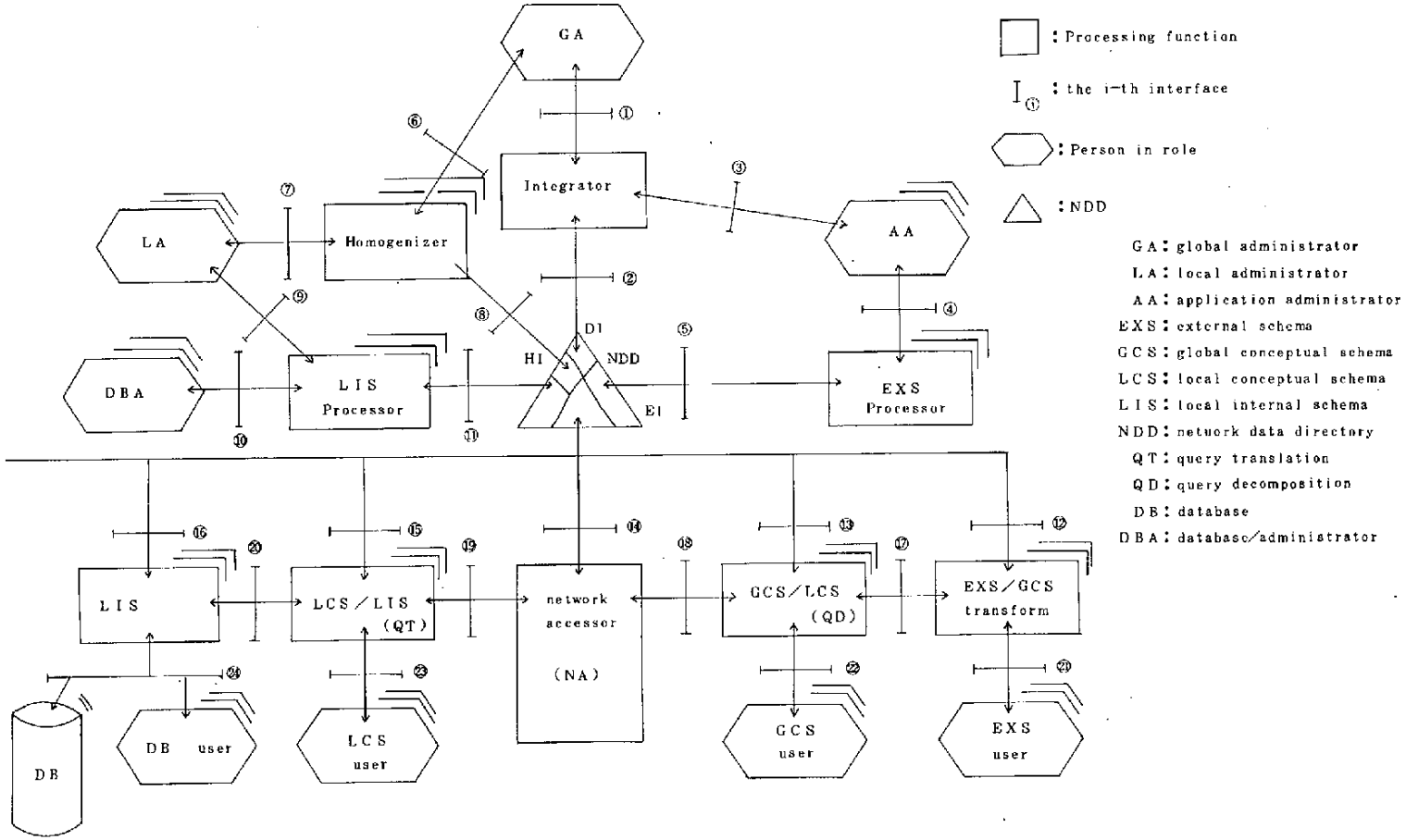


図 3.3　分散型データベースシステムの処理機能と機能間インタフェース

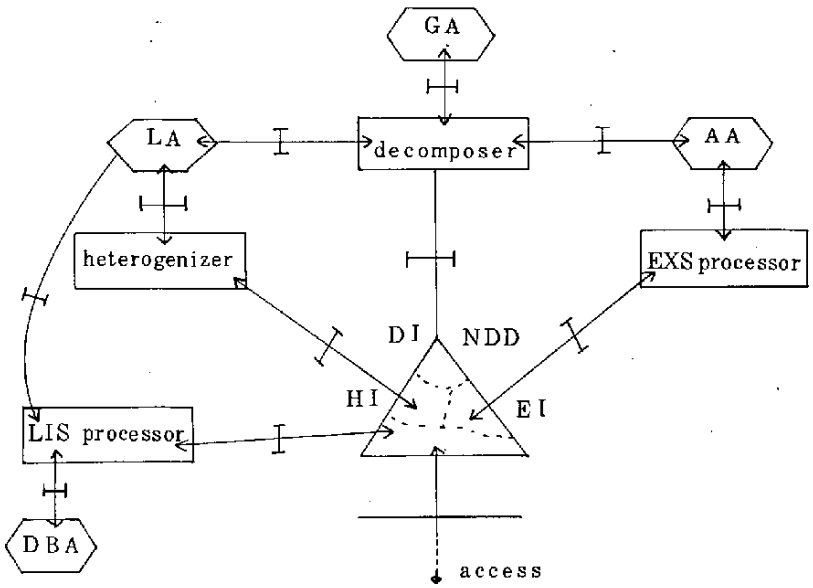


図3.4　分割型ＤＤＢＳ設計

全体概念スキーマ層の問合せ（我々は、これをＧＣＳ問合せと呼ぶ）は、一般に、複数の局所概念スキーマを参照することができる。このため問合せ分割システムは、全体概念スキーマ要素から局所概念スキーマ要素への表現の変換とともに、サイト間の問合せ処理を行なう必要がある。複数サイトにまたがった問合せを処理するためには、サイト間でデータの転送とともに各サイトでの処理の同期の管理も必要になる。これらは、コンピュータネットワークのプロトコルを用いて行なわれる。こうしたネットワークの制御を行なうものとしてネットワークアクセスシステム（network accessor）（以降、ＮＡと略記する）がある。ネットワークアクセスシステム（ＮＡ）は、ある通信プロトコルを用いて問合せ分割システムの必要とするコマンドを、必要な相手へ届け、その応答を再び問合せ分割システムへ返す役目をもっている。

問合せ変換システム（ＱＴ）は、局所概念スキーマ層の問合せ（我々はこれをＬＣＳ問合せ）から、そのデータベースシステムで実行可能な局所内部スキーマ層のアクセス言語プログラムを生成する。この時必要となる情報は、ネットワークデータディレクトリ（ＮＤＤ）内の異種性情報（ＨＩ）である。問合せ変換システムと異種性情報とは、各データベースシステムと同じサイトにある。

上述してきた処理機能については、図3.5にまとめてある。これらの機能間のインタフェースについては図3.6に、ネットワークデータディレクトリについては図3.7に、さらに管理者について図3.8にまとめてある。

処 理 機 能

processing functons	説　　明	points to be discussed
homogenizer (LCS processor)	ＬＩＳからＬＣＳの生成	共通モデル ＬＩＳとＬＣＳとの対応法 homogenization
integrator (GCS processor)	ＬＣＳの集合から ＧＣＳの生成	共通モデル ＬＣＳとＧＣＳの対応方法 データのセマンディクス integration
EXS processor	ＥＸＳの生成	application－oriented data model,application の semantics
EXS/GCS transform	EXS query→ GCS queries	EXS language 共通言語
GCS/LCS transform (QD)	GCS query→ LCS queries	共通言語
LCS/LIS transform (QT)	LCS query→ LIS DMLs	共通言語 DB－specific DML
network accessor (NA)	transport network command の生成	NA command
LIS transform	DML → access method	
LIS processor	ＬＩＳの生成	

図 3.5　処理機能

インターフェース

インターフェース番号	説　　明	data through the interface and operations
1	GCS description - source format	DD
2	GCS description - object format	DI
3	GCS description - display format	HVE-R diagram GCS RD
4	EXS description - source format	
5	EXS description - object format	
6	LCS description - display format	E-R diagram LCS RD
7	LCS description - source format	LIS DDL
8	LCS description - object format	HI
9	LIS description - display format	
10	LIS description - source format	DDL
11	LIS description - object format	
12.13.14. 15.16.	NDD interface	12 EI　　15 HI 13 DI 14 PI
17, 22	GCS query	QUEL (petrieval / update)
18 19, 23	network access command LCS query	 QUEL (retrieval / update)
20	DBMS DML	communication level (query / host language)
21	EXS query	retrieval update

図3.6　インターフェース

NDD	説　．　明	処理機能との関連
DI	LCSとGCSとの対応情報	GCSprocessor（Integretor）で生成 QDで利用
HI	LISとLCSとの対応情報	LCSprocessor(homogenizer)で生成 QTで利用
EI	GCSとEXSとの対応情報	EXS processorで生成 EXS/GCS transformで利用
PI	network accessorの必要情報	NAで利用

図3.7　ネットワークデータディレクトリ

Persons in role	説　　明	points to be discussed
GA (global administrator)	LCSの集合からGCSの定義 LCSとGCSのmappingの定義	・schema design ・GCSのsemantics ・networkcommunity とは？
LA (local administrator)	LISからLCSの定義 LISとLCSとのmappingの定義	・model transformation ・common model ・syntactic structure
AA (application administrator)	GCSからEXSの定義 GCSとEXSとのmappingの定義	・schema design ・EXSのsemantics ・Application とは？
DBA	LISの定義	・schema design ・storage structure ・schema level

図 3.8　管理者

次に、これまで述べてきた処理機能とこれらの関係性を基にして、分散型データベースシステムのシステムアーキテクチャを考える。我々は、これを統一アーキテクチャ（unified architecture）と呼び、その概要を図3.9に示す。分散型データベースシステムのサイト（site）とは、1つのホストコンピュータシステムと定義する。サイトは1つ又はそれ以上のデータベースシステムから成っている。各サイトは、分散型データベースシステムの管理機能として、次の4種の論理要素から成っている。

a） ネットワークアクセスシステム（network accessor or NA）

b） 全体データベースプロセッサ（global database processor or GDP）

c） 局所データベースプロセッサ（local database processor or LDP）

d） ユーザインタフェース（user-friendly interfaces）

ネットワークアクセスシステム（NA）は、各サイトに1つ存在し、他のサイトの全体及び局所データベースプロセッサとの通信とともに、自分のサイト内の全体データベースプロセッサ（GDP）、局所データベースプロセッサ（LDP）間の通信手段を提供する。分散型データベースシステム内の全てのGDPとLDPはユニークな識別名を持っている。GDPとLDPは、メッセージに相手の識別名を付してネットワークアクセスシステム（NA）にわたす。NAは、この識別名の存在するサイトを明らかにして、メッセージをネットワークへ流す。このサイトを明らかにする情報も又自分のサイト内のLDPが管理している。

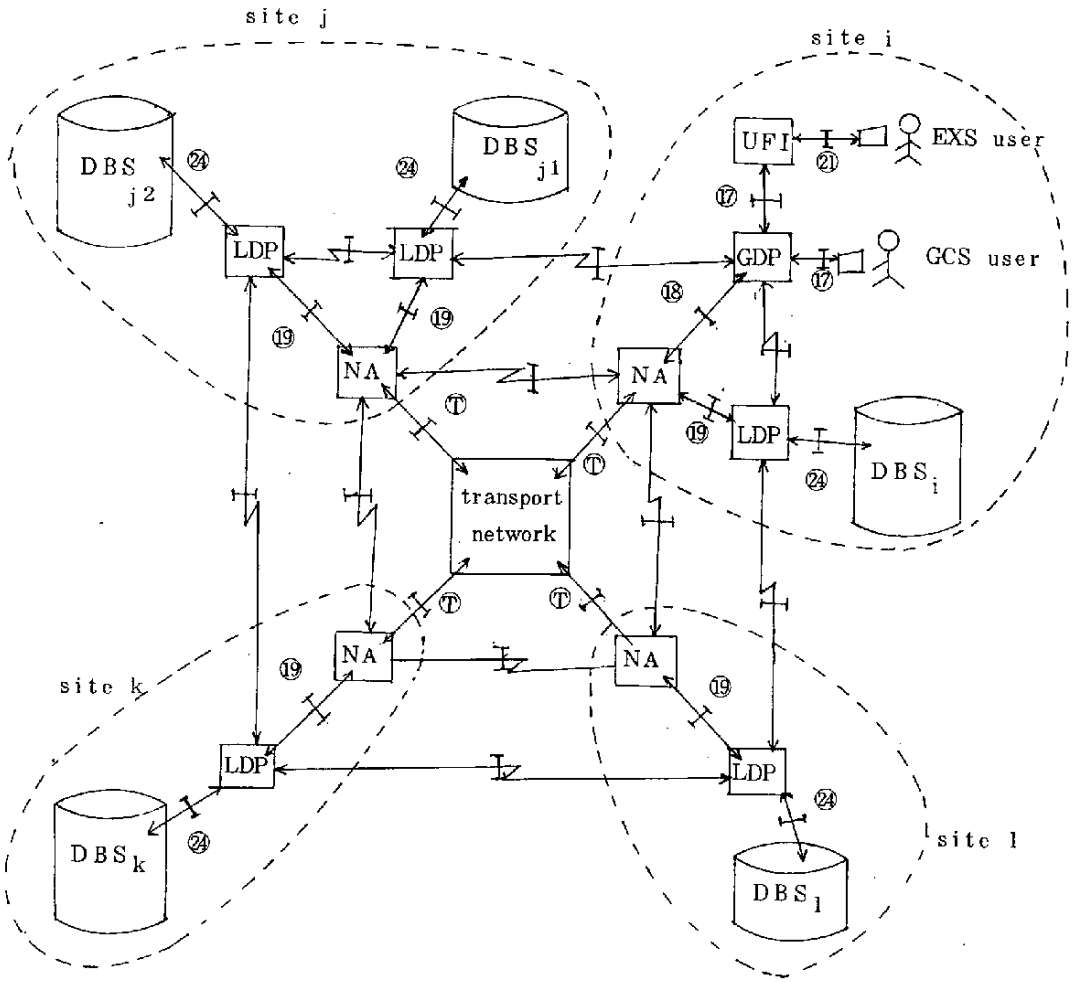
全体データベースプロセッサ（GDP）は、各サイトに1つ存在する。これは次の機能を持つ。

a） 問合せ分割（QD）機能

b） 分散情報（DI）の管理

c） 統合化機能

全体概念スキーマ（GCS）層のユーザの発したGCS問合せは、まず全体データベースプロセッサ（GDP）が受けとる。GDPは、このGCS問合せを解析し、分散情報（DI）を用いて必要な情報を持っているサイトの局所概念スキーマ（LCS）問合せへ分割する。こうして分割されたLCS問合せはネットワークアクセスシステム（NA）を介して目的とするサイトの局所データベースプロセッサ（LDP）へ送出される。局所データベースプロセッサからの応答もまたネットワークアクセスシステムを介してGDPへ届けられる。我々のシステムでは、1つのGCS問合せは、それが発せられたサイトの1つのGDPと、必要な情報を保持しているデータベースシステムに対応するLDPの集合とによって処理される。このユーザと同一のサイトのGDPを統合全体データベースプロセッサ（coordinate global database processor or CGDP）と呼ぶ。



UFI : user-friendly interface
GDP : global database processor
LDP : local database processor
NA : network accessor
DBS : database system
Ⓣ : transport interface
: interface
: protocol

図3.9 統一アーキテクチャ

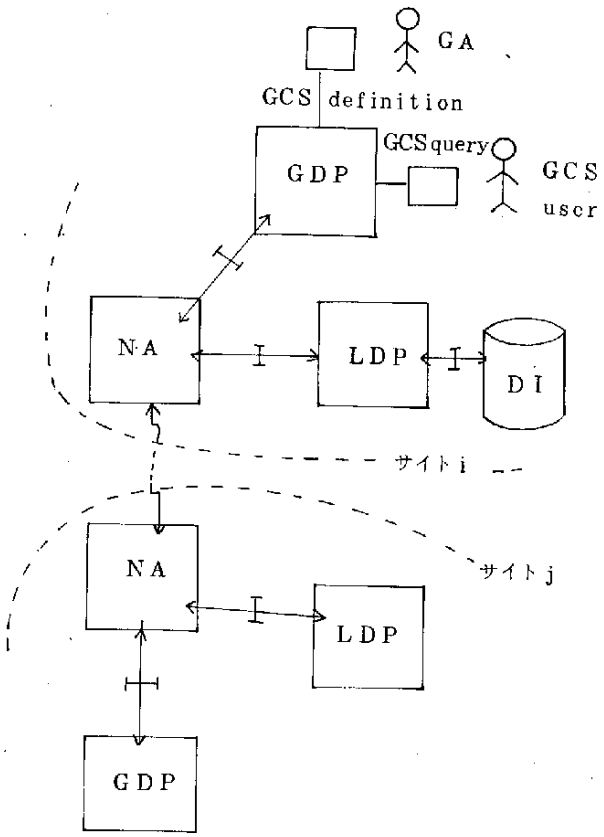


図3.10　GDP and LDP

全体データベースプロセッサ（ＧＤＰ）は、統合化システムの機能も持つ。全体管理者（ＧＡ）が、全体概念スキーマ（ＧＣＳ）の定義記述をＧＤＰに与える 、ＧＤＰは必要な異種性情報をアクセスしながら分散情報を生成する。この分散情報を他のＧＤＰへ送る。各ＧＤＰは、分散情報（ＤＩ）を自分のサイト内のＬＤＰの１つに管理させる。又、分散情報に対する更新も、更新を受けたＧＤＰが、この分散情報を保持するＧＤＰを制御しながら、これらの分散情報の更新を行なう。

局所データベースプロセッサ（ＬＤＰ）は、１つのデータベースシステムに対して１つ存在する。ＬＤＰの機能としては、次のものがある。

a）　問合せ変換（ＱＴ）機能

b）　異種性情報（ＨＩ）管理

c）　同種化機能

d）　作業領域（ＷＳ）管理

局所データベースプロセッサ（ＬＤＰ）は全体データベースプロセッサで分割されたＬＣＳ問合せを受けると、これを自分のデータベースシステムが実行可能な局所内部スキーマ（ＬＩＳ）アクセス言語プログラムへ変換する。この時ＬＤＰは、そのデータベースシステムに関する異種性情報（ＨＩ）を変換情報として用いる。

異種性情報は、ＬＤＰ内の同種化システムによって生成される。ＬＤＰのデータベースシステムの変化によって局所内部スキーマが変化した時は、ＬＤＰによって異種性情報が更新され、全ＧＤ

Pへこの変化を知らせる。GDPは変化に対応して分散情報の更新を行なう。このサイト内のデータベースシステムについての異種性情報は、特別なLDPが管理する。この特別なLDPを異種性情報管理局所データベースプロセッサ（heterogeneity information management local database processor or HIMLDP）と呼ぶ。HIMLDPは、これ自身の異種性情報を持たないことが特徴である。HIMLDPはリレーショナルデータベースシステムとしてインプリメントされることが望ましい。

LDPはこの他にサイト間での問合せ処理のための作業領域（working space or WS）を持つ。このWSは、HIMLDPと同じくリレーショナルデータベースシステムとしてLDPによって管理される。他のサイトから送られてくるデータや、サイト間処理のための中間結果がこのWSに格納される。

4. 同種化と異種性情報〔TAKIM 79a〕

統合型（bottom-up）の分散型データベースシステム（ＤＤＢＳ）は、一般に既存の種々のタイプのデータベースシステム（ＤＢＳ）から構成されている。分散型データベースシステムを構築するためには、まず第１に構成要素たる各データベースシステムの異種性を除去せねばならない。我々の四層スキーマ構造（ＦＳＳ）アプローチ〔ＴＡＫＩＭ　７８，ＴＡＫＩＭ　７９ａ〕では、データベースシステムのスキーマ、即ち局所内部スキーマ（ＬＩＳ）、を分散型データベースシステム全体で共通なデータモデルによって記述し、共通アクセス言語を設定することによって異種性を解決する。我々は共通モデルとしてＥ－Ｒモデル〔ＣＨＥＮＰ　７６〕を、共通アクセス言語としてＱＵＥＬ〔ＨＥＬＤＧ　７５〕を採用した。この過程を同種化（homogenization）と呼び、本章では同種化について論じる。

まず4.1ではデータベースシステムの異種性とは何かを明らかにする。ついで共通モデルとしてのＥ－Ｒモデルを4.2に、共通言語としてのＱＵＥＬを4.3に論じる。4.4では既存データモデル（リレーショナル、ＤＢＴＧ、ＩＭＳモデル）の一般構造を解明する。4.5では同種化過程と異種性情報について、３つの既存データモデルの各々について論じる。4.6では同種化の例を３種のデータベースシステムについて示す。4.7では、我々のインプリメントしたＤＢＴＧ　ＬＩＳからＬＣＳへの同種化システム(homogenizer)としての異種性情報処理システム（ＨＩＰＳ）について述べる。

4.1　異種性の定義

データベースシステム（ＤＢＳ）は次の５つの要素によって特徴づけられる。

１）　データモデル

２）　データ言語

３）　格納されたデータのセマンティクス構造

４）　ＤＢＭＳの機能性

５）　ホストコンピュータ

よってデータベースシステムの異種性とは、これら点の各々の相違であると定義できる。

データベースシステムを利用しようとする場合、まずユーザは、スキーマ（schema）を通して格納されたデータベース（ＤＢ）の意味を知ることが出来る。ここでスキーマは、ＤＢＳ固有のデータモデルによってＤＢの意味を記述したものである。次に、このスキーマに基づいて、ＤＢＭＳ

の提供するアクセス言語を用いて自分の必要とするデータをアクセスする。よって、ユーザにとって、データモデル、データ言語、そしてデータベースのセマンティクスは、データベースシステムを特徴づける最も主要な要素となる。

既存データモデルとして、リレーショナルモデル〔ＣＯＤＤＥ　７１〕、ネットワークモデル〔ＣＯＤＡＳ　７３〕、階層型モデルの３つが典型的である。ネットワークモデルの代表例として、ＣＯＤＡＳＹＬ　ＤＢＴＧ〔ＣＯＤＡＳ　７３、７８〕モデルを指摘できる。階層型モデルとしてはＩＢＭのＩＭＳモデルがある。よって、この３つのデータモデルの相異はデータベースシステムを考えるうえでの第１の異種性である。

各データベースシステムは、スキーマを記述するための言語とスキーマに基づいてデータベースをアクセスするための言語とを備えている。前者をデータ記述言語ＤＤＬ、後者をアクセス言語（ＤＡＬ）と呼ぶことにする。同種化の対象としては後者のアクセス言語のみを考える。理由は、データベース管理者（ＤＢＡ）によって既にＤＤＬによって記述されたスキーマのレベルを同種化の対象としているからである。ＤＤＬを含めて言語は、前述したデータモデルと密接に関連し、互いに不可分であると言える。例えば、ＤＢＴＧモデルはＤＢＴＧ　ＤＭＬを、ＩＭＳモデルはＤＬ／Ｉを、リレーショナルモデルは関係代数又は関係計算に基づいた問合せ言語（guery language）を持っている。ここではリレーショナルモデルの言語としては、関係計算に基づいた問合せ言語ＱＵＥＬ〔ＨＥＬＤＧ　７５〕を仮定することにする。このようなＱＵＥＬ、ＤＢＴＧ　ＤＭＬ、ＤＬ／Ｉという３つの言語の相違は第２の異種性である。

データモデルと言語とは不可分なものであることを先に述べた。しかしこのことはデータベースシステムと言語との不可分性を意味していない。我々はデータベースシステムは、ＡＮＳＩ／Ｘ３／ＳＰＡＲＣ〔ＡＮＳＩＸ　７５，ＴＳＩＣＤ　７８〕の３つの階層（内部スキーマ、概念スキーマ、外部スキーマ）からなっており、各々の階層は、あるデータモデルと言語とを備えていると仮定している。分散型データベースシステム（ＤＤＢＳ）を構成するデータベースシステム相互の通信レベルとしてどの階層を用いるかは重要な問題であるが、我々は、概念スキーマまたは外部スキーマレベルを考えている。内部スキーマは、データベースシステムの物理構造に依存し、データベースアクセスの有効性に関して考慮されたレベルである。これをデータベースシステム相互の通信レベルとすると、データベースシステムの物理構造の変化のたびに分散型データベースシステムを変化させねばならなくなってしまう。このため、データベースシステムの物理構造とは独立な概念スキーマ又は外部スキーマレベルを通信レベルとして設定した。どちらを選択するかは、分散型データベースシステム（ＤＤＢＳ）に参加しようとするサイトのデータベース管理者（ＤＢＡ）と分散型データベースシステム（ＤＤＢＳ）の全体管理者（ＧＡ）とが、ＤＤＢＳに対するユーザの利

用要求、各データベースシステムのセキュリティ等を考慮して決定される。

第３番目のデータベースシステムのセマンティクスは、統合型分散型データベースシステムを考える時に、最も主要な要素となる。先述したように、分散型データベースシステム（ＤＤＢＳ）の問題の１つは、異なったセマンティクスデータ構造を持ったデータベースシステムをいかに１つのセマンティクスデータ構造へマッピングするかである。この問題は、データモデルと言語とを同種化した後の問題であり、統合化（integration）問題（第６章）のなかに論じられる。

第４番目のデータベース管理システム（ＤＢＭＳ）機能性としては、次のものがある。

ⅰ）　検索

ⅱ）　更新（追加、置換、消去）

ⅲ）　インテグリティ制御　（integrity control）

ⅳ）　同時実行制御　　（concurrency control）

ⅴ）　セキュリティ制御　（security control）

ⅵ）　リカバリ

ⅶ）　レポート生成

ⅷ）　アプリケーション向けの計算（統計処理etc.）

我々は、これらのなかで、ⅰ）のみを考えることにする。ⅰ）はどのＤＢＭＳも共通的に持つ最も基本的機能である。ⅱ）～ⅶ）の問題は今後の課題とし、まず検索について考えていくことにする。

最後のホストコンピュータの相違については考えないことにする。我々は、種々のコンピュータの相違は、通信ネットワークによって解決するものとしている。

以上のことより、同種化とは、分散型データベースシステムを構成する各データベースシステムのデータモデルとデータ言語との相違を、克服する過程であると再定義することができる。即ち、データベースシステム（ＤＢＳ）の異種性とはこの構文的要素たるデータモデルと言語の相違である。

4.2　共通モデル　―　Ｅ―Ｒモデル

4.1において、まず同種化すべき異種性として、データモデルとアクセス言語との２つがあることを論じた。我々の同種化は次の２つから成り立っている。１）各データベースシステム（ＤＢＳ）サイトにおいて、ＤＢＳ固有のデータモデルで記述されたスキーマ、即ち局所内部スキーマ（ＬＩＳ）、を分散型データベースシステム（ＤＤＢＳ）全体で共通なデータモデルによって記述する。ＬＩＳを共通モデルによって共通記述したものを局所概念スキーマ（ＬＣＳ）と呼ぶ。

２）ＤＤＢＳ全体に、この共通モデルに基づいたアクセス言語を設定する。

我々は、共通モデルとして、Chen, P.P.S によるE－Rモデル（entity-relationship model）〔CHENP　76〕を採用した〔TAKIM　79a〕。E－Rモデルを採用した理由は、次のようである。

1）　E－Rモデルは事象集合（entity-set）と関係性集合（relationship-set）の概念を最も簡潔な形式で備えている。この概念の欠如は、リレーショナルモデルのセマンティクス問題として種々指摘されている。しかし、リレーショナルモデルの最大の長所は、数学的基礎に基づいた簡潔さである。よって、セマンティクネットワーク〔MYLOJ　76〕の様に、より多くのセマンティクスをデータモデルに持ちこむことは、セマンティクス記述能力を高めるが、一方では記述とアクセスを複雑で困難なものとしてしまう。

2）　E－Rモデルは、リレーショナルモデルと良く対応できる。
このことは、E－Rモデルの実現をリレーショナルモデルで行なえることを意味している。即ち、データベースの概念的な意味をE－Rモデルで記述し、実際のデータの記述とアクセスは、この記述に基づいてリレーショナルモデルによって行なうことができる。

4.2.1　E－Rモデル〔CHENP　76〕

本節では、各データベースシステムのスキーマを共通記述するための共通モデルとしてのE－Rモデルについて述べる。

データモデルとは、データのセマンティカルな構造を表わすための記述系である。格納された、データベースの持つ意味は、あるデータモデルを用いて記述され、この記述されたものがスキーマ（schema）と呼ばれるものである。

まずデータベースの表わそうとする世界の意味（セマンティクス）について考えてみよう。データベースの意味は、最も単純には、データベースの表わそうとする世界の最少要素、これらの集合、及び集合間の関連によって表わせる。このような最少要素を事象（entity）と呼び。事象とは〝太郎〟、〝国語〟といったデータベースが表わそうとする世界の最少要素である。事象のなかで共通的性質を持ったものの集合を事象集合（entity-set　or　ES）と呼ぶ。例えば、事象としての〝太郎〟〝花子〟〝三郎〟は、全てどこかの学校の生徒であるとすると、これらの３つの事象は事象集合〝生徒〟を形成する。〔図4.1〕。同様に〝国語〟〝算数〟〝理科〟といった事象の集合は、事象集合〝課目〟となる。

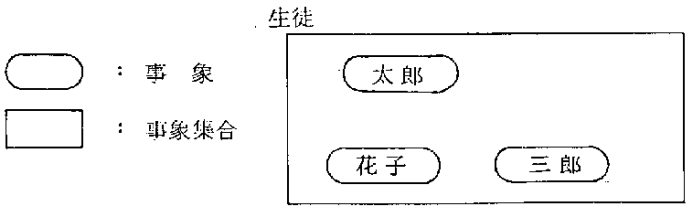


図4.1　事象と事象集合（生徒）

次に２つの事象集合〝生徒〟と〝課目〟について考えてみよう。例えば、〝太郎は算数と理科が得意で、花子は国語が、三郎は算数が得意である〟とすると、この事実は、図4.2のように表わせる。即ち、事象集合〝生徒〟と〝課目〟とは関係性集合（relationship-set or RS）〝得意課目〟によって関連づけられている。

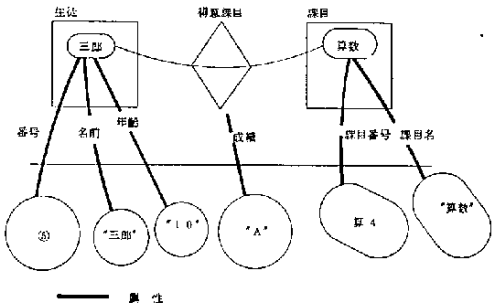


図4.3　事象、ES、RS、と属性、値集合

このように、データベースの意味は事象集合（ＥＳ）、関係性集合（ＲＳ）を用いて、概念的に示すことができる。実際のデータベースでは、事象と関係性という要素は値として表わされる。これは、これらのこの事象及び関係性と、実際の値とのマッピングを属性（attribute）と言う。例えば、生徒は番号、名前、年齢という属性を持っている〔図4.3〕。属性のとり得る値の集合を値集合（value set or VS）という。属性とは、事象集合（ＥＳ）又は関係性集合（ＲＳ）から値集合へのマッピングとして定義される。

さらに、ある事象集合内で同一の関係性集合（ＲＳ）と関連している事象をこの関係性集合に関連した事象（associated entity）と呼ぶ。関連した事象の集合は、この関係性内で事象が演じる役割（role）を持っている。例えば前の例で、関係性集合〝得意課目〟において、これと関連する事象集合〝生徒〟内の事象は〝優秀な生徒〟という役目を持つことになる。

Ｅ－Ｒモデルは、上述した様な事象集合、関係性集合、属性、値集合とから成っている。

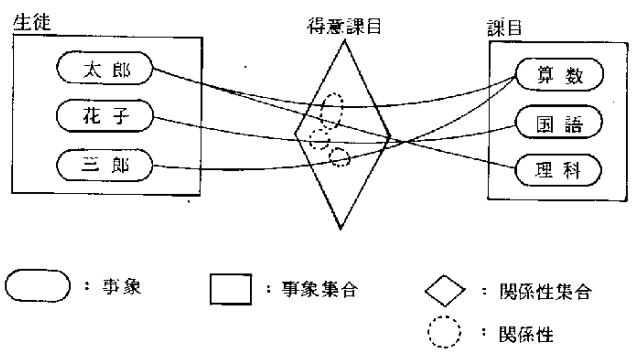
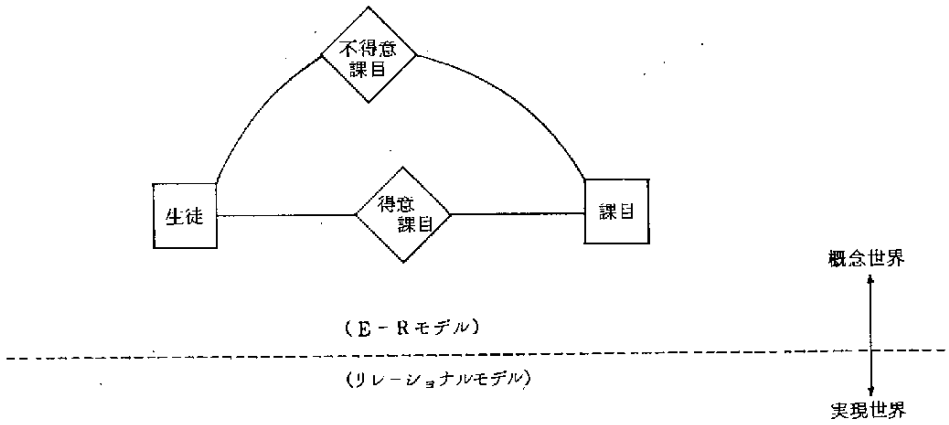


図4.2　事象、事象集合、及び関係性集合

4.2.2　E－Rモデルとリレーショナルモデルとの関係

次に、E－Rモデルとリレーショナルモデルとの関係について考えてみよう。先に、E－Rモデルを採用した理由として、これとリレーショナルモデルとが良く対応できることを述べた。図4.4は、E－Rモデルとリレーショナルモデルとの対応関係を図示している。

事象集合"生徒"は番号、名前、年齢という3つの属性を持っている。リレーショナルモデルでは、事象集合"生徒"に対してリレーションスキーム生徒（番号、名前、年齢）を対応させられ



生徒

番号	名　前	年齢
41	太　郎	10
62	花　子	12
59	三　郎	11
11	百　恵	6
45	五　郎	10
：	：	：

得意課目

番号	課目番号	成績
41	算 4	A
41	理 4	A
62	国 6	A
59	算 5	B
11	音 1	A
11	図 1	B
45	算 4	A

課目

課目番号	課目名
算 4	算　数
理 4	理　科
国 6	国　語
算 5	算　数
音 1	音　楽
図 1	図　工
算 6	算　数
	：

不得意課目

番号	課目番号	成績
62	体 6	D
45	理 4	E
11	社 1	D

図4.4　E－Rモデルとリレーショナルモデル

る。リレーション生徒内の各組（tuple）は、事象集合〝生徒〟内の各事象に対応している。事象集合〝課目〟も同様である。事象集合を表わすリレーションを事象集合リレーション（entity-set relation or ESR）と呼ぶ。

次に関係性集合〝得意課目〟を考えてみよう。これは事象集合〝生徒〟と〝課目〟内の各々の事象を互いに関連づけている。関係性集合〝得意課目〟は、リレーションスキーム得意課目（番号、課目番号、成績）として表わせる。属性 番号は、ＥＳＲ生徒のキー属性である番号と同じ領域に属している。又、属性 課目番号は、ＥＳＲ課のキー属性 課目番号と同じ領域に属している。例えば、太郎は算数と理科が得意で成績はともにＡであるという関係性は、リレーション得意課目内の第１及び第２番目の組（tuple）として表わせる。これらの組内の属性 番号の値41は、太郎の学籍番号であり、この番号を通してＥＳＲ生徒から〝太郎〟をみつけられる。同様にリレーション得意課目と課目とは、属性 課目番号を介して関係づけられている。ここで関係性集合を表わすリレーションを関係性集合リレーション（relationship-set relation or RSR）と呼ぶことにする。今までみてきた様に、ＲＳＲは、対応する関係性集合が関連づける事象を表わすＥＳＲのキー属性を持っている。

データベース設計を考えてみる。この設計過程は、次の３つのステップから成っている。

1） Ｅ－Ｒ図式の生成

2） 事象集合リレーション（ＥＳＲ）のスキームと外延（extension）の生成

3） 関係性集合リレーション（ＲＳＲ）のスキームをＥＳＲスキームよりつくる。ついで、ＥＳＲの外延をみながらＥ－Ｒ図式内の関係性の意味を満足するＥＳＲのキー属性の集合を関係性集合リレーション（ＲＳＲ）に格納する。

以上のことより、Ｅ－Ｒモデルとリレーショナルモデルとは、自然に対応していることが解かる。即ち、我々はまずＥ－Ｒモデルを設計手段として、対象の意味を概念的に記述するための手段として用い、そしてこのＥ－Ｒモデル記述を、実現する手段としてはリレーショナルモデルを用いている。このことによって、

1） Ｅ－Ｒモデルの持つ事象及び関係性概念の記述能力と

2） リレーショナルモデルの持つデータ記述とアクセスの簡潔性とを生かすことができる。

我々は、局所概念スキーマ（ＬＣＳ）は、Ｅ－Ｒモデルによってまず局所内部スキーマ（ＬＩＳ）から記述される。次に実現はリレーショナルモデルによって行なう。ＬＣＳのリレーショナルモデルによる記述をＬＣＳのリレーショナル記述（relational description or RD）と呼ぶ。即ち、局所概念スキーマ（ＬＣＳ）の設計はＥ－Ｒモデルによってなされ、実際のデータ記述とデータへのアクセスとはリレーショナルモデル、即ちリレーショナル記述（ＲＤ）、によってな

される。

4.3 共通アクセス言語ーＱＵＥＬ

4.2節で述べたように、簡潔性の点からデータへのアクセスは、リレーショナルモデルに基づいて行なわれる。このため共通アクセス言語としては、リレーショナル言語を用いることになる。

リレーショナル言語は大別して、リレーショナル代数言語とリレーショナル計算言語との２つがある。リレーショナル代数は、リレーションに対する代数演算機能、ＪＯＩＮ，PROJECTION，UNION，INTERSECTION 等を提供するものである。ユーザは、これらの演算のシーケンスによって必要なデータを獲得する。この意味で、リレーショナル代数はより手続的である。

一方、リレーショナル計算言語（relational calculus language）は、第１階述語論理に基づいた非手続的言語である。この意味で我々は、リレーショナル計算言語は、リレーショナル代数言語より高位にあると考えている。このタイプの言語として、ＡＬＰＨＡ〔CODDE 70〕、ＱＵＥＬ〔HELDG 75，YOUSK 77〕等の問合せ言語がつくられている。

我々は共通アクセス言語として、リレーショナル計算に基づいたＱＵＥＬを採用した。ＱＵＥＬは、利用が困難な∀、∃といった限定作用素（quantifier）をもたないことが、この大きな理由となっている。ＱＵＥＬは、組（tuple）単位のアクセスを行なう。

4.3.1 ＱＵＥＬの概要

ＱＵＥＬの概要を以降に示す。ＱＵＥＬの形式的な定義は、付記１に示してある。以降、例としては次の３つのリレーションスキームを用いる。

ＥＭＰ（e#，ename，d#）　　ＤＥＰＴ（d#，dname，manager）

ＳＡＬ（e#，sal）

ＥＭＰは従業員の情報を持ち、ＤＥＰＴは部門の情報を持っている。ＳＡＬは従業員の給料の情報を持っている。

ＱＵＥＬは大別して、次の３種の文からなっている。

1）　range 文　（＜range-st＞）

2）　検索文　　（＜retrieve-st＞）

3）　更新文　　（＜update-st＞）

range 文は、問合せで参照するリレーションの組変数（tuple variable）を定義するためのものであり、次の形式をもつ。

range　（x1，x2，…，xn）（X1，X2，…，Xn）；…(1)

X1,…,Xnはリレーション名であり、x1,…,xnは各々X1,…,Xnに対応する組変数である。ここでX1,…,Xnは互いに全て異なっている必要はない。例えば

range (e, d, s) (EMP, DEPT, SAL);

は、リレーションEMP、DEPT、SALに対して、各々組変数e, d, sを定義している。

次の検索文は、検索を行なうためのもので、次のような形式をもっている。

retrieve into < rel-name > (< target-list >)
where < qual > ; …(2)

< rel-name >は、検索の結果生成されるリレーション名である。< target-list > は、結果リレーションの属性（結果属性（result-attribute）と呼ぶ）と、検索の対象となっているリレーションの属性（<v-attribute>）との対応を示している。

< target-list > ::=< t-clause > {, < t-clause > }
< t-clause > ::=< result-att > =< a-exp > | < v-attribute > …(3)
< result-att > ::=< att-name >

< a-exp >は、< v-attribute > から成る算術式〔(6)を参照〕である。結果属性< result-att >と< v-attribute > の属性名が同じならば、結果属性を書く必要はない。リレーションの属性は、問合せ内で< v-attribute > ::=< variable >. < attribute >として表わされる。< variable >は組変数名で、< attribute >は属性名である。例えばリレーションEMPの属性enameは、e.enameとして表わされる。

< qual >は問合せの条件式で、述語論理形式で表わされる。

< qual > ::=< c-pred > | < qual > and < qual > |
< qual > or < qual > | not < qual > |
(< qual >) …(4)

ここで< c-pred >は、比較述語（comparison predicate）を表わしている。

< c-pred > ::=< a-exp > < cop > < a-exp > …(5)
< cop > ::=< | ≦ | = | ≦ | > | ≠

< a-exp > ::=< constant > | < v-attribute > |
< < a-exp >からなる算術式 > |
< < a-exp >についての算術関数 > |
< < a-exp >についてのaggregate関数> …(6)

例えば、「研究部門に属している従業員」という条件は、

d.d#=e.d# and d.dname="研究" と表わせる。

ここで、d．d#＝e．d#のように比較述語の両辺に異なった組変数を持っている場合、この述語式を結合式（join formula）と呼ぶ。一方、d．dname＝"研究"のように、両辺の片方が定数（ここでは"研究"）である場合、これを制限式（restriction formula）と呼ぶ。特に比較演算子が＝の時、各々を等価結合式（equi-join formula）、等価制限式（equi-restriction formula）と呼ぶ。

次にQUELの関数について考える。算術関数は、数値の組に対して1つの数値を結果として出す関数である。例として、LOG、EXP、SIN等がある。

これに対してaggregate関数は、組の集合に対して作用して1つの数値を求めるものである。QUELのaggregate関数としては、AVG、SUM、MAX、MIN、COUNT、ANY、UAVG、USUM、UCOUNTがある。頭にUのついたものは、作用する対象内のユニークな値に対してのみ働くものである。aggregate関数<g-function>は次のような形式を持っている。

<g-function> ::= <g-func-name> (<a-exp> 〔by <v-att>{by <v-att>}〕〔where <qual>〕)

<g-func-name> ::= AVG | SUM | MAX | MIN | COUNT | ANY | UAVG | USUM | UCOUNT

…(7)

例えば、"開発部の職員の給料の平均値"は

AVG（s.sal where s.e#＝e.e# and e.d#＝d.d#
and d.dname ＝"開発"）　となる。

各部ごとの給与合計は

SUM（s.sal by d.d#
where s.e# ＝ e.e# and e.d# ＝ d.d#）　となる。

Byは、この後にある属性の値ごとに組をグループ化するオペレータである。

QUELは、更新機能として追加（append）、削除（delete）、置換（replace）の3つの機能を持っている。

追加は、条件を満足する組の集合をリレーションに加えるもので、次の形式を持つ

append 〔to〕 <rel-name> (<target-list>)
〔where <qual>〕;

<target-list>は、追加される組の属性値を記述している。例えば、"太郎"のもとで働いている従業員として"次郎"（従業員番号151）を加えることは、次のように書ける。

range of (e,d) (EMP, DEPT);

append to EMP (e.e#=151, e.ename="次郎", e.d#)

where e.d#=d.d# and d.manager="太郎";

削除は、条件を満足する組をリレーションから除くもので、次の形式をもつ。

delete <variable> 〔where <qual>〕;

これは<target-list>を持たない。例えば、給料が10万以下の従業員をEMPリレーションから除去することは、次のように書ける。

range of (e, s) (EMP, SAL);

delete e where e.e#=s.e# and s.sal ≤ 10万;

置換は、条件を満足する全ての組の属性の値を、<target-list>に基づいて置き換える。<target-list>は、置換えられる属性の値を定めている。

replace <variable> (<target-list>) 〔where <qual>〕;

例えば、開発部員の給料を全員1割上げることは次のように書ける。

range of (e, s, d) (EMP, SAL, DEPT);

replace s (sal=e.sal*1.1)

where s.e#=c.e# and e.d#=d.d#

and d.dname="開発";

上記した更新文はいずれも1つのコマンドで1つのリレーションに対してのみアクセスできる。

4.3.2 QUELの例

次に、QUELの使用例を考える。前節で用いた3つのリレーションについて考える。

問合せ1：開発部の管理者を求めよ

range of (d) (DEPT);

retrieve into R1 (d.manager) where d.dname="開発";

問合せ2：開発部の管理者の給料を求めよ

range of (e, s) (EMP, SAL);

retrieve into R2 (s.sal)

where s.e#=e.e# and e.d#=d.d# and

e.ename=d.manager and d.dname="開発";

問合せ3：各部ごとに、会社全体の平均給料より多くもらっている社員数をもとめよ。

range of (s1) (SAL);

retrieve into R3 (d.d#,

```
numb = COUNT (e.e#  by  e.d#
                where e.e#=s.e#  and
                s.sal > AVG(s1.sal));
```

4.4 既存データモデルの考察

各サイトのデータベースシステム（DBS）の局所内部スキーマ（L1S）を記述しているデータモデルの相違は、最も主要なDBSの異種性である。本節では、既存データモデルとして3つの代表的データモデル、リレーショナルモデル、DBTGモデル、IMSモデルをとりあげ、これらの一般的構造を解明することを試みる。

4.4.1 データモデルの一般構造 〔TAKIM 79 a〕

既存データモデルにおいて、データのセマンティクスは、データモデル以外の種々の規則（e.g.インテグリティ規則）としてドキュメントの形で表わされている。即ち、リレーショナルモデルを例にとれば、あるリレーションが事象を表わしているのか、事象間の関係性を表わしているのかは、このモデルを見るだけでは解からない。このことを知るためには、モデル以外のドキュメント又はデータベース管理者（DBA）に聞くしかない。上述したデータモデルの性質は、モデルの構造に対する次の様な考察を導き出せる。即ち、データモデルは、モデルの要素と、これらの要素間の構文的関係性との記述法である。

ここで、要素（Eと記す）とは、データモデルの最少の名前づけられる要素である。リレーショナルモデルでは、Eは属性であり、DBTGではデータ項目名である。

ブロック（Bと記す）とは、要素（E）の集合である。例えば、Bはリレーショナルモデルではリレーション名、DBTGではレコード型である。この時、データモデルの構造は、EとBとの明確化と、EとBとの関係性、即ち下記の4つの関係性を示すことによって明らかに出来る。

1) $E_1 \hat{R} E_2$
2) $E \hat{R} B$
3) $B \hat{R} E$
4) $B_1 \hat{R} B_2$

$E_1 \hat{R} E_2$は、要素同志の関係性である。例えば、リレーショナルモデルでは、この関係性として属性間の従属性（dependency）をあげることができる。

$E \hat{R} B$は、要素がブロック内でどのような役割を演じているかを示している。例えば、EはBのキーであるとか、インデクスをもっているかである。

B $\overset{R}{\Rightarrow}$ Eは、Bがどのようなエンティティから構成されているかを示している。

B_1 $\overset{R}{\Rightarrow}$ B_2 は、ブロック間の関係性を表わしている。このブロック間の関係性は、多くのデータモデルを特徴づけるものである。例えば、DBTGモデルにおけるレコード型間のセット型、IMSモデルにおける階層パスがそれである。この2つのモデルでは、この関係性は、物理的なアクセスパスに対応している。

4.4.2 リレーショナルモデル

ここでは、リレーショナルモデルの基本構造を、4.4.1項に基づいて、

1) リレーショナルモデルの基本構成要素

2) これらの基本要素間の関係性　　との2点について検討する。

A. リレーショナルモデルの基本要素

リレーショナルモデルの基本構成要素は、領域（domain）、属性（attribute）、リレーション（relation）の3つである。すべての属性は、ある領域内の値のみをとり得る。リレーションスキームは、属性の集合から構成され、リレーションはこれらの属性の属する領域の直積の部分集合である。このように、属性はオカーランス（O）であり、属性は基本要素（E）であり、リレーションはEの集合としてのブロックである。これらを図4.5にまとめる。以降属性をAと記し、領域をDと記し、リレーションをRと記すことにする。

要素	リレーショナルモデル
O	領　域 (D)
E	属　性 (A)
B	リレーション (R)

図4.5 リレーショナルモデルの基本要素

B. 基本要素間の関係性

リレーショナルモデルにおける関係性としては、次の4種がある。

1) 属性→属性　　(A_1 $\overset{R}{\Rightarrow}$ A_2)

2) 属性→リレーション　　(A $\overset{R}{\Rightarrow}$ R)

3) リレーション→属性　　(R $\overset{R}{\Rightarrow}$ A)

4) リレーション→リレーション　　(R_1 $\overset{R}{\Rightarrow}$ R_2)

1) 属性間の関係性 (A_1 $\overset{R}{\Rightarrow}$ A_2)

リレーショナルモデルにおける属性間の関係性（R）としては、関数従属性（functional dependency）〔CODDE 70〕がある。A_1 $\overset{R}{\Rightarrow}$ A_2 は、属性A_1は属性A_2に関数従属していることを表わしている。即ち、$A_2 \rightarrow A_1$。これは属性A_2の取り得るどの値

も、正確にただ1つのA_1のとり得る値と結びついていることを示している。

関数従属性は、1つのリレーション内で閉じているものであり、正規化されたリレーション内の全ての属性は、主キー属性にのみ関数従属している。この主キー属性は、他のモデルにおけるキーの概念と等価なものと考えられる。実際に、リレーション内の組を一意に識別するために、多くのリレーショナルデータベースシステム〔ASTRM 76, STONM 76〕は、組識別子（tuple identifier or TID）を内部的に持っている。即ち、主キー属性のどの値も一意に対応するTIDをもっていると言える。このように$A_1 \overset{\Rightarrow}{R} A_2$は、後述する。$A \overset{\Rightarrow}{R} R$の関係性と同等である。

ある属性の1つの値が、他の属性の値の集合を決定する関係性として多値従属性（multi-valued dependency）〔FAGIR 77〕がある。しかし、我々は、この従属性は、極めてセマンティカルなレベルのものであり、リレーショナルモデルへは取り入れるべきではないと考えている。数学的に、関数とは、領域内の1つの値を、値域（range）内の1つの値へマッピングするものであり、この逆関数は、値域内の1つの値を領域内のnコの値（$n \geq 0$）へマッピングする。このことは関数従属性の逆は、一般に多値従属性となり得ることを示している。このように、属性間の多値従属性は、数学的に種々あり得る。こうした多くの多値従属性から意味のあるものをみつけるのは、全くセマンティクスの問題である。又、属性間に多値従属性が存在する時、リレーションの実体のレベルで何を意味するのだろうか。実体レベルで、2つの属性が多値従属であることを保障するのは極めて多くの処理を必要としてしまう。

テーブル表現は、ある識別子がある組をユニークに識別し得ることを表わしている。従って1：n、n：mといった関係性を表わすためには、リレーショナルモデルの様なテーブル表現は適していない。このような関係性を表わすためには、関係従属モデル（functional dependency model or FDM）〔HOUSB 79〕のような2項関係性モデルが適していると考える。

2） 属性のリレーションに対する関係性（$A \overset{\Rightarrow}{R} R$）

これは、ある属性Aがリレーション R に対してどのような役割を演じているかを示している。役割としては、AはRの主キーであるが、非キーであるかである。

3） リレーションの属性に対する関係性（$R \overset{\Rightarrow}{R} A$）

これは、リレーション R がどのような属性から構成されているかを示している。

4） リレーション間の関係性（$R_1 \overset{\Rightarrow}{R} R_2$）

2つのリレーション R_1 と R_2 の間の関係性は、R_1 と R_2 が同じ領域を共有することによって生じる。共有される領域に属する属性が各々のリレーション内で演じる役割によって次

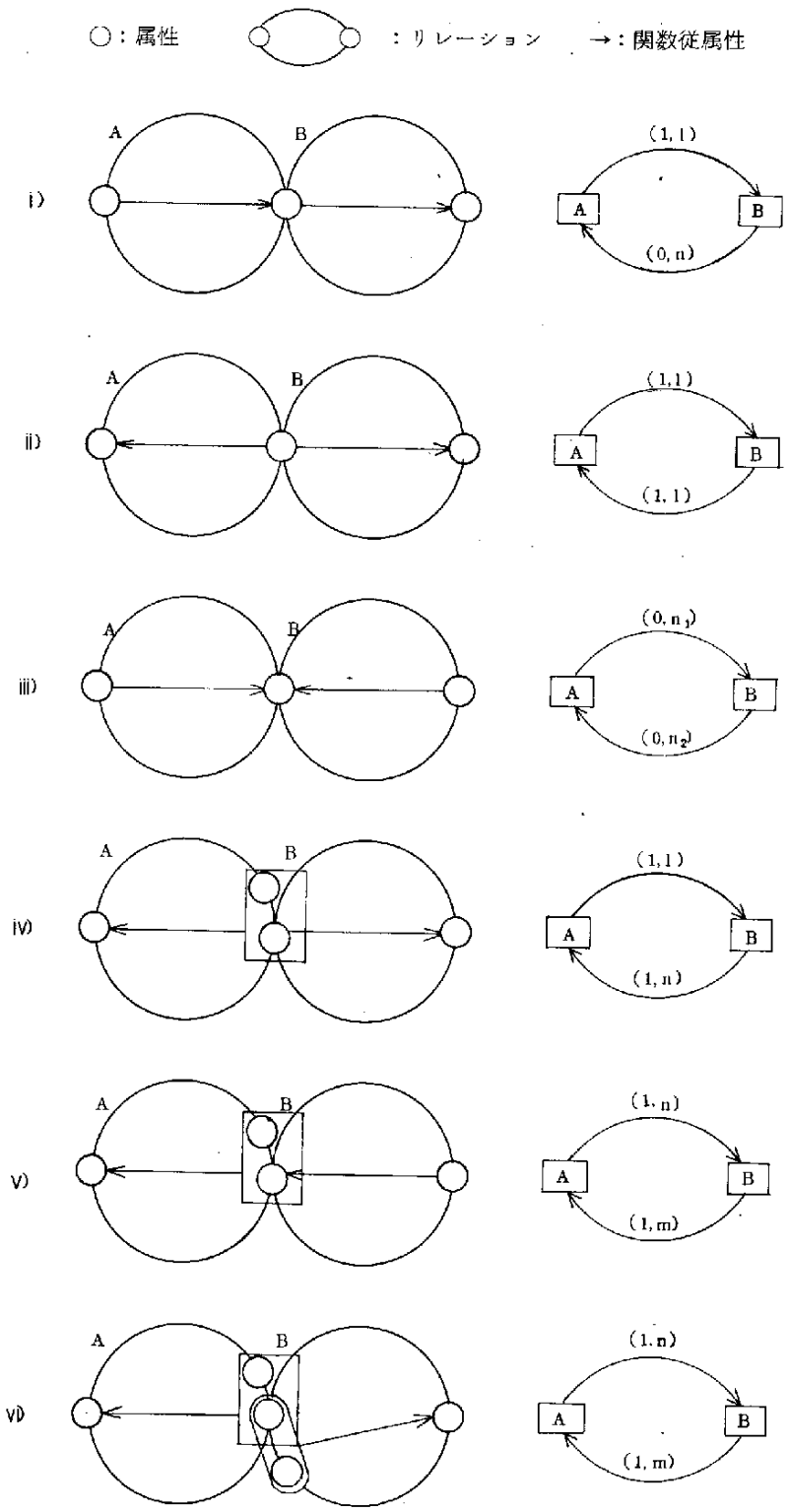


図 4.6　リレーション間の関係性

のような関係性のパターンがある。

i） 主キーと非キー　　（外来キー）

ii） 主キーと主キー　　（共有キー）

iii） 非キーと非キー　　（共有属性）

iv） 複合キーと非キー

v） 複合キーと主キー

vi） 複合キーと複合キー

これらの関係性を図4.6に図示する。図4.6内には、２つのリレーションＡとＢとの間の組の対応関係をデータセマンティクスモデル〔ＡＢＲＩＪ　７４〕によって記述してある。

4.4.3　ＤＢＴＧモデル

ここで検討するＤＢＴＧモデルは、〔ＤＡＴＥＣ　７７〕に準拠したものである。

Ａ．ＤＢＴＧモデルの基本要素

モデル要素	ＤＢＴＧモデル
O	項目のオカーランス
E	項目名　(I)
B	レコード型　(R)

図4.7　DBTGモデルの基本要素

図4.7に、ＤＢＴＧモデルの基本構成要素をまとめてある。ここでデータ項目名をＩ．レコード型をＲによって示すことにする。

Ｂ．基本要素間の関係性

ＤＢＴＧモデルの関係性として、項目名間（$I_1 \hat{R} I_2$）は、モデル内に陽に存在していないので論じない。よって、次の３つについて考える。

1）　項目名→レコード型　　（$I \hat{R} R$）

2）　レコード型→項目名　　（$R \hat{R} I$）

3）　レコード型→レコード型　　（$R_1 \hat{R} R_2$）

1）　項目名のレコード型に対する関係性（$I \hat{R} R$）

これは、ある項目名がレコード型内でどのような役割を演じているかを示している。ＤＢＴＧモデルでは、ＩがＲのキーであるか、非キーであるかである。ＤＢＴＧのキーとしては次のような種類がある。

ⅰ） ＣＡＬＣ－key 項目

ⅱ） ＳＯＲＴ－key 項目

ⅲ） ＳＥＡＲＣＨ－key項目

レコード型はＬＯＣＡＴＩＯＮ（配置）句の定義によって次の５種類のモードを持てる。

a） ＳＹＳＴＥＭ　作製者による

b） ＤＩＲＥＣＴ　ＤＩＲＥＣＴ句で指定した項目の内容

c） ＣＡＬＣ　指定した項目の内容の乱数化

d） ＶＩＡ　セット実現値の子レコード

ここではＣＡＬＣ方式のみを考える。ＣＡＬＣ項目は、レコード型のどの項目に対してもどのような組合せも可能であるが、我々は簡単にするためにレコード型の１つの項目のみと仮定した。さらにＤＵＰＬＩＣＡＴＥ　ＩＳ　ＮＯＴ　ＡＬＬＯＷＥＤ（ＤＮＡ）句で修飾されているとする。即ち、ＣＡＬＣ項目がある値をもつオカーランスはユニークである。

ＤＢＴＧモデルではセット実現値において、子レコードの並ぶ順序を定めることができる。この時順序づけに用いられる項目がＳＯＲＴ－key　項目である。

セット型の実現値ごとに、その全ての子レコードを探索できるように、ＣＡＬＣの他にＩＮＤＥＸ　ＮＡＭＥで任意個のデータ項目が指定できる。

2）　レコード型の項目名に対する関係性（$R \overset{A}{R} I$）

これは、レコード型がどのような項目から構成されているかを示している。

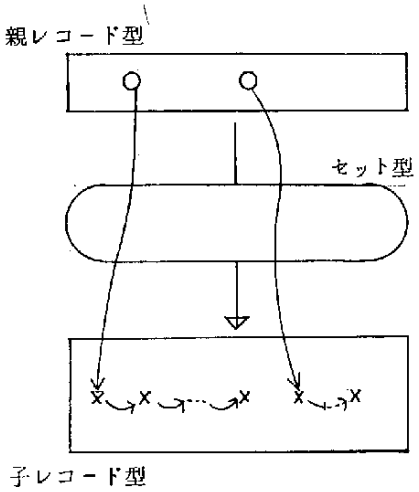
3）　レコード型間の関係性（$R_1 \overset{A}{R} R_2$）

これはレコード型R_1とレコード型R_2との関係性を示しており、ＤＢＴＧモデルはセット型（set－type）によって表わされる。セット型はＤＢＴＧモデルを特徴づける最も主要な要素である。

セット型は、１つの親レコード型（owner　record－type）と１つの子レコード型（member　record－type）との間の関係性である。このセット型の存在は、１の親レコード型のオカーランスは、セット型を介して、子レコード型のｎコのオカーランス（$n \geq 0$）とリンクされていることを示している〔図4.8〕。即ち２つのレコード型間の１：ｎの関係性の示している。

ＤＢＴＧモデルでは、図4.9に示すように４つの基本構造から成っている。階層構造は、今述べた２つのレコード型間の１：ｎの関係性を表わしている。ネットワーク型構造は、２つのレコード型の間のｎ：ｍの関係性を表わすために、いわゆるリンクレコード型（Ｒ）が導入されている。

即ち、レコード型Ａの１つのオカーランスは、セット型S_1を介してＲのｎコのオカーランス



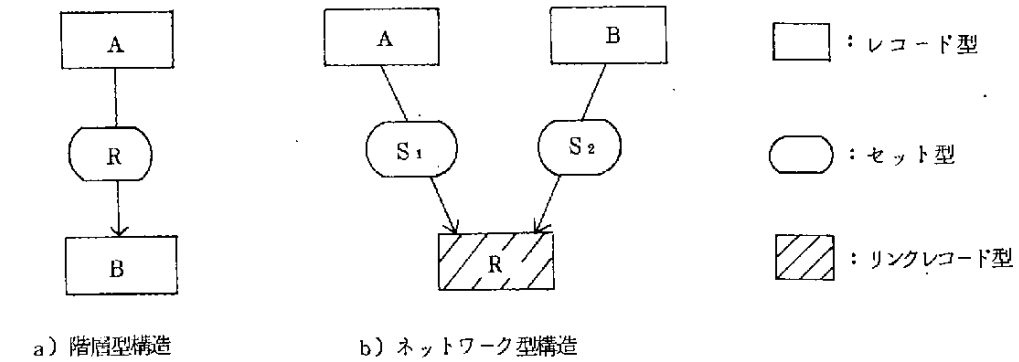
○：親レコード型のオカーランス

×：子レコード型のオカーランス

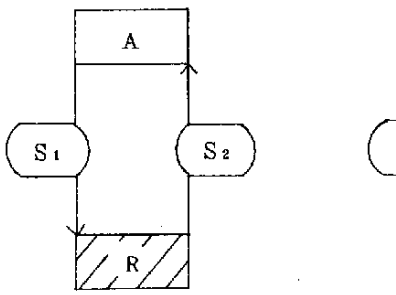
図4.8　セット型

とリンクされている。Rの各オカーランスはS_2を介してBの1つのオカーランスとリンクされている。逆にBの各オカーランスは、S_2、R、S_1を介してAのmコのオカーランスとリンクされている。よって、AとBとの間のn：mの関係性をネットワーク型構造は実現している。

次の階層型サイクル構造と、ネットワーク型サイクル構造とは、1つのレコード型内の関係性（サイクル構造）を示している。階層型サイクル構造は、企業の人事情報のような階層的関係を表わしている。例えば企業の従業員をオカーランスとするレコード型Eがあったとする。ある社員の部下はS_1，R，S_2によってn_1人の部下と



a）階層型構造　　b）ネットワーク型構造

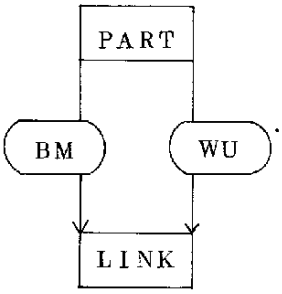


c）階層型サイクル構造　　d）ネットワーク型サイクル構造

図4.9　DBTGモデルの基本構造

リンクされ、各部下は又同じくn₂ 人の部下とリンクされている。しかし、ある社員は自分よりも上役をけっして部下と出来ない。このような同一レコード型内のオカーランス間の階層関係を示している。

ネットワーク型サイクル構造は、同一レコード型内のオカーランス間のネットワーク的（n：m）関係性を表わしている。この例として、工場製品の部品展開が良い（図4.10）。即ちある部品は、いくつかの部品から構成されるとともに、他のいくつかの部品となる。PART内のオカーランス（部品）は、セット型BM（LINK→WU）を介して、この部品から構成されるいくつかの部品をもとめられる。一方WU（LINK→BM）を介しては、この部品を構成するいくつかの部品をもとめられる。



ここで BM＝bill of materials
WU＝where used

図4.10　部品展開

セット型のメンバーシップクラスは、AUTOMATICでかつMANDATORYとする。即ち、図4.11におけるセット型Sの子レコード型Bの全てのオカーランスは、セット型Sを介して親レコード型Aのオカーランスとリンクされている。また、セット型Sは、子レコード型Bのある項目についてDUPLICATE IS NOT ALLOWED（DNA）とする。これは、セット型Sのあるオカーランス内の子レコード型Bのオカーランスを考えると、これらは全てDNAと宣言された項目の値をユニークに持っている。この時我々はこのレコード項目をキーと呼ぶ。

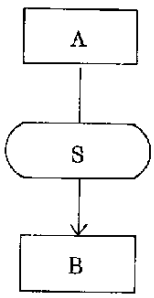


図　4.11

4.4.4 ＩＭＳモデル

ここでは、ＩＭＳのモデル〔ＤＡＴＥＣ　７７〕を検討する。

Ａ．ＩＭＳモデルの基本要素

モデル要素	ＩＭＳモデル
O	フィールドのオカーランス
E	フィールド名 (F)
B	セグメント型 (S)

図4.12　ＩＭＳモデルの基本要素

ＩＭＳモデルの基本要素を図４.１２にまとめる。ここで、フィールド名をＦ，セグメント型をＳによって表わすことにする。

Ｂ．基本要素間の関係性

ＩＭＳの基本要素間の関係性は、ＤＢＴＧモデルの場合と同様に次の３種である。

１）　フィールド名→セグメント型　（F $\overset{R}{\Uparrow}$ S ）

２）　セグメント型→フィールド名　（S $\overset{R}{\Uparrow}$ F ）

３）　セグメント型→セグメント型　（S_1 $\overset{R}{\Uparrow}$ S_2 ）

１）　フィールド名のセグメント型に対する関係性（F $\overset{R}{\Uparrow}$ S）

これは、あるフィールド名Ｆがセグメント型Ｓ内でどのような役目、即ち、キーであるか、非キーであるかを示している。

２）　セグメント型のフィールド名に対する関係性（S $\overset{R}{\Uparrow}$ F）

これは、セグメント型Ｓがどのようなフィールド名から構成されているかを示している。

３）　セグメント型間の関係性（S_1 $\overset{R}{\Uparrow}$ S_2 ）

このセグメント間の関係性は、階層パス（hierarchical　path ）によって表わされる。階層パスは２つのセグメント型間の階層関係を表わし、１つの親セグメント型（parent segment－type）と最低１つの子セグメント型（child　segment－type）を持っている。親セグメント型の１つのオカーランスは、階層パスを介してその子セグメント型のｎコのオカーランスとリンクされている。即ち、階層パスは、親と子セグメント型間の１：ｎの関係性を示している。ＤＢＴＧモデルのセット型との差は、親に対してｎコの子セグメント型がある時、その順序がアクセス上大きな意味を持っていることである。

4.5　同種化と異種性情報（ＨＩ）〔ＴＡＫＩＭ７９ａ〕

異種性の主要要素としてデータモデルと言語とがあると我々は仮定した。前節では、同種化すべき既存データモデルとしてリレーショナル、ＤＢＴＧ、ＩＭＳとの３つのデータモデルをとり上げその一般的構造を明らかにした。さらに4.2節では、共通データモデルとしてのＥ－Ｒモデルについて検討を行ない、5.3節では共通アクセス言語としてのＱＵＥＬを検討した。本節では、このような検討を基にして、１）既存データモデルによって記述された局所内部スキーマ（ＬＩＳ）から、Ｅ－Ｒモデルによって共通記述された局所概念スキーマ（ＬＣＳ）への同種化（homogenization）と、２）この過程で除去されるＬＩＳとＬＣＳとの対応情報（我々はこれを異種性情報（heterogeneity information or HI）と呼ぶ）とについて論じる。

4.5.1　同種化と異種性情報との概要

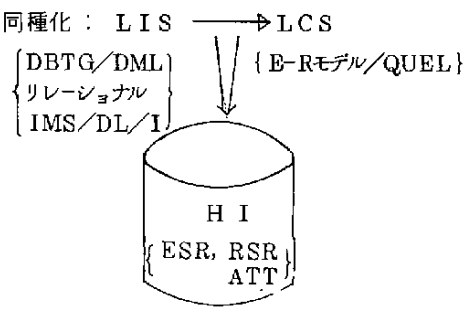
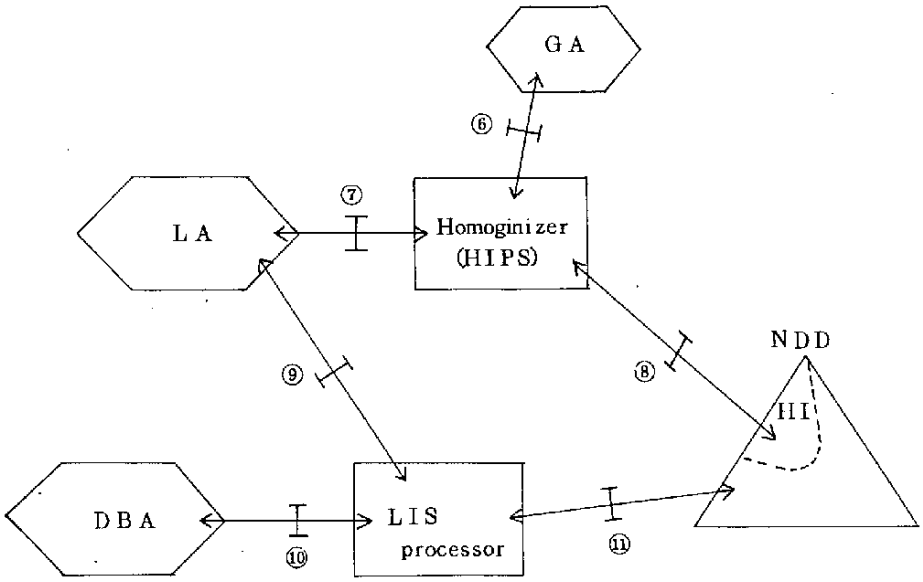


図4.13　同種化とＨＩ

同種化と異種性情報（ＨＩ）との概要を図4.13に示す。これより詳細な関係の記述は図4.14に示す。

同種化は、表4.1に基づいて、既存データモデルの各要素を対応するＥ－Ｒモデルの要素へ変換することによってなされる。この過程はほとんど自動化できるが、若干の部分は人手でやらねばならない。その部分は、例えばＤＢＴＧモデルでは、レコード型が通常のものか、リンクレコード型かの区別はＤＢＴＧのＬＩＳスキーマからは出来ない。



LA：local administrator
GA：global administrator
DBA：database administrator
NDD：networkdata directory
HI：heterogeneity information
HIPS：HI processing system

図4.14　同種化

表 4.1　モデルの対応

E−Rモデル / LISモデル	値集合	属性	事象集合	関係性集合
リレーショナルモデル	領域	属性	リレーション名	外来キー　他
DBTGモデル	項目のオカーランス	項目名	レコード型	セット型 リンクレコード型
IMSモデル	フィールドのオカーランス	フィールド名	セグメント型	階層パス

図4.14の同種化システム（homogenizer ）は、4.7で述べるプログラム、ＨＩＰＳ〔詳細は8.1節に述べられてある〕によって実現されている。局所管理者（ＬＡ）は、そのサイトのデータベースシステム（ＤＢＳ）のスキーマ、即ち局所内部スキーマ（ＬＩＳ）をみて（⑨）ＨＩＰＳの入力形式（⑦）へ変換する。ＨＩＰＳは、ＬＩＳをＥ－Ｒモデルで記述された局所概念スキーマ（ＬＣＳ）へ変換し、異種性情報（ＨＩ）として出力する。ＬＩＳを使うユーザは⑥のインターフェースを通してＬＣＳのＲＤ（relational description）をみることができる。後述する統合化〔第６章〕では全体管理者（ＧＡ）は、このインタフェースを介して各データベースシステムの局所概念スキーマ（ＬＣＳ）を視ながら、全体概念スキーマ（ＧＣＳ）を記述する。

異種性情報（ＨＩ）は、大別して、次の２つの情報から成っている。

１）　ＬＣＳとＬＩＳとの対応情報

２）　ＬＩＳに基づいたデータアクセス言語（これをＬＩＳ　ＤＭＬと呼ぶ）

１）は、さらに、ＬＣＳ要素とＬＩＳ要素との対応情報に加えて、各ＬＩＳ要素についてのカーディナリティ、選択度といったパフォーマンス情報を保持している。２）については図5.34を参照されたい。

異種性情報（ＨＩ）は、局所内部スキーマ（ＬＩＳ）のモデル要素と局所概念スキーマ（ＬＣＳ）のモデル要素との対応情報を格納している。これらの情報はリレーショナル形式で管理されている。各モデルごとに若干の相違はあるが、ＥＳＲ、ＲＳＲ、ＡＴＴという３つのリレーションによってＨＩを表わしている。ＥＳＲリレーションは、事象集合名、それのサイズ、属性数、組の長さ等の情報を保持している。ＡＴＴリレーションは、事象集合又は関係性集合を構成する属性について、その名前、属する事象集合又は関係性集合名、選択度、キーか非キーか等の情報を保持している。

ＲＳＲリレーションは、関係性集合についての情報を保持している。ＲＳＲリレーションは、各モデルの特徴を最も反映している。これは、原則的に、関係性集合名、これが対応づける事象集合、及び各モデル固有の物理的アクセスパス情報（セット型、etc.）を持っている。

この３つのリレーションのスキームを図4.15に示す。

ESR (entity-set-name, area-root-name, number-of-keys, mode, degree, width, size, protection, integrity, cardinality)

属　性　名	説　明
entity-set-name	事象集合名　(i. e. レコード型)
area-root-name	DBTGモデルのエリア名、IMSのルートセグメント名、
number-of-keys	キー属性の数
mode	アクセスモード(e. g. location mode)
degree	属性数
＊ width	組のサイズ
＊ size	width × cardinality
protection	プロテクションフラグ
integrity	インテグリティフラグ
＊ cardinality	組　数　(オカーランス数)

＊:　パフォーマンス情報

図4.15　異種性情報（HI）スキーマ(1)

RSR (relationship-set-name, s-es-name, d-es-name, mode,
rs-type, degree, width, size,
source-set-name, dest-set-name, protection, integrity,
cardinality)

属　性　名	説　明
relationship-set-name	関係性集合名（e.g. セット型、リンクレコード型
s-es-name	sourceのES名 （e.g. 親レコード型）
d-es-name	destinationのES名 （e.g. 子レコード型）
mode	アクセスモード
rs-type	階層型 (H)、ネットワーク型 (N)、階層サイクル型 (HC)、ネットワークサイクル型 (NC)
degree	この関係性集合が独自でもつ属性の数
＊width	組のサイズ
source-set-name	S－ESとこのRSとの間のセット型
dest-set-name	d－ESとこのRSとの間のセット型
protection	プロテクションフラグ
integrity	インテグリティフラグ
＊cardinality	組　数
＊size	サイズ (width × cardinality)

＊ ： パフォーマンス情報

図4.15 HIスキーマ (2)

ＡＴＴ（es-rs-name, es-rs-type, attribute-no, attribute-name,
value-set , role-of-att,
type , length , protection, integrity,
cardinality , selectivity ）

属　性　名	説　　明
es-rs-name	ＥＳ又はＲＳ名
es-rs-type	ＥＳ　or　ＲＳ
attribute-no	属性番号
attribute-name	属性名
value-set	値集合名
role-of-att	属性の役目（キーか非キー）
＊ cardinality	属性かリレーション（ＥＳ　or　ＲＳ）内でとりうるユニークな値数
＊ selectivity	選択度
type	型　（文字又は整数）
＊ length	バイト長
protection	プロテクションフラグ
integrity	インテグリティフラグ

＊：　パフォーマンス情報

図4.15　ＨＩスキーマ　(3)

4.5.2　リレーショナルモデルＬＩＳからＬＣＳへ

リレーショナルモデルにおいて、リレーション間の関係性は、共有された属性の領域を通してのみ表わされる。即ち、リレーショナルモデルでは、他のモデル〔4.4.3及び4.4.4を参照のこと〕と異なり、データモデル要素として関係性を表わす構造をもたない。よって、問題は、この関係性をどう表わすかである。このことは又、リレーショナルモデルにおけるリレーションとは何かという問題を提起する。

このため、我々はリレーショナルモデルに対して次のような仮定をもうけた。即ち、リレーションは、事象集合を表わすもの、即ち事象集合リレーション（ＥＳＲ）、とこれらの関係性集合を表わすもの、即ち関係性集合リレーション（ＲＳＲ）との２種類に分類できる。このことは、リレーションは、全て第４正規化されていることを意味している。このことは又、リレーショナルモデルの局所内部スキーマ（ＬＩＳ）が、局所概念スキーマ（ＬＣＳ）のリレーション記述そのものであることを意味している。

4.5.3　ＤＢＴＧモデルＬＩＳからＬＣＳ

ここでは、ＤＢＴＧモデルで記述されたＬＩＳからＬＣＳへの同種化と異種性情報（ＨＩ）について論じる。

Ｅ－Ｒモデルの属性、事象集合には、各々ＤＢＴＧモデルの項目名、レコード型が対応する。関係性集合とＤＢＴＧモデル要素との対応は図4.２８に示してあるように、関係性集合にはセット型又はリンクレコード型が対応している。

局所概念スキーマ（ＬＣＳ）は、図4.２８のようにＥ－Ｒモデル記述へ変換することによってなされる。Ｅ－Ｒモデルは、実際にはリレーショナルモデルによって記述される。我々は、これをＬＣＳのリレーショナル記述（relational description or RD）と呼ぶ。

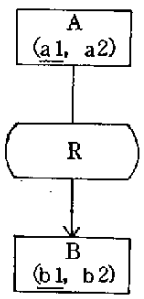


図4.16　階層型構造

ＤＢＴＧモデルは、階層、ネットワーク、階層サイクル、ネットワークサイクルの４つの構造からなっている。以降これらの４つの構造についてＬＣＳへの変換について考えてみよう。

階層型構造を考えてみよう。図4.１６は、レコード型Ａがセット型Ｒの親レコード型で、レコード型Ｂは子レコード型であることを示している。レコード型ＡとＢは、各々項目名集合 a_1 、a_2 と項目名集合 b_1 b_2 とから構成されている。さらに a_1 と b_1 とは、各々レコード型Ａ

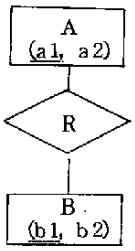


図4.17　階層型構造のE－Rモデル表現

ESR　A　(a_1 , a_2)

　　B　(b_1 , b_2)

RSR　R　(a_1 , b_1)

図4.18　図4.16 のRD

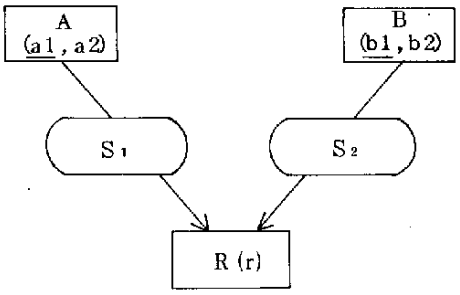


図4.19　ネットワーク型構造

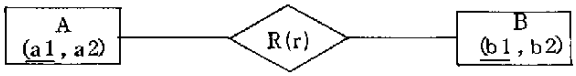


図4.20 図4.19 に対応するE－Rモデル表現

とBのキー（例えばCALC項目名）であることを示している。この構造のE－Rモデルによる表現は図4.17の様になる。図4.17のE－Rモデル表現から生成されるRDを図4.18に示す。事象集合Aは、事象集合リレーション（ESR）スキームA（a_1 , a_2 ）へ変換される。a_1 はESR Aの主キー属性集合、a_2 はキーでない属性の集合である。ESR Bも同様である。セット型Rは、関係性集合Rとなる。

関係性集合Rに対応する関係性集合リレーション（RSR）のスキームはR（a_1 , b_1 ）となる。a_1 とb_1 は、関係性集合Rが関係づけるESR AとBとの各々の主キー属性である。RSR内の属性の順番は重要である。Rの第1番目の属性がAの主キー属性であり、第2番目がBの主キー属性である。

次に、図4.19に示す様なネットワーク型構造について検討しよう。レコード型Rは、いわゆるリンクレコード型である。即ち、レコード型AとBとの間のn：mの関係性を表わしている。ネットワーク型構造に対応するE－Rモデル表現を図4.20に示す。図4.21は、これのRDを示している。レコードRは固有の項目名集合rを持っているために対応するRSRのスキームはR（a_1, b_1, r）となる。

DBTG表現内に現われるセット型S_1 とS_2 はLCSのE－Rモデル表現とRDとのなかに、現われない。このようにLIS内には存在するがLCS内に存在しない要素は異種性情報（HI）内に保持される。

ESR　A　(a_1 , a_2)

　　B　(b_1 , b_2)

RSR　R　(a_1 , b_1 , r)

図4.21 図4.19のRD

次に階層型サイクル構造について考えてみよう。これは、レコード型A内の1つのオカーランスがセット型S_1 を介してRの1つのオカーランスとリンクされ、さらにこのRオカーランスはセット型S_2 を介してn個のオカーランスとリンクされている。即ち、レコードA内のオカーランスが木構造的にリンクされていることを示している。これに対応するE－Rモデル表現を図4.23に、RDを図4.24に示す。V_1 は、ESR　Aの主キー属性a_1 の領域名である。RSR　Rの属性r_1 とr_2 は同一の領域V_1 に属している。ここでも属性r_1 とr_2 のR-SR　R内でのポジションは重要である。即ち、ESR　AとRとで、属性a_1 と属性r_1 との結合（join）は、DBTG上のAからセット型S_1を介し、レコード型R、そしてセット型S_2 を介してAをアクセスすることと等価である。a_1 とr_2 との結合は、この逆を意味している。ネットワーク型構造と同様に、LISに存在するセット型S_1 とS_2 はLCS上には現われない。この情報は異種性情報（HI）として保持される。

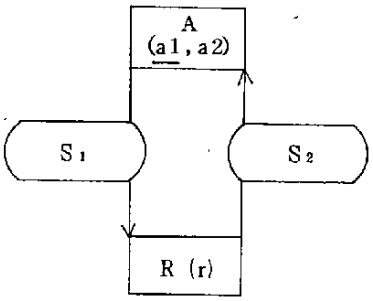


図4.22 階層型サイクル構造

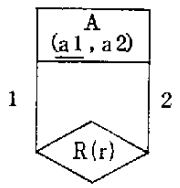


図4.23 図4.22のE－Rモデル表現

ESR　A (a1／V1 , a2)

RSR　R (r1／V1 , r2／V1 , r)

図4.24 図4.22 のRD

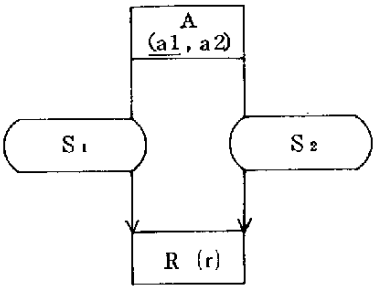


図4.25 ネットワーク型サイクル構造

ESR A (a1／V1, a2)
RSR R (r1／V1, r2／V2, r)

図4.27 図4.27 のRD

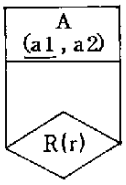


図4.26 図4.25に対応する
E－Rモデル表現

最後にネットワーク型サイクル構造について考えよう。これはレコード型A内のオカーランスがネットワーク的にリンクされていることを表わしている。ここでも RSR R内の属性 r_1 と r_2 のポジションは、階層型サイクル構造の場合と同様の意味を持っている。

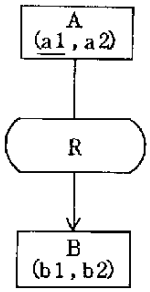
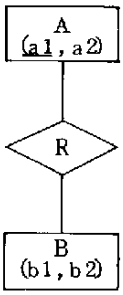
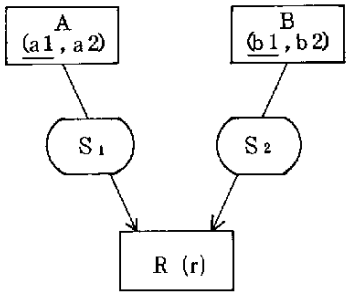
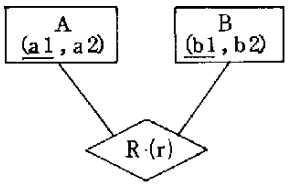
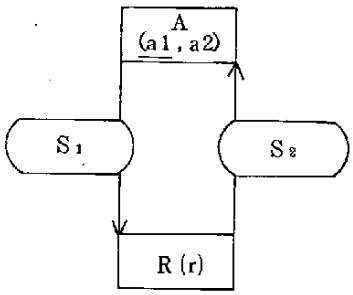
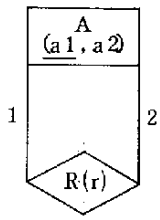
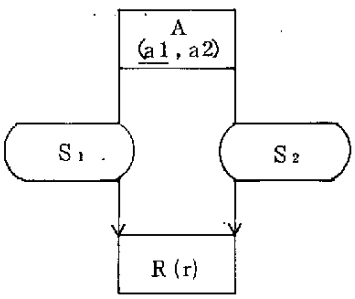
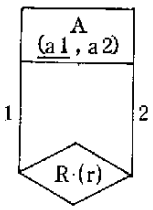
	DBTG 図式	E－R 表現	RD
階層型構造			ESR A(a1, a2) B(b1, b2) RSR R(a1, b1)
ネットワーク型構造			ESR A(a1, a2) B(b1, b2) RSR R(a1, b1, r)
階層型サイクル構造			ESR A(a1/v1, a2) RSR R(r1/v1, r2/v1, r)
ネットワーク型サイクル構造			ESR A(a1/v1, a2) RSR R(r1/v1, r2/v1, r)

図 4.28　DBTG図式とE－R表現、RDとの対応

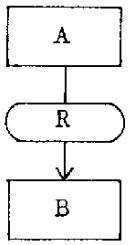
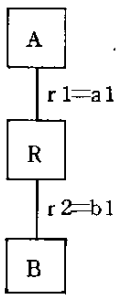
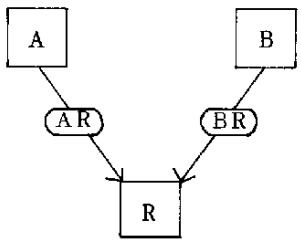
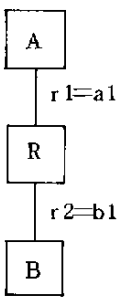
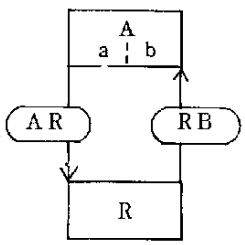
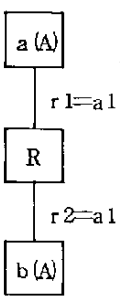
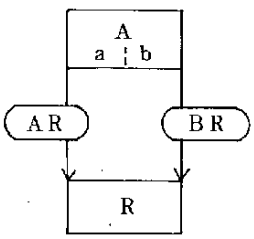
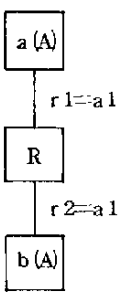
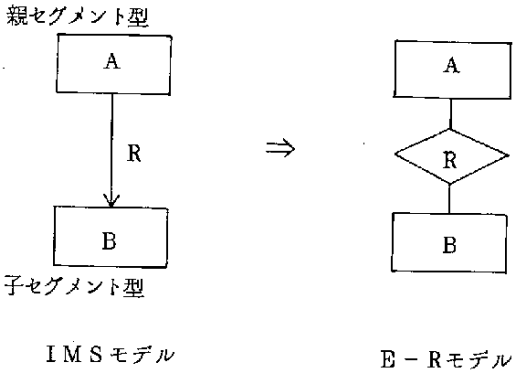
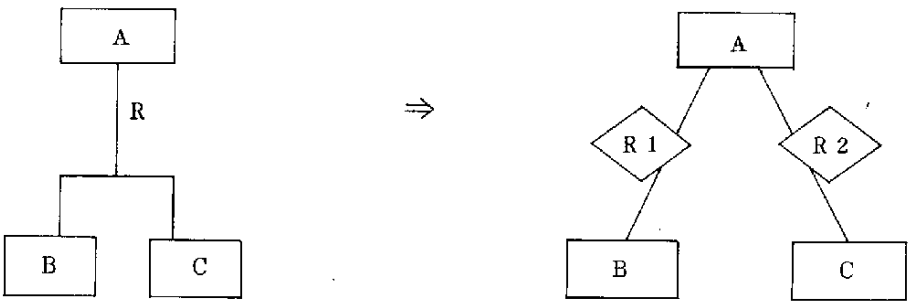
DBTG図式	問合せ図	RSR				
		s-es	d-es	s-set	d-set	rs-type
		A	B			H
		A	B	AR	BR	N
		A	A	AR	RB	HC
		A	A	AR	BR	NC

図4－29　DBTG図式とHI／RSRリレーション

4.5.4　ＩＭＳモデルＬＩＳからＬＣＳ

ＩＭＳモデルＬＩＳからＥ－ＲモデルＬＣＳへの変換は、表4.1に基づいて行なわれる。ＩＭＳモデルのフィールド名はＥ－Ｒモデルの属性へ、セグメント型は、事象集合へ、階層パスは関係性集合へ１対１に対応させることによって変換される。





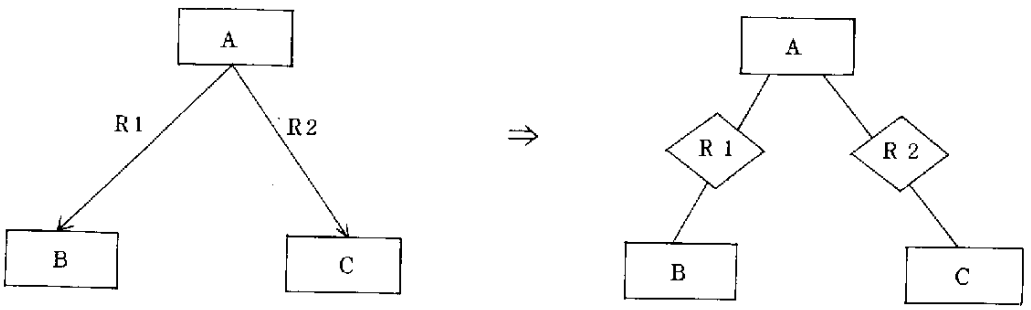


図4.30　ＩＭＳモデルからＥ－Ｒモデルへ

4.6 同種化の例

本節では、同種化の例を次の3種のデータベースシステム（ＤＢＳ）について示す。

1） プロジェクト管理データベースシステム （ＰＭＤＢＳ）

2） プロジェクトデータベースシステム （ＰＲＤＢＳ）

3） 文献データベースシステム （ＢＬＤＢＳ）

ＰＭＤＢＳは、各研究所に存在して、このプロジェクト内の人事及び発表論文の管理を行なっているリレーショナルＤＢＳである。ＰＲＤＢＳは、クリアリングセンタにあり全国のプロジェクトがどの様な研究を行なっているかの情報を保有しており、ＤＢＴＧ型である。ＢＬＤＢＳは、文献情報を管理しているデータベースシステム（ＤＢＳ）である。これらのＤＢＳは、ＤＤＢＳの構成要素でありネットワークによって結合されている。以降これらの3種のＤＢＳの各々について、その概要を示すとともに、同種化によって生成されるＬＣＳ及び異種性情報（ＨＩ）を示す。最後に、これら3種のＤＢＳの関係を示す。

4.6.1 プロジェクト管理データベースシステム（ＰＭＤＢＳ）

ＰＭＤＢＳのＬＩＳはリレーショナルモデルに基づいており、これを図4.31に示す。ＬＩＳ内の属性の関係を示したリレーショナル図式を図4.32に示す。

ＰＭＤＢＳは、各研究所にあり、その研究所内のプロジェクトとそのメンバー、及び出版物（論文）についての情報を保持している。ここで各プロジェクトは1つの研究所内で閉じているものとする。5つのリレーションのなかで、PROJ—MEM, MEM—REPは関係性集合リレーションであり、残りの3つは事象集合リレーションであるとする。するともとめるＬＣＳのＥ－Ｒ図式は、図4.33のようになり、ＬＣＳのＲＤは図4.34のようになる。

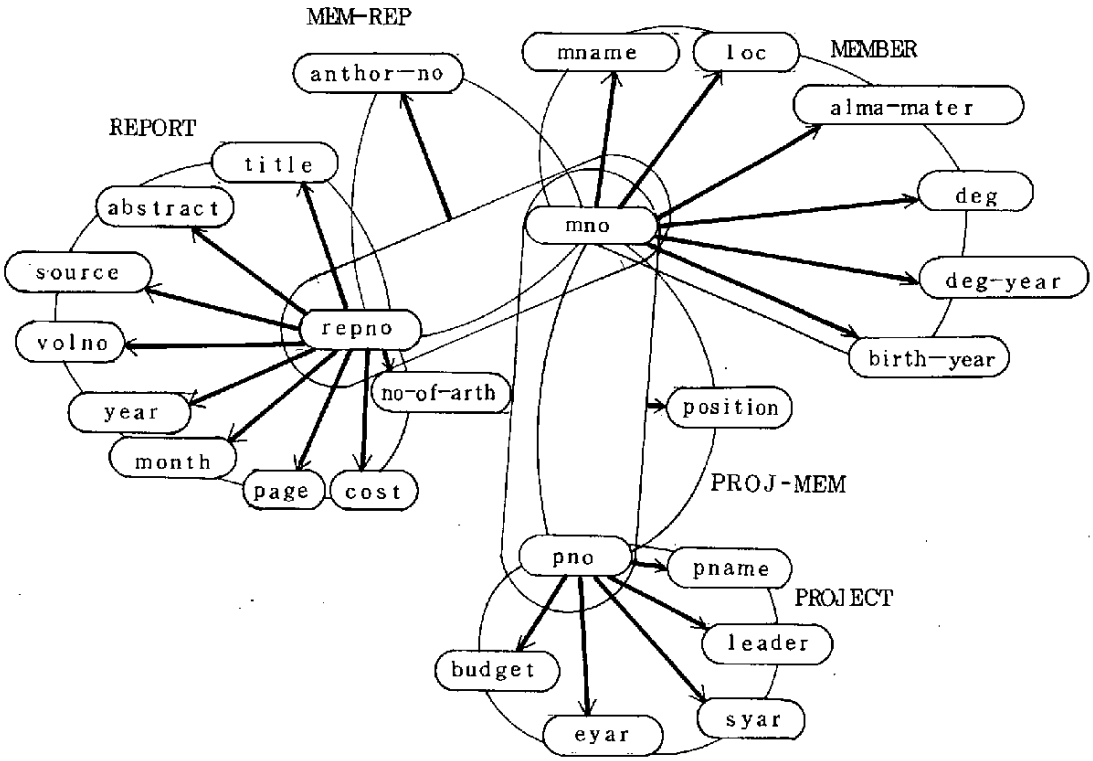


図4.31　PMDBSのリレーショナル図式

MEMBER (mno, mname, loc, alma-mater, deg, deg-year, birth-year)

PROJECT (pno, pname, leader, syar, eyar, budget)

REPORT (repno, title, abstract, source, volno, year, month, page, cost, no-of-auth)

PROJ-MEM (pno, mno, position)

MEM-REP (mno, repno, author-no)

図4.32　PMDBSのスキーマ

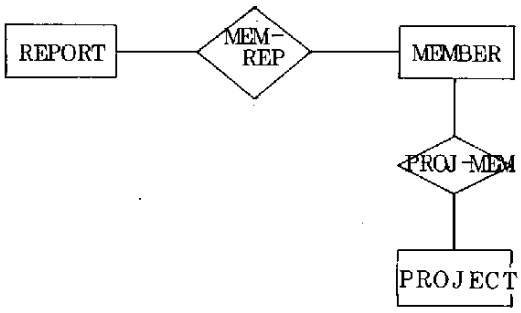


図4.33　PMDBSのLCSのE－Rモデル表現

ESR MEMBER (mno, mname, loc, alma-mater, deg, deg-year,
birth-year
PROJECT (pno, pname, leader, syar. eyar, budget)
REPORT (repno, title, abstract, source, volno, year, month,
page, cost, no-of-auth)
RSR PROJ-MEM (pno, mno, position)
MEM-REP (mno, repno, author-no)

図4.34　PMDBSのLCSのRD

4.6.2　プロジェクトデータベースシステム（PRDBS）

PRDBSは、クリアリングセンタ（CC）に存在し、研究テーマに対して、それを研究しているプロジェクトの情報を保有している。PRDBSの局所内部スキーマ（LIS）はDBTGタイプである〔図4.35〕。このLCS　E－Rモデル表現を図4.36に示し、図4.37はLCS　RDを表わしている。異種性情報（HI）は図8.31～33に示してある。

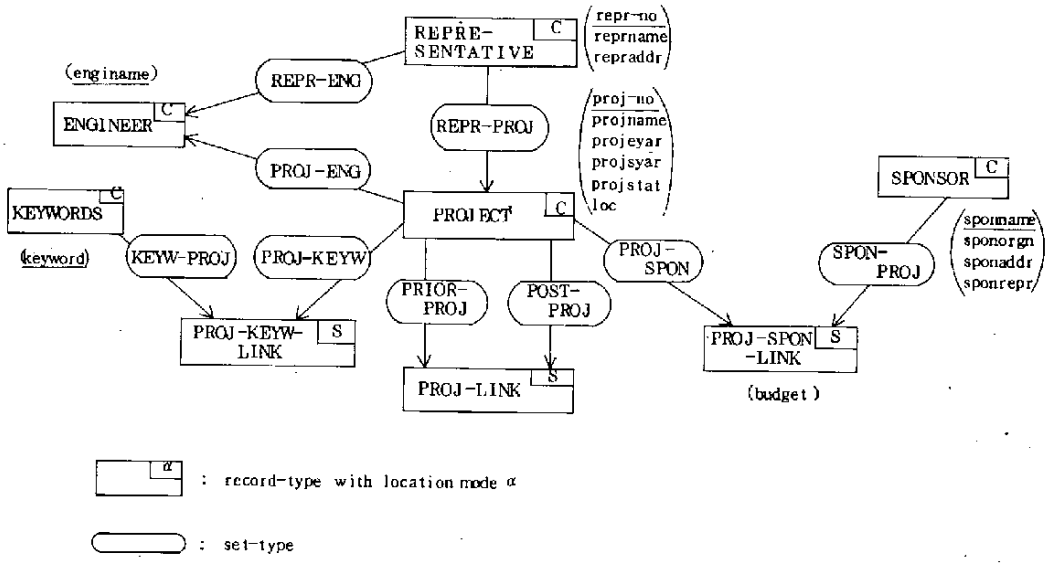


図4.35　PRDBSのLIS

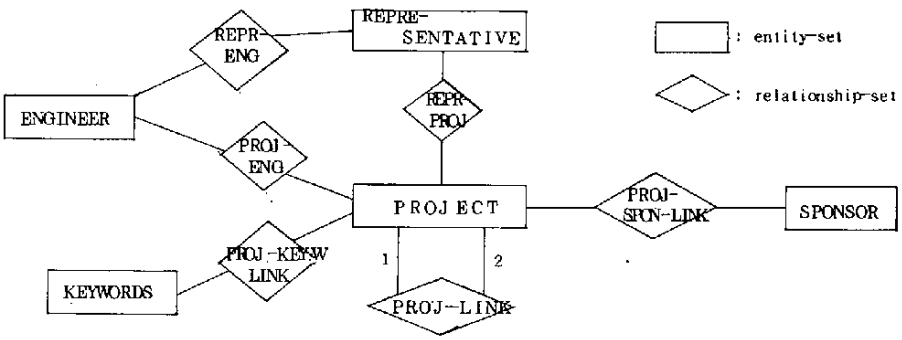


図4.36　PRDBSのLCS　E－Rモデル表現

ESR REPRESENTATVE (repr−no, reprname, repraddr)

PROJECT (proj−no, projname, projeyar, projsyar, projstat, loc)

SPONSOR (sponname, sponorgn, sponaddr, sponrepr)

KEYWORDS (keyword)

ENGINEER (enginame)

RSR REPR−PROJ (repr−no, proj−no)

PROJ−SPON−LINK (proj−no, sponname)

PROJ−KEYW−LINK (proj−no, keyword)

REPR−ENG (repr−no, enginame)

PROJ−ENG (proj−no, enginame)

PROJ−LINK (proj−no, /PROJ−NO, prior−no/PROJ−NO)

図4.37 PRDBSのLCS RD

4.6.3 文献データベースシステム（BLDBS）

BLDBSは、ドキュメントセンター（DC）にあり、全国の研究所で発表された論文の情報を持っており、ユーザはキーワードによって 論文を検索できる。BLDBSの局所内部スキーマ（LIS）はIMSタイプである〔図4.37〕。これの局所概念スキーマ（LCS）E−Rモデル表現を図4.38に、LCSのRDを図4.39に示す。

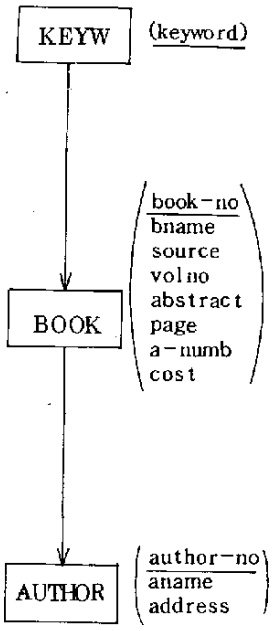


図4.37 BLDBSのLIS

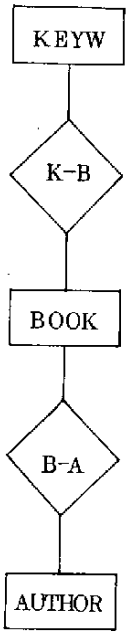


図4.38 BLDBSのLCS E-R図式

ESR KEYW (keyword)

BOOK (book-no, bname, source, volno, abstract, page, a-numb, cost)

AUTHOR (author-no, aname, address)

RSR K-B (keyword, book-no)

B-A (book-no, author-no, a-no)

図4.39 BLDBSのLCS RD

4.7 HIPS

異種性情報生成システム（HIPS）は、局所内部スキーマ（LIS）を入力として、局所概念スキーマ（LCS）とともに異種性情報（HI）を生成するシステムである〔図4.40〕。HIPSの詳細は、8.1節を参照されたい。我々がインプリメントしたシステムは、DBTGスキーマに基づいた入力をすると、E-Rモデルに基づいた局所概念スキーマと、対応情報としての異種性情報（HI）とを出力する。

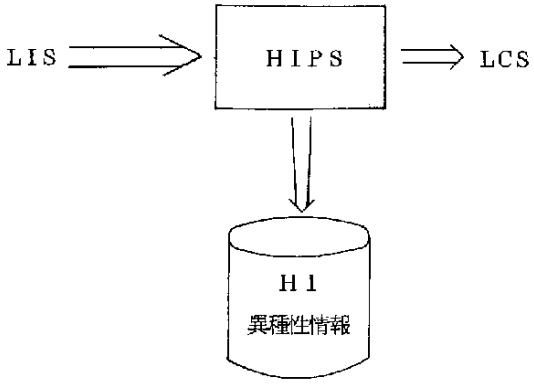


図4.40　ＨＩＰＳの概要

4.8　まとめと問題点

本章では、局所内部スキーマ（ＬＩＳ）層から、局所概念スキーマ（ＬＣＳ）層の設計問題、即ち同種化について論じた。特に局所内部スキーマ（ＬＩＳ）の記述用データモデルとして既存の代表的３つのデータモデル、即ちリレーショナルモデル、ＤＢＴＧモデル、ＩＭＳモデル、をＥ－Ｒモデルに基づいたリレーショナル記述（ＲＤ）への変換問題について論じた。これは、Ｅ－Ｒモデルの要素と、既存データモデル要素とを１対１に対応させることによってなされる。この対応情報は、異種性情報（ＨＩ）として、ネットワークデータディレクトリ内に格納される。本章では、異種性情報のスキーマを示した。

同種化では、既存データモデルに対して、種々の制限を設けている。現在の同種化のレベルは、データモデル要素間の１：１又は１：ｎの全関数性のみである。これは例えばＤＢＴＧモデルでは、セット型に対してＡＵＴＯＭＡＴＩＣかつＭＡＮＤＡＴＯＲＹのメンバシップクラスを設けていることに対応している。モデルの表わしうるインデクリティ条件をデータモデル構造と呼ぶことにする。データモデルの持つ全てのモデル構造、即ち１：１、１：Ｎ、Ｎ：Ｍ及び全関数性（ｔｏｔａｌ　ｆｕｎｃｔｉｏｎ）及び半関数性（ｐａｒｔｉａｌ　ｆｕｎｃｔｉｏｎ）を局所概念スキーマ層で表わせることが必要である。上述した全てのデータモデル構造をリレーショナルモデルによって完全に表わし得るかは、今後多くの検討が必要である。このためには、未定義値（ｎｕｌｌ　ｕａｌｕｅ）の概念の導入等が必要である。

5. 問合せ変換（ＱＴ）〔TAKIM79a,b,TAKIM80a〕

問合せ変換（query translation or ＱＴ）は、局所概念スキーマ（ＬＣＳ）レベルの問合せから局所内部スキーマ（ＬＩＳ）レベルのアクセス言語への変換であり〔図5.1参照〕、4章で述べた同種化の逆過程である。問合せ変換において必要な情報は、同種化で除去された異種性情報（ＨＩ）内のＬＣＳとＬＩＳとの対応情報である。ここでＬＣＳに基づいた問合せをＬＣＳ問合せ（ＬＣＳ query or ＬＱ）と呼ぶ。ＬＣＳ問合せは分散型データベースシステム全体で共通なリレーショナル計算言語ＱＵＥＬ〔HELD G75、YOUSK77〕〔付記1を参照〕を用いて記述されている。一方、ＬＩＳに基づいたアクセス記述をＬＩＳ DMLプログラム（ＬＩＳＤ）と呼び、この記述言語をＬＩＳ DMLと呼ぶ。ＬＩＳ DMLは、既存ＤＢＳが実行可能なデータアクセス言語である。

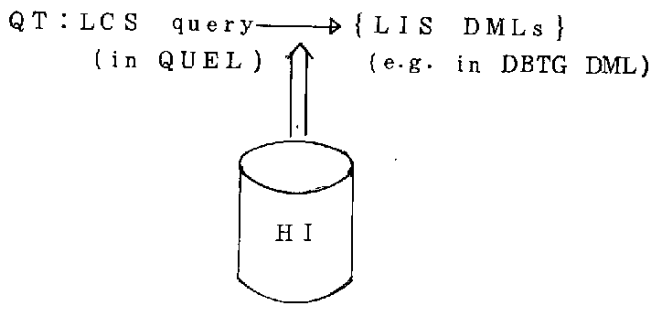


図5.1　問合せ変換（ＱＴ）の定義

本章は、特にＬＩＳ DMLとしてDBTG DML〔ＤＡＴＥＣ７７の第２２章〕を設定した場合について論じ、その後ＩＭＳ等の他のデータモデルに基づいたアクセス言語についても考えることにする。まず、5.1では、問合せ変換（ＱＴ）を論じるうえで、我々が設けたいくつかの基本的仮定について論じる。5.2では、ＱＴの処理概要を述べる。5.3、5.4、5.5では問合せ変換の３つの主要な処理モヂュール各々を論じる。5.6では問合せ変換のインプリメンテーションについて述べ、5.7ではその評価を行なう。本章で述べた問合せ変換（ＱＴ）システムのインプリメンテーションについては、8.2節において論じてある。

5.1　基　本　仮　定

本節では、問合せ変換（ＱＴ）を論じるうえで、我々の設けた基本仮定について述べる。これらの仮定は次の様である。

1) 変換目標としてのＬＩＳ DMLとしてはCODASYL DBTG DMLを考える。DBTG DMLとしては、〔ＤＡＴＥＣ７７〕に基づくものとする。

ＬＩＳ要素については、さらに次の様な仮定を設ける。

i) レコード型のキー項目はＣＡＬＣ項目でありDNA（DUPLICATE NOT ALLOWED）とする。

ii) セット型のメンバーシップクラスはＡＵＴＯＭＡＴＩＣ MANDATORYとする。

iii) 子レコード型において、あるセットオカーランス内での子レコードオカーランスのキー項目の値はユニーク（ＤＮＡ）であるとする。

2) ＬＣＳ問合せ（ＬＱ）は、ＱＵＥＬによって記述されている。これは、我々のＦＳＳ概念における基本前提である〔3.1及び4.3節を参照のこと〕。

3) ＬＱは、ＳＵＭ，ＣＯＵＮＴ，ＭＡＸ，ＭＩＮといったaggregate関数〔4.3.1を参照〕を含まない。問合せがaggregate関数を持っている時、まずaggregate関数を最初の問合せとは独立な1つの問合せとして先に処理させる。次に、この結果の定数によって最初の問合せ内の対応するaggregate関数を置き換える。aggregate関数を持つ問合せは、このようにaggregate-freeな形で各ＤＢＳサイトへ発せられるとする。

4) ＬＱとしては、検索機能のみを考える。追加（append）、削除（delete）、置換（replace）といった更新機能は、一般に、更新すべきデータの所在場所を見つけるための検索に続く、データベース（ＤＢ）への書き込みと考えられる。セキュリティ、インテグリティの制御については、これらの制約がＬＱと同一の言語レベル、即ちリレーショナル計算を用いて記述出来るならば、問合せ変形（query modification）技術〔ＳＴＯＮＭ７６〕によって更新個所を指定する検索文の中にユーザと同じ記述レベルで組み込めると考えている。冗長コピーの存在する場合の更新は、各データベースシステム間（ＮＡ間〔3.3節を参照〕）のプロトコルの問題となり、問合せ変換とは独立な問題である。

5) ＬＱの条件部（qualification）は、積正規形（conjunctive normal form or ＣＮＦ）であるとする〔図5.2〕。図5.2で<pred>は x. a＝"Ａ"のような比較論理式を表わしている。又ＯＲで結ばれた<pred>は全て同一の組変数（tuple variable）を参照していなければない。論理式を手続き化するためには、式が積正規化されていなければならない。

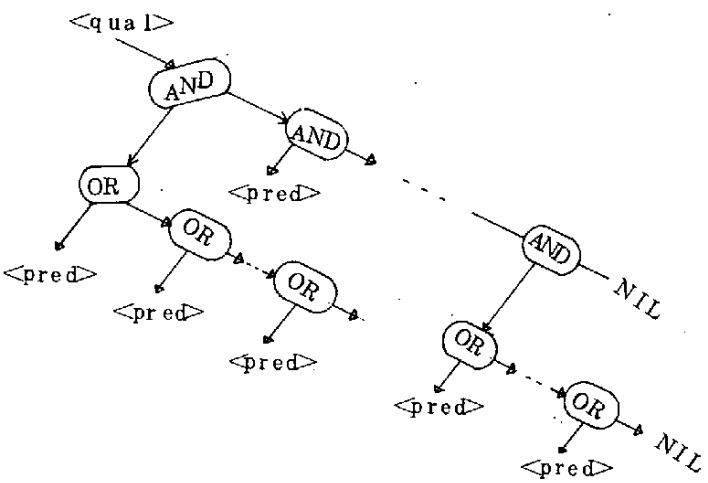


図 5.2　積正規形（木表現）

6) ＬＱ内の結合式（join predicate）としては等価結合（equi-join）のみを考え、ＬＣＳ内の関係性集合（ＲＳ）に沿って書かれねばならない。即ち、結合式の比較演算子の少くなくともどちらか1つのオペランドは、ＲＳリレーション（ＲＳＲ）を参照しており、他方はこのＲＳに関係のある（リンクされた）リレーション(RSR

or ESR) でなければならない。これ以外の結合式(例えば、無関係な2つのESRの結合)は無意味なものとする。

7) 問合せに対する応答時間は、目標LIS上のアクセスされるオカーランス数に比例するものとする。実際の応答時間は、物理的なページアクセスが重要な要因となるが、ここでは考えないことにする。何故ならページアクセスは各DBSのインプリメンテーションに大きく依存した部分であり、データベースシステムの内部スキーマの問題である。我々の想定するLISは、データベースシステムの概念又は外部スキーマであり、我々の目標はこのレベルのデータアクセス言語のシーケンスへ変換することである。

5.2 問合せ変換(QT)の処理概要

問合せ変換(QT)の処理概要を図5.3に示す。QTは、1)構造変換(structure transform or ST)、2)最適化(optimizer or OPT)、3)DML生成(DML generator or DMLG)の3つの主要なモデュールから成っている。

データアクセス言語は、アクセスしようとするスキーマを記述しているデータモデルによって規定されている。このため、問合せを変換するためには、先ず、問合せが参照するデータモデル構造を、目標言語のデータモデル構造へ変換することが必要になる。例えば、リレーショナル問合せの参照するリレーションを、対応するDBTGモデルのレコード型へ変換することである。この過程を構造変換(ST)と呼ぶ。変換に必要なモデルの対応情報は、異種性情報(HI)内に格納されている。特に構造変換に必要な情報、即ちモデル構造に関するものをHIースキーマ情報と呼ぶ。

構造変換された問合せは、目標LISのデータモデルによって問合せの参照する全てのLIS要素を非手続き的に表わしたものである。実際には図5.3で示した様に、DBTG問合せグラフ(DBTG query graph or DQG)と呼ばれるグラフ表現となる。次の問題は、この様な非手続的なグラフ表現から、LISのデータモデル(即ちDBTGモデル)に基づいたアクセスのシーケンスを得ることである。一般にグラフからは複数の可能なシーケンスを得ることができるので、この中で最適と思われるものを決定せねばならない。我々は、アクセスされるオカーランスがなるべく少なくなる様なアクセスシーケンスを求める。これを最適化(OPT)と呼ぶ。

OPTから出力されるものは、目標LISの基本アクセス単位のシーケンスとなっている。DML生成(DMLG)は、このアクセスシーケンスを受けとり、個々の基本アクセス単位ごとにDMLのブロックを生成する。アクセスシーケンスに基づいたこれらのDMLブロックの集合は、目標とするLIS DMLプログラムとなる。

このように、我々の問合せ変換(QT)は、DBTGモデルを対象としているが手法は全く一般的であり、他のモデルへも容易に適応させうる。

以降ST, OPT, DMLGについて詳論する。このなかで用いる例として、4.6.2で述べたプロジェクトデータベースシステム(PRDBS)に対する問合せを考えることにする。LCS問合せ(LQ)としては、次のようなものを考える。

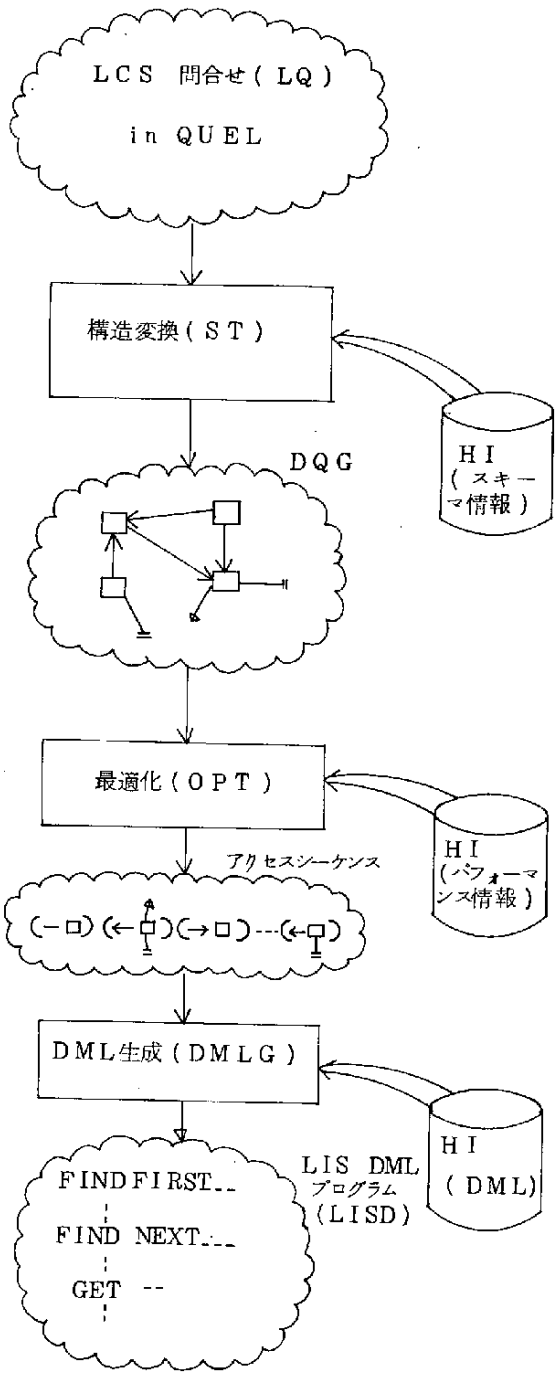


図5.3　QTの概要

「データベースを1975年以降研究しているプロジェクトの代表者の部下が、同じプロジェクトに属している場合、そのプロジェクト名とその部下の名前を求めよ。」この問合せをQUELで書くと図5.4の様になる。

```
range of (e, p) (ENGINEER,
    PROJECT);
range of (re, rp, pe) (REPR-
    ENGI, REPR-PROJ, PROJ-
    ENGI);
range of (pk1) (PROJ-KEYW,
    LINK);
retrieve into R (p.projname,
    e.enginame)
where pk1.keyword="database"
      and
      pk1.proj-no=p.proj-no
      and p.proj-no=rp.proj
      -no and
      rp.repr-no=re.repr-no
      and re.enginame=e.en-
      giname and
      e.enginame=pe.engina-
      me and pe.proj-no=p.
      proj-no and
      p.projsyar≧1975 and
      p.proj-stat="ON";
```

図5.4　LCS問合せ(LQ)の例

以降図5.4の例に基づいて、検討する。

5.3 構造変換（ＳＴ）

ユーザから視たデータベースシステム（ＤＢＳ）は、あるデータモデルで記述されたスキーマとこのデータモデルに基づいたアクセス言語とによって特徴づけられる。よって、２つの異なったアクセス言語を変換するためには、まず第１に、言語の基礎となっているデータモデルの対応づけ、即ちあるデータモデルによる問合せ構造を他のモデル構造への変換を行なわねばならない。同種化〔第４章を参照〕においては、ＬＩＳ要素をＥ－Ｒモデルへ１対１に対応させることによって局所概念スキーマ（ＬＣＳ）を生成し、この対応情報を異種性情報（ＨＩ）として除去した。同種化の逆過程である問合せ変換（ＱＴ）では、逆にＨＩを用いてＬＣＳ問合せの参照するＬＣＳ要素を対応するＬＩＳ要素におき換えることによって目標ＬＩＳのデータモデル構造へ変換できる。これを問合せの構造変換（ｓｔｒｕｃｔｕｒｅ ｔｒａｎｓｆｏｒｍ）と呼ぶ。

構造変換では問合せをグラフで表わし、このグラフの変形を通して問合せ構造の変換を行なう。ＬＣＳ問合せのグラフ表現をリレーショナル問合せグラフ（ｒｅｌａｔｉｏｎａｌ ｑｕｅｒｙ ｇｒａｐｈ）と呼ぶ。構造変換の最終出力は、ＤＢＴＧ構造による問合せの構造表現としてのＤＢＴＧ問合せグラフ（ＤＢＴＧ ｑｕｅｒｙ ｇｒａｐｈ）である。本節では、これらの２つのグラフ表現を論じるとともに、これらの２つのグラフ間の変換について論じる。

5.3.1 リレーショナル問合せグラフ（ＲＱＧ）

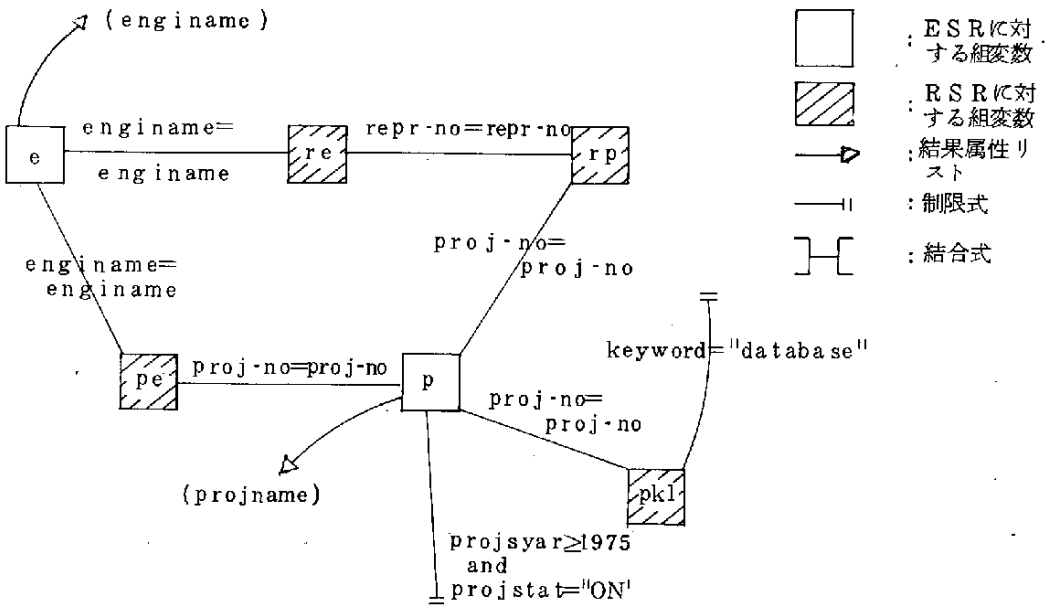


図 5.5　図 5.4 のＲＱＧ

ＬＣＳ問合せ（ＬＱ）は、図 5.5 に示す様なグラフによって表わせる。ノードは組変数

(tuple variable) に対応している。特に斜線のついたノードは関係性集合リレーション(RSR)に対応する変数を、他のノードは事象集合リレーション(ESR)に対応した変数を表わしている。例えばノードreは、RSR REPR—ENGIに対応し、ノードpは ESR PROJECTに対応している。リンクは3種類のものがある。ノード間に存在するリンクは、これらのノードを結合(join)する論理式を表わしている。これを結合リンク(join link)と呼ぶ。例えばノードpとpeとの間には、結合式 p・proj-no=pe・proj-no を表わす結合リンクが存在している。あるノードから出る矢印のついたリンク(——▷)は、このノードの表わす変数に関する結果属性(result attribute)を表わしている。これを結果リンク(result link) と呼ぶ。例えば、ノードpをみると、変数pの参照するリレーションPROJECTから属性 projname の値を出力することを示す結果リンクがついている。最後のリンクは制限リンク(restriction link) と呼ばれるもので、電気回路のアース記号(——⫪)によって表わされている。これはこのリンクのついたノードに制限式 (restriction formula) があることを示している。例えばノードpをみてみよう。これは、制限式 p・projsyar ≧ 1975 and p・projstat="ON" を持っていることを示している。

RQGは、制限リンクと結合リンクとによって表わされる論理式の論理積(conjunction)を表わしていることは注意されるべきである。即ち、x.a=y.a and y.b=z.c and (x.d="A" or x.d="C")のような積正規化(conjunctive normal form) された論理式は、図5.6のようにRQGによって表わされる。又5.1節の基本仮定において、or で結ばれた論理式は全て同一の変数をもたねばならないと述べたことを思い出していただきたい。このことよりx.a=y.a or y.b=z.c のような論理式の和(disjunction) をRQGは表わせない。この様に、or が結びつける論理式が異なった変数を参照している時には、問合せの条件式を和正規化し、or で結ばれた各論理式ごとに問合せを生成し、これらを独立に処理させ、これらの結果の和(union) をとる必要がある。

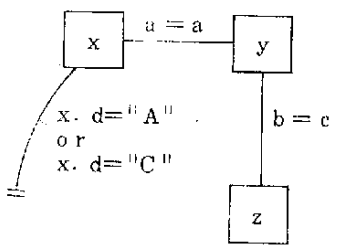


図5.6 RQGの例

リレーショナル問合せをこの様にグラフ表現したものを、リレーショナル問合せグラフ(relational query graph or RQG)と呼ぶ。

5.3.2 隠れ構造(HS)

次に、リレーショナル問合せグラフ(RQG)を用いて、LCS問合せ(LQ)のLCS構造(即ちリレーショナルモデル)をDBTGモデル構造に変換することを試みる。先ず、RQG内のLCS要素(ESR、RSR属性)をそのサイトの異種性情報(HI)を用いて対応するLI

S要素（レコード型、セット型、リンクレコード型、項目名）によって置き換える。図5.5のRQGでノードeは、LCS要素のESR ENGINEERを表わしている。PRDBS〔4.6.2を参照〕の異種性情報（HI）によると、LCSのESR ENGINEERは、LISのレコード型ENGINEERであることがわかる。よって、ノードeをレコード型を表わすように変える。次にRQG内のRSR REPR−PROJを表わすノードrpを考えてみよう。同じくHIを用いて、これがセット型REPR−PROJに対応していることがわかるので、このノードrpをセット型を表わすように変える。この様にしてRQG〔図5.5〕から生成されたグラフを図67に示す。このグラフはDBTG構造を表わしているので、これをDBTG問合せグラフ（DBTG query graph or DQG）と呼ぶ。四角はレコード型を長円はセット型を表わしている。RQG内の結合リンクは、DQG内のセット型を介した親子レコード型の組へ変換される。RQGの結果リンクと制限リンクとは、DQGでもそのまま保存される。

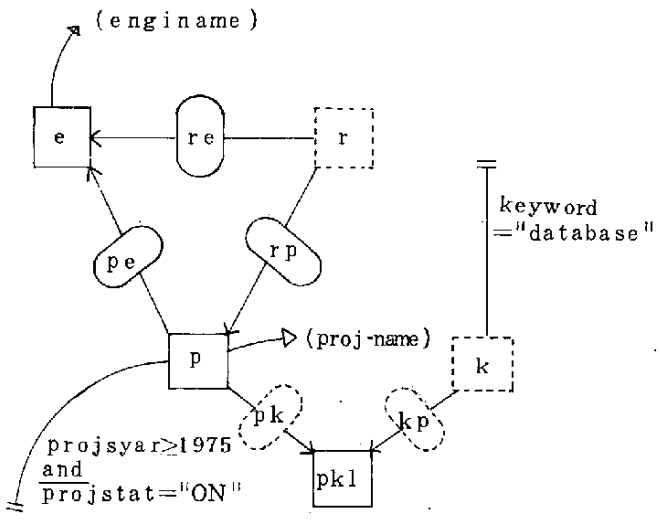


図5.7 RQG〔図5.5〕のDQG

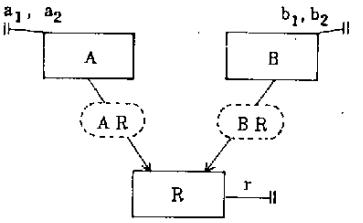
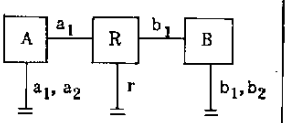
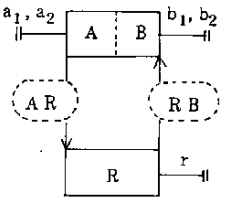
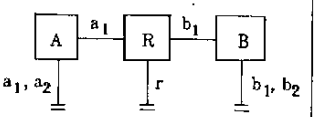
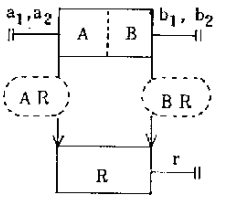
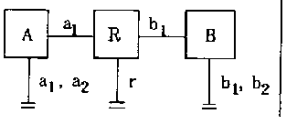
図5.7内のノードrを考えてみよう。ノードrはLIS内のレコード型REPRESENTATIVEを表わしている。しかし、図5.5のRQG内には、これに対応するノードは存在していない。これは、LCS内の関係性集合リレーション（RSR）は、これが関係づける2つの事象集合リレーション（ESR）の各々の主キー属性を、その複合キーとして持っているからである。即ち、ESRの主キー属性のみを参照する問合せは、このESRではなく、これと関連するRSRのみを参照できるからである。2つのRSR REPR−ENGIとREPR−PROJとは、ともにESR REPRESENTATIVEの主キー属性repr-noを持っている。よって、REPRESENTATIVEとREPR−ENGI及びREPR−PROJとの2つの結合は、REPR−ENGIとREPR−PROJの属性repr-noにつ

いての結合として表わせることになる。よって図5.5のRQG内には、ＥＳＲ　ＲＥＰＲＥＳＥＮＴＡＴＩＶＥに対応するノードがない。この様に、ＬＩＳ要素のなかで、これに対応するものが存在するが、ＬＣＳ問合せ（ＬＱ）には表われないＬＩＳ要素（ i.e. レコード型）を明隠れ構造（ explicitly hidden structure or EHS）と呼ぶ。ＤＱＧ〔図5.7〕内のノードｋも又ＥＨＳである。

次に、図5.7内のノードpk とkp とを考えてみよう。これらは各々セット型ＰＲＯＪ－ＫＥＹＷとＫＥＹＷ－ＰＲＯＪとを表わし、リンクレコード型ＰＲＯＪ－ＫＥＹＷ－ＬＩＮＫの２つのセット型で、各々レコード型ＰＲＯＪＥＣＴ，ＫＥＹＷＯＲＤＳと関連している。ＰＲＤＢＳの局所概念スキーマ（ＬＣＳ）〔5.6.2節を参照〕を見ると解かる様に、これらのセット型についての情報はＬＣＳ上には存在しない。このことはＤＢＴＧモデルとＥ－Ｒモデルとの対応づけ〔4.5.3を参照〕において、リンクレコード型は関係性集合（ＲＳ）に対応させているために生ずる。この様にＬＣＳにも現われないセット型は、異種性情報（ＨＩ）のＲＳＲリレーション〔4.5.3を参照〕をサーチして見つけねばならない。ＬＩＳ要素の中で、対応するものがＬＣＳにもＬＱにも表われないもの、即ち、ＤＢＴＧモデルではリンクレコード型に関連するセット型を暗隠れ構造（ implicitly hidden structure or ＩＨＳ）と呼ぶ。ノードpk とkp とはＩＨＳである。

ＩＨＳとＥＨＳを総称して隠れ構造（ＨＳ）と呼ぶ。構造変換（ＳＴ）では、ＨＩを用いてＲＱＧ内のノードとリンクとを、対応するＬＩＳ要素へ置き換えるとともに、前述した隠れ構造（ＨＳ）を明らかにせねばならない〔図5.9〕。ＩＨＳとＥＨＳのパターンの詳細を図5.8に示す。RDは、局所概念スキーマ（ＬＣＳ）のリレーショナル記述を示している。リレーションＡとＢとは事象集合リレーション（ＥＳＲ）であり、リレーションＲは関係性集合リレーション（ＲＳＲ）である。この図は、ＬＣＳリレーションＡ，Ｂ，Ｒに対する問合せに対応するDBTG構造を示している。ＤＢＴＧ図式では、四角はレコード型を長円はセット型を表わしている。点線で囲まれた四角形と長円とは、各々EHSとIHSとを示している。

1) IHS

RD	DBTG 図式	問合せグラフ（QG）
ESR： A（$\underline{a_1}$, a_2） B（$\underline{b_1}$, b_2）		
RSR： R（$\underline{a_1}$, $\underline{b_1}$, r）		
：IHS		

2) EHS

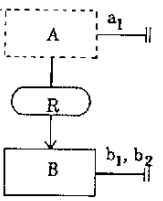
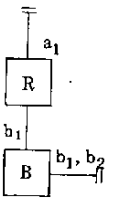
RD	DBTG 図式	問合せグラフ（QG）
ESR： A（$\underline{a_1}$, a_2） B（$\underline{b_1}$, b_2） RSR： R（$\underline{a_1}$, $\underline{b_1}$） ：EHS		

図5.8 隠れ構造 (1)

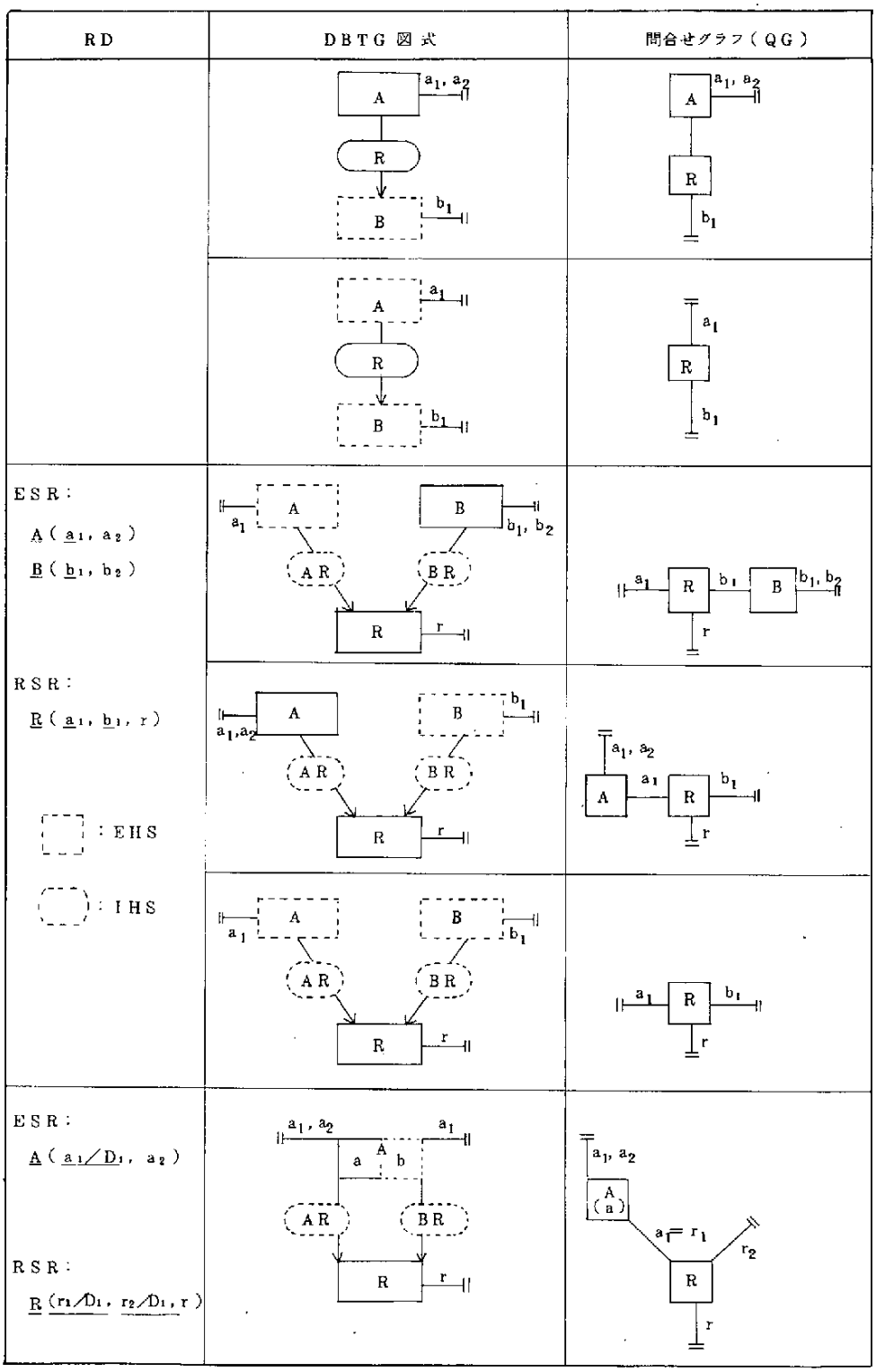
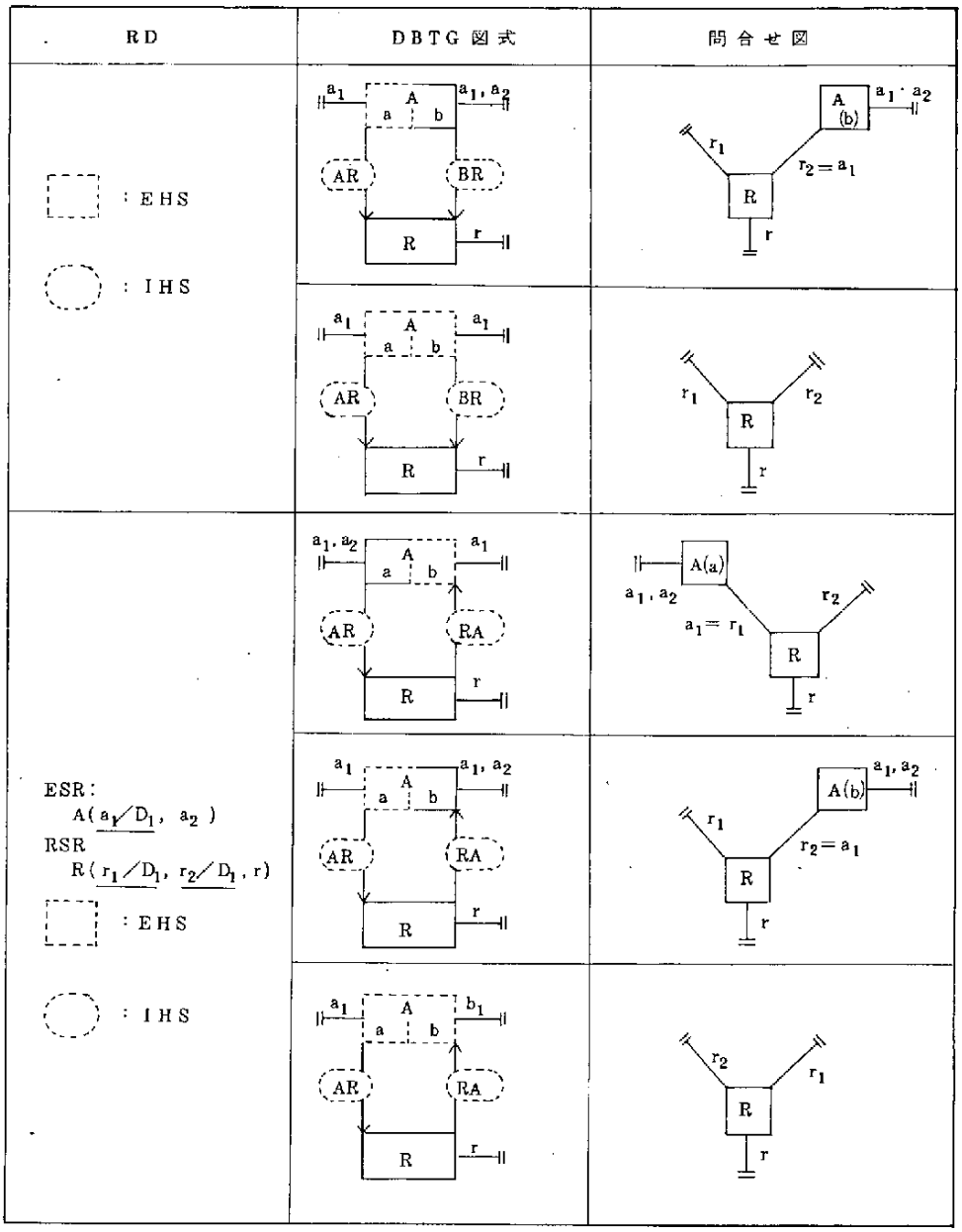

RD
DBTG 図式
問合せグラフ（QG）
ESR：
A（a1, a2）
B（b1, b2）
RSR：
R（a1, b1, r）
：EHS
：IHS
ESR：
A（a1／D1, a2）
RSR：
R（r1／D1, r2／D1, r）

図5.8 隠 れ 構 造 (2)



ここで、各リンク上の a_i b_j は、各々のリンクが表わしている条件式が参照する属性又は結果属性を示している。

図5.8 隠 れ 構 造 (3)

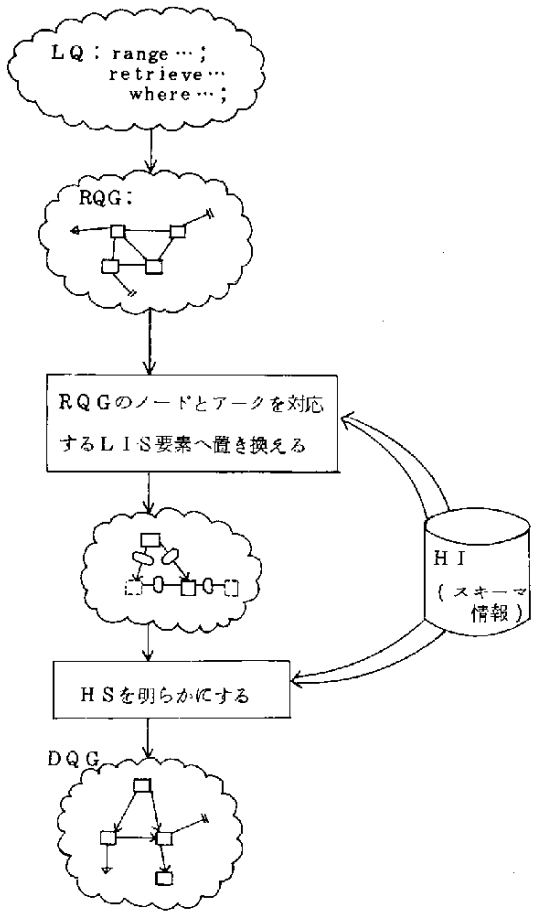


図5.9　構造変換の概要

5.3.3　ＤＢＴＧ問合せグラフ（ＤＱＧ）

図5.7のＩＨＳとＥＨＳとを明らかにしたグラフを図5.10に示す。ノードはレコード型を表わし、ノード間有向リンク（アーク）はレコード型の親子関係を示すセット型を示している。ノード内の記号とアーク上の記号とは、各々レコード型とセット型を表わす変数である。これは又ＬＣＳ問合せ（ＬＱ）内の組変数（tuple variable）にも対応している。この様なグラフ表現をＤＢＴＧ問合せグラフ（ＤＢＴＧ query graph or ＤＱＧ）と呼ぶ。ＤＱＧは、ＬＣＳ問合せの問合せの意味を、ＤＢＴＧモデルによって非手続的に記述したものである。このＤＱＧは構造変換（ＳＴ）の最終出力となる。

図5.11はＤＱＧの定義を、図5.12は図5.10をこれに基づいて記述したものである。

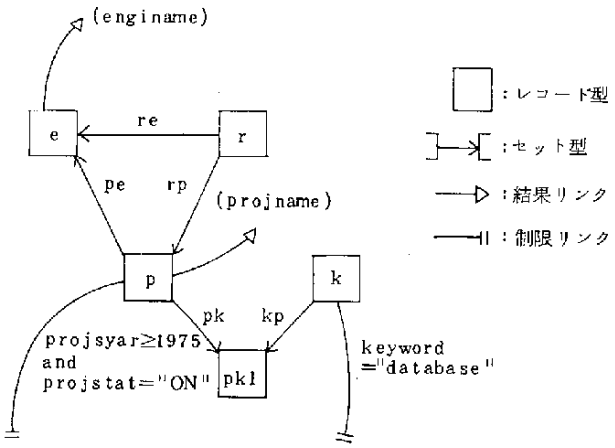


図 5.10　ＤＢＴＧ問合せグラフ（ＤＱＧ）

```
<DQG>::= (DQG (ARC   <arc>  {<arc> })
              (NODE <node> {<node>}))

<arc> ::= (<set-var> (SET <set-type>)
             (OWNER <rec-var>)
             (MEMBER <rec-var>))

<node>::=(<rec-var> (REC <rec-type>)
             [(RSTR <restriction>)]
             [(RSLT <result-att-list>)])
```

図5.11 DQGの定義

```
(DQG
 (ARC   (RE (SET REPR-ENGI)(OWNER R)(MEMBER E))
        (PE (SET PROJ-ENGI)(OWNER P)(MEMBER E))
        (RP (SET REPR-PROJ)(OWNER R)(MEMBER P))
        (PK (SET PROJ-KEYW)(OWNER P)(MEMBER PKL))
        (KP (SET KEYW-PROJ)(OWNER K)(MEMBER PKL)))

(NODE   (E  (REC ENGINEER)(RSLT (ENGINAME)))
        (R  (REC REPRESENTATIVE))
        (P  (REC PROJECT)
            (RSLT PROJNAME)
            (RSTR (AND (GE PROJSYAR 1975)
                       (EQ PROJSTAT (QUOTE ON)))))
        (PKL(REC PROJ-KEYW-LINK))
        (K  (REC KEYWORDS)
            (RSTR (EQ KEYWORD (QUOTE DATABASE))))))
```

図5.12 図5.10のDQG記述

5.4 最 適 化 （OPT）

構造変換（ＳＴ）で生成されたＤＢＴＧ問合せグラフ（ＤＱＧ）は、ＬＣＳ問合せ（ＬＱ）の意味をＤＢＴＧモデルによって非手続的に記述したものである。この非手続的グラフ表現から、アクセスの手続きを生成することが次の問題となる。ＤＱＧから生成されるアクセスパスは、ＤＱＧ内の全てのノードと全てのアークとを持たねばならない。このアクセスパスは、アークとこれに結合したノードとの対のシーケンスとして表わされる。

本節では、まず種々のパスのなかから最適なものをみつける目標関数について論じる。ついで、パスの選択の規準となるアクセスされるオカーランス数の見積りについて検討する。最後に、これらをもとにしたＤＱＧからアクセスのシーケンスを表わす木（アクセス木と呼ぶ）への変換アルゴリズムについて論じる。

5.4.1. 目 標 関 数

最適化の目的関数は、次の２点である。

1) 中間結果の生成（数と量）を最少とする。

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。

ＤＢＴＧはリレーショナルモデルと異なり、完全な（ｃｏｍｐｌｅｔｅ）〔ＣＯＤＤＥ７１，ＤＡＴＥＣ７７〕アクセス言語を提供できない。即ち、リレーショナルモデルでは問合せによって生成された結果に対して、これをリレーショナルモデルとして再度問合せを発することができる。しかし、ＤＢＴＧ　ＤＭＬでは一度導出された結果を再度ＤＭＬを用いてＤＢＴＧモデルとしてアクセスできない。このためＤＢＴＧモデルでは、導出された中間結果は、親言語（ｅ．ｇ．ＣＯＢＯＬ）によってＤＢＴＧ　ＤＭＬとは独立に物理的ファイルのアクセスを行なわねばならない。これはアクセスの一貫性（ｃｏｈｅｒｅｎｃｙ）の点から望ましくない。従って、なるべく中間結果リレーションの数と量とを少なくし、アクセスパスの先頭のレコード型から最後のレコード型まで一貫してＤＢＴＧ　ＤＭＬでアクセスできることが、スキーマレベルでは最も重要である。実際には中間結果の生成は、これらの中間結果間で結合（ｊｏｉｎ）等の集合演算を行なうことを意味している。結合演算は多くの計算時間を要するものであり、アクセスの高速性及び運用からも中間結果の数と量とをなるべく少なくすることが望ましい。

5.1節で、ＤＢＳへのアクセスの応答時間は、アクセスされるオカーランス数に比例すると仮定した。このため出来る限りアクセスされるオカーランス数を少なくすることが望ましい。

5.4.2 アクセスされるオカーランス数の見積もり

可能な幾つかのパスのなかから、１つのパスを決定する時、アクセスされるオカーランス数がなるべく少ないものを選ぶことは、5.4.1の２）の目標によって良い選択となる。ここでは、あるパスが与えられた時、このパス内でアクセスされるオカーランスの期待値（ＯＣＡと記す）について考えることにする。

ＤＱＧがｍコのノードをもっているとする。この時、ＤＱＧ内のノードを１からｍまで適当に番号づける。a_{ij} をｉ番目のノード（n_i と記す）のｊ番目の属性とする。c_i を n_i のカーディナリティ、即ち、n_i 内の全オカーランス数とする。s_{ij} を属性 a_{ij} の選択度（selectivity）〔ＨＥＶＮＡ７８，ＳＥＬＩＰ７９〕とする（$0 \leq s_{ij} \leq 1$）。s_{ij} はノード n_i で属性 a_{ij} がとるユニークな値の数を n_i のカーディナリティで割ったものとして定義される。これらを用いるとノード n_i に関する等価制限式 $a_{ij}=v$ を満足するｎｉ内のオカーランス数の期待値（ＯＣＡ）をもとめることができる。ここでｖは a_{ij} が n_i 内でとり得るある定数値とする。この期待値は $c_i \cdot s_{ij}$ である。

Ａ．１つのノードのアクセス

n_i に関する制限式は一般に制限述語の論理式となる。s_i を n_i に関する制限式の選択度とする。これは表5.1のようにして求められる。5.1で述べた仮定より制限論理式は、積正規形（ＣＮＦ）であるとする。この時、制限論理式は、キー（ｉ.ｅ. ＤＢＴＧのＣＡＬＣ項目）を参照する論理式と、そうでないものとの論理積として変形できる。前者のなかで等価制限式のみを考えた時の選択度を s_i'、同様にして後者の選択度を s_i'' とする。ここで $s_i = s_i' \cdot s_i''$ の関係が成り立つ。あるノード（n_i）を独立してアクセスする時のアクセスされるオカーランス数（これを OCA_i と記す）を考えてみる。n_i の制限式がＣＡＬＣを参照していない時（$s_i'=1$，$s_i=s_i''$）は、全てのオカーランスをサーチせねばならない（ソートされていないとする）ので $OCA_i = c_i$ となる。一方、n_i がＣＡＬＣを参照する等価制限式を持つ時は、$OCA_i = c_i \cdot s_i'$ となる。更に、条件を満足するオカーランス数の期待値（OCS_i）は前者では $c_i \cdot s_i$ となり、後者では $c_i \cdot s_i'' \cdot s_i'' = c_i \cdot s_i$ となる。

表 5.1　論理式とその選択度

論　　理　　式	選　択　度
$a_{ij}=v_1$ or $a_{ij}=v_2$ or ⋯ or $a_{ij}=v_k$	$s_{ij}+s_{ij}+\cdots+s_{ij}=k \cdot s_{ij}$
$a_{ijl}=v_1$ or $a_{ij_2}=v_2$ or ⋯ or $a_{ijk}=v_k$ ここで $a_{ijl} \neq a_{ijm}$ $l=1, \cdots, k,\ m=1, \cdots, k$ ただ $l \neq m$	$1-(1-s_{ij_1})(1-s_{ij_2})\cdots(1-s_{ijk})$ $=1-\prod_{l=1}^{m}(1-s_{ijl})$
p_1 and p_2 and ⋯ and p_k	$s_1 \cdot s_2 \cdot \cdots \cdot s_k = \prod_{l=1}^{k} s_l$ （ここで s_i は論理式 p_i の選択度）
$a_{ij} \neq v$	$1-s_{ij}$

B．セット型を経由したノードのアクセス

今までは、独立したノードに対すアクセスされるオカーランス数の見積もり（ＯＣＡ）を検討してきた。次にあるノード（ n_j とする）からセット型（Ｓとする）を介してノード n_i をアクセスする時の n_i のＯＣ A_i を求めてみよう〔図5.13〕。ここで sl_{ji} を n_j 内の1つのオカーランスとセット型Ｓによってリンクされている n_i のオカーランス数の平均値とする。これを結合度（connectivity）と呼ぶ。図5.13で(1)は、n_j がセット型Ｓの親レコード型で、n_i が子レコード型である場合、(2)は n_j がＳの子レコード型、n_i は親レコード型の場合を示している。(1)では、$sl_{ji}=n$（≥ 1）であり、(2)では $sl_{ji}=1$ である。ＯＣＳj を n_j 内の条件を満足するオカーランス数とすると n_i 内のアクセスされるオカーランス数の期待値（ＯＣＡi）は、$OCAi=OCSj\cdot sl_{ji}$ となる。

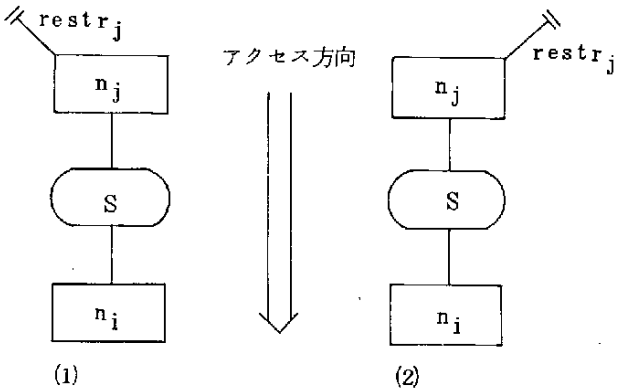


図5.13　n_j からs経由の n_i アクセス

次に図5.14の様に、ｋコのノードが線型に結ばれた場合を考えてみる。ノードは図のように順に番号づけされており、各ノード(i)は制限式 r_i をもっている。各 r_i は選択度 s_i をもっている。もし r_i がなければ $s_i=1$、あれば $0<s_i\leq 1$ となる。ノードｉとｉ＋1（ｉ＝1，…，ｋ－1）との間にはセット型が存在しており、このリンクは結合度 $sl_{i,i+1}$ を持っているとする。ノード1はアクセスの開始ノードであり、r_1' はＣＡＬＣ項目についての等価制限式の論理式、r_1'' はこれ以外の制限式であり、各々は選択度 s_1' と s_1'' を持っている。この時、ノード1からｋまででアクセスされるオカーランス数の期待（ＯＣＡ）は、

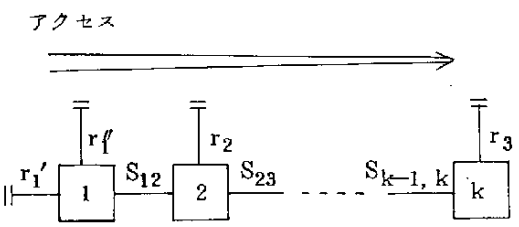


図5.14　ｋコのノードのチェイン

$$OCA=c_1\cdot s_1'\,(1+s_1''\cdot sl_{12}\,(1+s_2\cdot sl_{23}\,(1+\cdots (1+s_{k-2}\cdot sl_{k-2,k-1}\,(1+s_{k-1}\cdot sl_{k-1,k}))\cdots))))$$

となる。

さらにｋ内の条件を満足するオカーランス数の期待値（ＯＣＳｋ）は、

$$OCSk = c_1 \cdot s_1' \cdot s_1'' \cdot sl_{12} \cdot s_2 \cdot sl_{23} \cdot \cdots \cdot s_{k-1} \cdot sl_{k-1,k} \cdot s_k$$

$$= c_1 \cdot \left[\prod_{i=1}^{k} s_i \right] \left[\prod_{j=1}^{k-1} sl_{i,j+1} \right] \quad \text{となる。}$$

ノード1内の条件を満足するオカーランス数（OCS_1）は、$OCS_1 = c_1 \cdot s_1$ である。

ノードkから逆に、ノード間のセット型を介してリンクされたノード1のオカーランス数は、$OCAk / \left[\prod_{i=1}^{k-1} sl_{i,i+1} \right] = c_1 \cdot \left[\prod_{i=1}^{k} s_i \right]$ となる。この数は、ノード1からノードkまでリンクされたオカーランスが全て各ノードの条件を満足しているようなノード1内のオカーランス数の期待値である。したがって、ノード1内のオカーランスは、このパスのアクセスによって、$c_1 \cdot \prod_{i=1}^{k} s_i$ に減ることを示している。よって

$$c_1 \cdot \prod_{i=1}^{k} s_i / OCA_1 = \left[c_1 \prod_{i=1}^{k} s_i \right] / \left[c_1 \cdot s_1 \right]$$

$$= \prod_{i=2}^{k} Si \quad \text{は、このパスによるノード1に対する選択度と考えられる。}$$

C. 木のアクセス

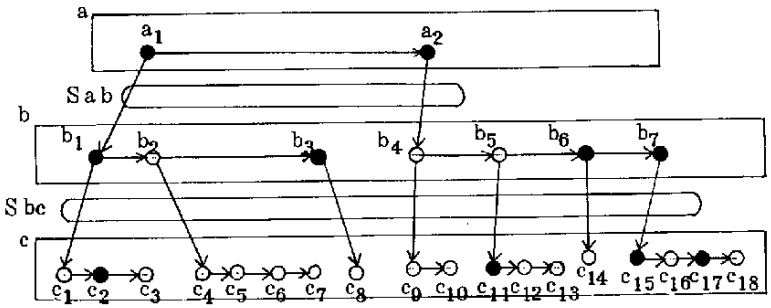
次に、ノードをレコード型、枝をセット型とした木におけるアクセスされるオカーランス数の期待値（ＯＣＡ）をもとめてみよう。木をpre-order〔KNUTD73〕にたどった時の枝とノードの対のシーケンスがアクセスのシーケンスを示しているとする。

まず木の結合度と選択度とを定義しよう。ct_T は、ある木Tの結合度を st_T はTの選択度を示しているとする。ノードはいくつかのオカーランスから構成されており、これらは互いにノード内のリンク及びセット型を介して他のノードのオカーランスとリンクされている。よって、ノードの木に対応して、木内のオカーランスも又木を形成する。オカーランスの木の中で、全てのオカーランスが条件を満足するものを、条件を満足するオカーランス木（tree of satisfiable occurrences or ＴＳＯ）と呼ぶことにする。木Tの結合度 ct_T とは木T内の条件を満足するオカーランス木（ＴＳＯ）が平均して何個の（条件を満足する）一つなぎのオカーランスの集合持つかを示している。

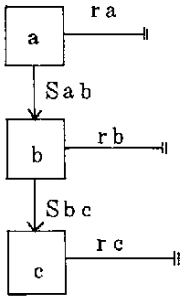
即ち、このアクセス木（ＡＴ）の結果リレーションのスキームを $\underline{R}(A_1, A_2, \cdots, Ai, \cdots, An)$ とし、属性の集合Ai, …, An はＡＴの部分木T内のＡＴノードに関する結果属性とする。又、Ai は木の根ノードについての結果属性とする。すると部分木Tの結合度 ct_T は、次の様に定義できる。

$ct_T = R〔Ai, \cdots, An〕$のカーディナリティ／$R〔Ai〕$のカーディナリティ

部分木Tの選択度（selectivity）st_T とは、Tの根ノード内のあるオカーランスが、T内の条件を満足するオカーランス木（ＴＳＯ）の根オカーランスである確率を示している。



(1) オカーランス木



(2) アクセス木

● 制限条件（ra, rb, rc）を満足するオカーランス

⊙ 〃 を満足しないオカーランス

図5.15 木とオカーランス木

例えば図5.15においてレコード型aのオカーランスa_1は、条件を満足するただ1つのオカーランス木$a_1-b_1-c_2$を持つ。よってこれは結合度1を持つことになる。

$OCAO_T$を、Tのルートノード内の1のオカーランスからつくられるオカーランス木内のアクセスされるオカーランスの平均数を示しているとする。

Tの選択度st_Tと結合度ct_T及び$OCAO_T$とは、次のように定義される。Tのルートノードをaとする。T_i'（i＝1，2，…，n）をaの部分木とする。T_i'は、木であるか又はノード（即ち葉ノード）である。s_aをaに関する制限式の選択度とする。$sl_{aTi'}$をノードaとその部分木Ti'のルートノードとの間のリンクのもつ結合度とする。リンクaTi'がノードaを親レコード型とするセット型ならば$sl_{aTi'} \geq 1$であり、子レコード型ならば$sl_{aTi'}=1$である。$st_{Ti'}$を部分木Ti'の選択度とし、$ct_{Ti'}$をTi'の結合度とする。

この時、木Tの選択度st_Tと結合度ct_Tとは、次のように定義される。

1) n＝1でTi'がノードbの時

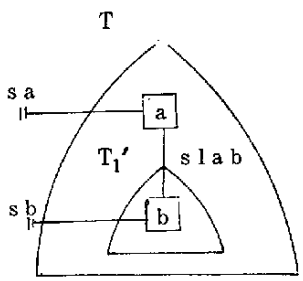


図5.15のように、ノードaとノードbとがセット型で結ばれている場合である。

この時 $st_{T_1'}=s_b$

$ct_{T_1'}=1$ である。

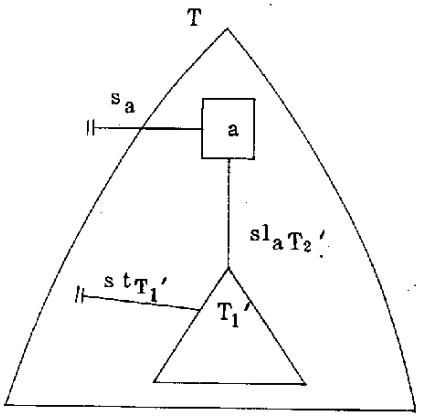
$OCAO_{T_1'}=1$

よって
$$\left.\begin{aligned} st_T &= s_a \cdot st_{T_1'} = s_a \cdot s_b \\ ct_T &= sl_{ab} \cdot ct_{T_1'} = sl_{ab} \\ OCAO_T &= 1 + s_a \cdot sl_{aT_1'} \cdot OCAO_{T_1'} \end{aligned}\right\} (1)$$

$= 1 + s_a \cdot sl_{ab}$　　となる。

2) n＝1でTi′は木の時

図5.17のようにノードaが1つの部分木T_1'を持つ場合である。



$$\left.\begin{aligned} st_T &= s_a \cdot st_{T_1'} \\ ct_T &= sl_{aT_1'} \cdot ct_{T_1'} \\ OCAO_T &= 1 + s_a \cdot sl_{aT_1'} \cdot OCAO_{T_1'} \end{aligned}\right\} (2)$$

図 5.17

3) aがnコの部分木T_1', T_2', …, Tn'を持つ場合

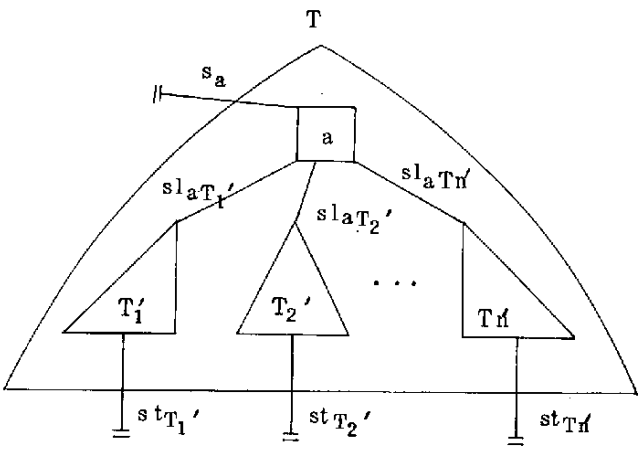


図5.18はノードaがnコの部分木T_1', … Tn'を持っていることを示している。ノードaと各部分木Ti′の根ノードとの間のリンクは結合度$sl_{aT_1'}$を持っている。

図 5.18

この時

$$\left.\begin{aligned} st_T &= sa \cdot st_{T_1'} \cdot st_{T_2'} \cdot \cdots \cdot st_{Tn'} = s_a \prod_{i=1}^{n} st_{Ti'} \\ ct_T &= sl_{aT_1'} \cdot ct_{T_1'} + s_a \cdot st_{T_1'} \{ sl_{aT_2'} \cdot ct_{T_2'} + st_{T_2'} \{ sl_{aT_3'} \cdot ct_{T_3'} + \cdots \\ &\quad + st_{Tn'-2} \{ sl_{aTn'-1} \cdot ct_{Tn'-1} + st_{Tn'-1} \{ sl_{aTn'} \cdot ct_{Tn'} \} \cdots \} \} \\ OCAO_T &= 1 + s_a \{ sl_{aT_1'} \cdot OCAO_{T_1'} + st_{T_1'} \{ sl_{aT_2'} \cdot OCAO_{T_2'} + st_{T_2'} \{ sl_{aT_3'} \\ &\quad OCAO_{T_3'} + \cdots + st_{Tn'-2} \{ sl_{aTn'-1} \cdot OCAO_{Tn'-1} + st_{Tn'-1} + \\ &\quad \{ sl_{aTn'} \cdot OCAO_{Tn'} \} \} \cdots \} \} \} \end{aligned}\right\} (3)$$

となる。(3)の中のT_1'に関するものは、(1)(2)(3)を用いて再帰的に定義される。

ある部分木Ti'は結合度$ct_{Ti'}$をもっている。よってノードaの条件を満足するオカーランスは平均して$sla_{Ti'} \cdot ct_{Ti'}$コの条件を満足するオカーランスをそのオカーランス木の中に持つことになる。ここでa内のオカーランスが条件を満足するとは、aについての制限を満足するとともに、Ti'の左手の部分木T_1', T_2', …, T_{i-1}'内の各々全てに、条件を満足するオカーランス木を持つことである。このようにa内のあるオカーランスが、条件を満足する確率は$sa \cdot st_{T_1'} \cdot st_{T_2'} \cdot \cdots \cdot st_{T_{i-1}'}$となる。よって、このオカーランスが$Ti'$内に条件を満足する木を持つ時、その期待値は、(4)式のようになる。

$$sa \cdot st_{T_1'} \cdot st_{T_2'} \cdot \cdots \cdot st_{T_{i-1}'} \cdot sla_{Ti'} \cdot ct_{Ti'} \qquad \cdots(4)$$

よって、部分木Tの結合度ct_Tは(3)のようになる。

Tのルートノード(a)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の期待値も同様にして求められる。aがカーディナリティc_nを持ち、aはキーに関する等価制限式の選択度sa'（もしなければ$sa'=1$）を持ち、この他の制限式の選択度sa''を持っているとしよう。ここで$sa = sa' \cdot sa''$である。するとT内のアクセスされるオカーランスの総数（OCA_T）は、次の様になる。

$$\begin{aligned} OCA_T &= c_a \cdot sa' \cdot OCAO_T \\ &= c_a \cdot sa' \{1 + sa'' [sla_{T_1'} \cdot OCAO_{T_1'} + st_{T_1'} \\ &\quad [sla_{T_2'} \cdot OCAO_{T_2'} + \cdots + st_{Tn-1} [sla_{Tn'} \cdot OCAO_{Tn'}] \cdots]]\} \end{aligned} \qquad (5)$$

(5)式は、次のことを意味している。図5.18において、i番目の部分木Ti'は、これより右方の部分木$T_1' \cdot T_2'$, …, $Ti'-1$によって、$sa \cdot st_{T_1'} \cdot st_{T_2'} \cdot \cdots \cdot st_{Ti'-1}$の選択度によってアクセスされるオカーランス数を減じることができる。このことは、より大きな結合度$ct_{Ti'}$をもつ部分木をより左方に配した方が、この部分木内のアクセスされるオカーランス数をより多く減じることができる。この点は後述するアクセス木生成アルゴリズム〔5.4.3〕における部分木の順序づけ（即ち、より小さな結合度をもつノードをより右方へ）への根拠となっている。

5.4.3 アクセス木 （AT）

次にDBTG問合せグラフ（DQG）からアクセスパスの生成について考えることにする。アクセスパスは、DQG内のアークとそれに結合しているノードとの対のシーケンスによって表わされる。DBTG問合せグラフ（DQG）はグラフとして表現されているために、アクセスパスは一般に木として表わされる。木内のノード又この対は、局所内部スキーマ（LIS）レベルでのアクセス単位でもある。DQGノードに対応し、枝はアークに対応している。このような木をアクセス木（access tree or AT）と呼ぶ。アクセス木（AT）を、その根ノードからpreorder〔KNUTD73〕にたどった場合の枝と、これの上位に結合されているノー

ドとの対を基本単位とするシーケンスは、アクセスパスを表わしている。DQG内のノードとアークとAT内のものとを区別するために、アクセス木内のノードをATノード、枝をAT枝と呼ぶことにする。

DBTG問合せグラフ（DQG）内のノード間のアークは、LISレベルでのアクセスパスの存在を示している。中間結果を最少とするためには、LISアクセスパスをなるべく連続的にたどることが必要である。よってDQGから生成されるアクセス木（AT）も、アークを連続して表わすものでなければならない。又、冗長なアクセスを避けるために、アークはアクセス木内でユニークな枝として表わされねばならない。DQGはグラフであるので、一般にノードは、それに結びついた複数のDQGアークを持っている。このため、アクセス木が、DQGノードに対応するAT枝をユニークに持つとDQGノードに対応するATノードを冗長に持つことになる。dをDQGノードとする。t_1，t_2，…，t_n をdに対応するATノードとする。ここで、n≧2の時、t_1，t_2，…，t_nをノードdに対する合流ノード（confluent node or CN）と呼び、各々を CN_d^1，CN_d^2，…，CN_d^n と記述する。各合流ノード CN_d^i（i＝1，2，…，n）は、各々異なったDQGアークを表わすAT枝によってアクセス木へリンクされている。よって合流ノードの存在は、次のことを意味している。

1）結果リレーション内の1の組（tuple）を得るためには、アクセス木内のオカーランスがオカーランス間の種々のリンク（e.g.セット型）を介してアクセスされる。1つの組を得るためにアクセスされるオカーランスのシーケンスを1つのアクセスパスと呼ぶことにする。この時、1つのアクセスパスにおいて、アクセス木（AT）内に合流ノードを持つDQGノード（i.e.レコード型）は、これらの合流ノードの数だけ冗長にアクセスされる。

2）あるDQGノードdのn個の合流ノードを CN_d^i（i＝1，…，n）とする。全てのアクセスパスにおいて、このパス内で条件を満足する各 CN_d^i（i＝1，…，n）内のオカーランスは同一でなければならない。

2）は、一般に各合流ノード CN_d^i で生成される中間結果リレーションの共通部分をとらねばならないことを意味している。中間結果リレーションは、1つのファイルとして管理されねばならない。このため、中間結果リレーションの数が増えれば、これらの管理を困難なものとする。さらにDBTG DMLの不完性によって、生成された中間結果リレーションをDBTG DMLによって再びDBTGモデルとしてアクセスできない。このため、親言語（e.g.COBOL）によって、結合演算、射影演算の様な集合演算機能を実現する必要がある。従って、中間結果リレーションの数（及び量）をなるべく少なくすることが重要である。

以降、グラフから木への変換アルゴリズムについて検討する。

5.4.4 横型アクセス木（AT）生成アルゴリズム（BFA）

まずグラフから横型（breadth-first）の木生成について考える。この方法は、あるDBTG問合せグラフ（DQG）ノードの全ての隣接ノードを求めこれを初めのノードの子ノー

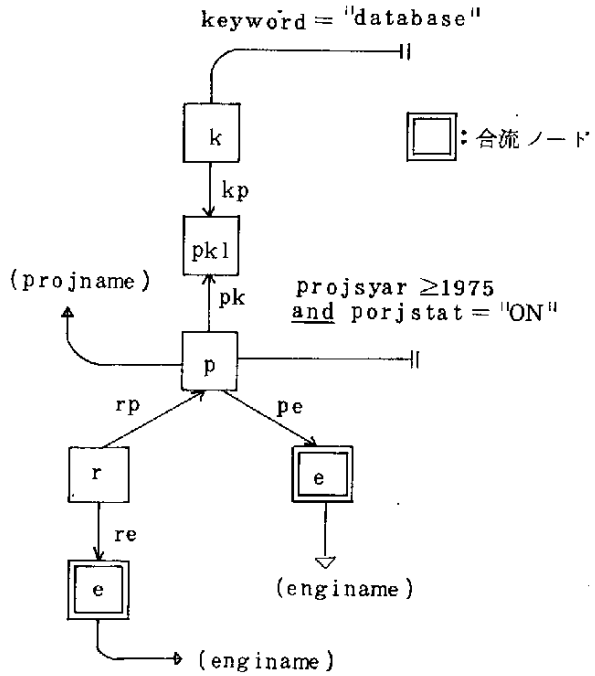


図5.19　図5.10のDQGから生成されるアクセス木〔BFA〕

ドとする。次のこの子ノードを親にして、同様にして子ノードを求める。これを木のあるレベルのノードを親として全て行なった後、次のレベルの一番左のノードを親として同様に子ノードを求めていく。子ノードは親ノードからの結合度が小さい順に並べる。

DBTG問合せグラフ（DQG）から上述のような方法で隣接ノードを調べていくと、どこかでもう既にアクセス木（AT）へ組みこまれたDQGノードに出会う。このようなノードがアクセス木内の合流ノードである。アクセス木生成アルゴリズム（breadth-first algorithm or BFA）の詳細は図5.20に示す。図5.10のDQGからノードkを開始ノードとした時のBFAによって生成されるATを図5.19に示す。

BFAによって生成されたアクセス木の特徴は、同一のDQGノードに対する合流ノード（CN）がAT内の異なった排他的な部分木に現われることである。図5.19では、合流ノードeがpの2つの異なった部分木内にある。このことは、同一のDQGノードが、部分木ごとに独立にアクセスされ、各々に中間結果を生成し、これらの部分木の根ノードにおいて、これらの中間結果の結合（join）をとる必要があることを意味している。図5.19の例では、pの左手のr－eの部分木の中間結果R_1と、右手のeの部分木の中間結果R_2とを生成し、ノードpにおいてR_1とR_2とをノードeについて結合をとらねばならない。この点は、BFAの本質的な欠点である。

しかし、複数のプロセッサを有している場合、ノードp以降のアクセスを並行して行なえる利点を持つ。現在のDBSでは可能ではないが、データベースマシン（DBM）が実現された時は、有力なアルゴリズムとなろう。

0'〔初期化〕　ＳＴＢ(x)をＤＱＧノードｘの隣接ＤＱＧノードのリストとする。

ＳＴＢ(x)内にはノードとアークとの対のリスト（y_1, a_1），（y_2, a_2），…，（y_n, a_n）がa_iの結合度が小さい順に格納されている。ここでy_iはＤＱＧノード，a_iはアークである。

ＤＱＧ内の全てのノードはＣ ｆｉｌｅｄをもち、初め０にｓｅｔされている。

ＮＯＤＥはｑｕｅｕｅであり、ＮＯＤＥ←ＮＩＬとなっている。

1〔開始ノードの決定〕

リンクレコード型に対応するもの以外の全てのＤＱＧノードのなかで、各々を開始ノードとした場合、最もアクセスされるオカーランス数の見積もり（ＯＣＡ）が小さいものを開始ノードとする。

2〔ＮＯＤＥリストへｑｕｅｕｅｉｎｇ〕

ｐｕｔ〔ｘ〕；　ｃ(ｘ)←１；

3〔ＮＯＤＥリストの先頭と取り出す〕

ｇｅｔ〔ｘ〕；

ｘ＝ＮＩＬならば、ｔｅｒｍｉｎａｔｅ。

4〔ＳＴＢ（ｘ）のアクセス〕

ｃ（ｘ）が２以上ならば３へ。この時ｘは既に合流ノードとなっている。

〔ｃ（ｘ）は１である〕

ＳＴＢ(x)がＮＩＬならば、２へ。

ＳＴＢ(x)よりｘの隣接ノードのｏｒｄｅｒｅｄ ｌｉｓｔ（y_1, y_2, …, yn）を得る。

ｉ←０；　ｊ←０；

5〔ＳＴＢ(x)の各ノードのアクセス〕

ｉ←ｉ＋１；

ｉがｎより大きければ、２へ。

6.〔y_i〕

ｃ(y_i)＝９９９９ならば、５へ。この時y_iは既にリンクレコード型としてＡＴにリンクされている。

ｃ(y_i)≧２の時、　８へ。

ｃ(y_i)＝１又は０の時、　ｊ←ｊ＋１；

ＡＴのｘのｊ番目の子としてy_iをリンクする。

ｐｕｔ〔y_i〕；

7〔y_iがリンクレコード型ならばマーク〕

y_iがリンクレコード型ならば、　ｃ(y_i)←９９９９；　５へ。

8〔ｃｏｕｎｔ ｕｐ〕

ｃ(y_i)←ｃ(y_i)＋１；　５へ。

図5.20　横型アクセス木生成アルゴリズム（ＢＦＡ）(1)

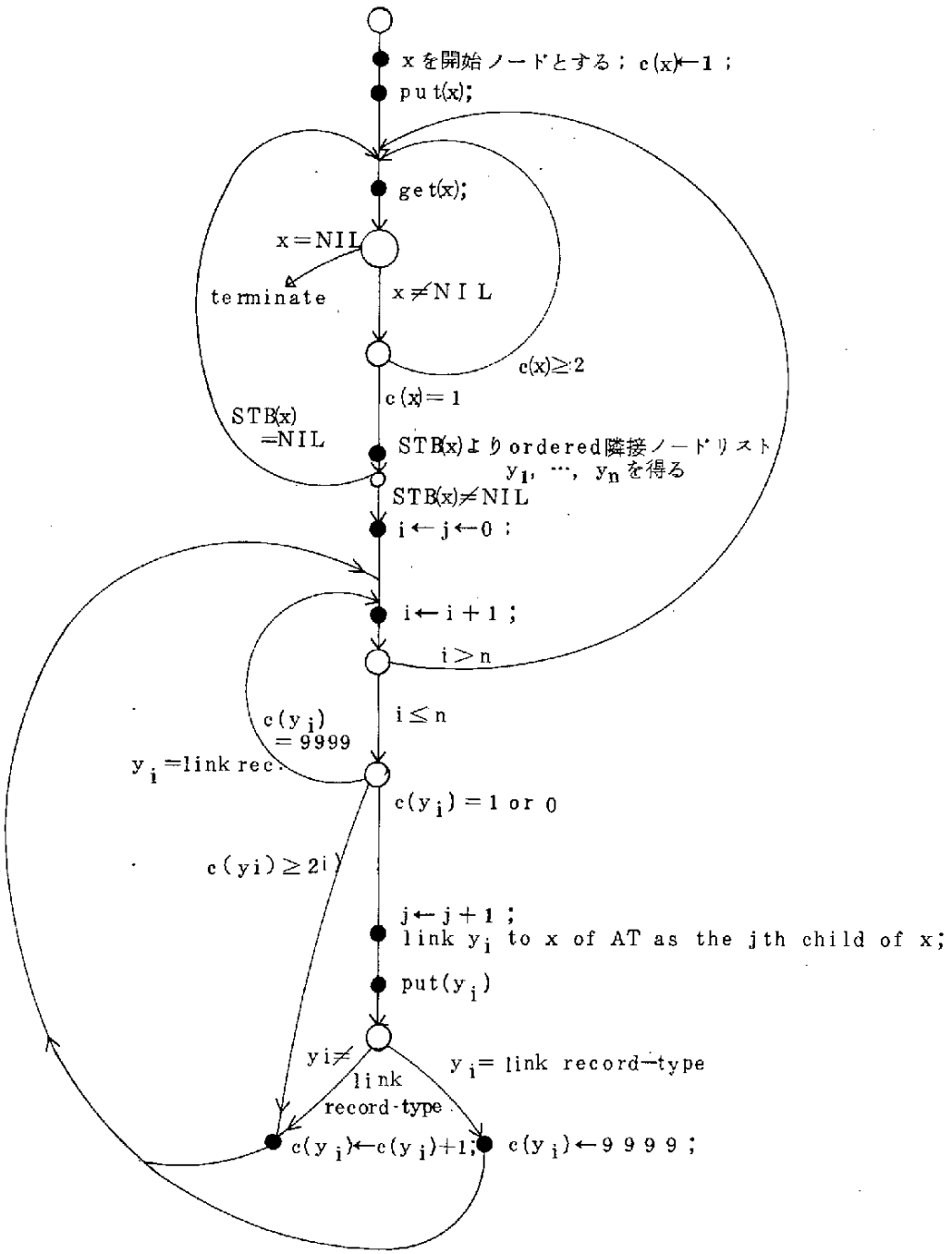


図5.20　横型アクセス木生成アルゴリズム (2)

5.4.5 縦型アクセス木（ＡＴ）生成アルゴリズム（ＤＦＡ）

縦型アクセス木生成アルゴリズム（depth-first algorithm or ＤＦＡ）はグラフのノードとアークとを縦型（depth-first）にたどって木（tree）を生成するアルゴリズムである。これは又、スパニング木（spanning tree）の縦型生成法〔ＡＨＯ　Ａ７４〕に類似している。スパニング木は、グラフ内の全てのノードをユニークに保持しており、グラフ内のリンクは一部のみを含めることができる。これに対して、我々のアクセス木は、ＤＢＴＧ間合せグラフ（ＤＱＧ）内の全てのＤＱＧアークをユニークに持ち、かつ全てのＤＱＧノードを保持している。ＤＱＧノードのあるものはアクセス木内で冗長に保持されることもある。ＤＦＡの詳細は図5.２２に示してある。

ＤＦＡの概要は次の様である。ＳＴＢ（ｘ）は、ＤＱＧノードｘの隣接ノードとアークとのリストになっている。即ちＳＴＢ（ｘ）＝（（ｙ１，ａ１）（ｙ２，ａ２），……，（ｙ2，ａ2））である。ｙｉはｘとアークａｉを介してリンクされているＤＱＧノードである。さらに、これらの要素（ｙｉ，ａｉ）は、ａｉの結合度が小さい順に並べられている。全てのＤＱＧノードは、ＮＥＷとマークされている。まず、リンクレコード型に対応するＤＱＧノード以外のＤＱＧノードから、アクセスされるオカーランス数が最少であるＤＱＧノードを、アクセス木の根ノードとして選ぶ。このＤＱＧノードをＯＬＤとマークする。ＤＱＧノードに、このＡＴノードへのポインタを保持する。ＳＴＢ（ｘ）の先頭の要素をとり出す。ａｉをＡＴ枝、ｙｉをＡＴノードとしてのアクセス木へリンクする。ＳＴＢ（ｘ）からこの要素を消去する。同時にＳＴＢ（ｙｉ）からも（ｘ，ａｉ）を消去する。ＤＱＧノードｙｉのマークがＯＬＤならば、これは合流ノードであるのでＣＯＮＦＬＵＥＮＴとマークする。対応するＡＴノードも同じくＣＯＮＦＬＵＥＮＴとマークする。さらにｙｉにポインタが保持されているＡＴノードも、ｙｉに対するものなので、ＣＯＮＦＬＵＥＮＴとマークする。ｙｉがＮＥＷならばＯＬＤとマークする。もしＳＴＢ（ｘ）が空リスト（即ち、ｘには、アクセス木へ組みこまれていない隣接ノードを１つももたない。）ならば、ＡＴノードｘはアクセス木の葉ノードとなり、ｘの親ノード（ｘ′）を調べる。ｘ′の子ノート（ｘ″）をみつける。ＤＱＧノードｘ″が、まだアクセス木にリンクされていないならば、このＤＱＧノードをｘの右隣のＡＴノードとしてアクセス木へリンクする。

上述したことを全てのＤＱＧアークをアクセス木へ組みこむまでくりかえすことによってアクセス木が生成される。

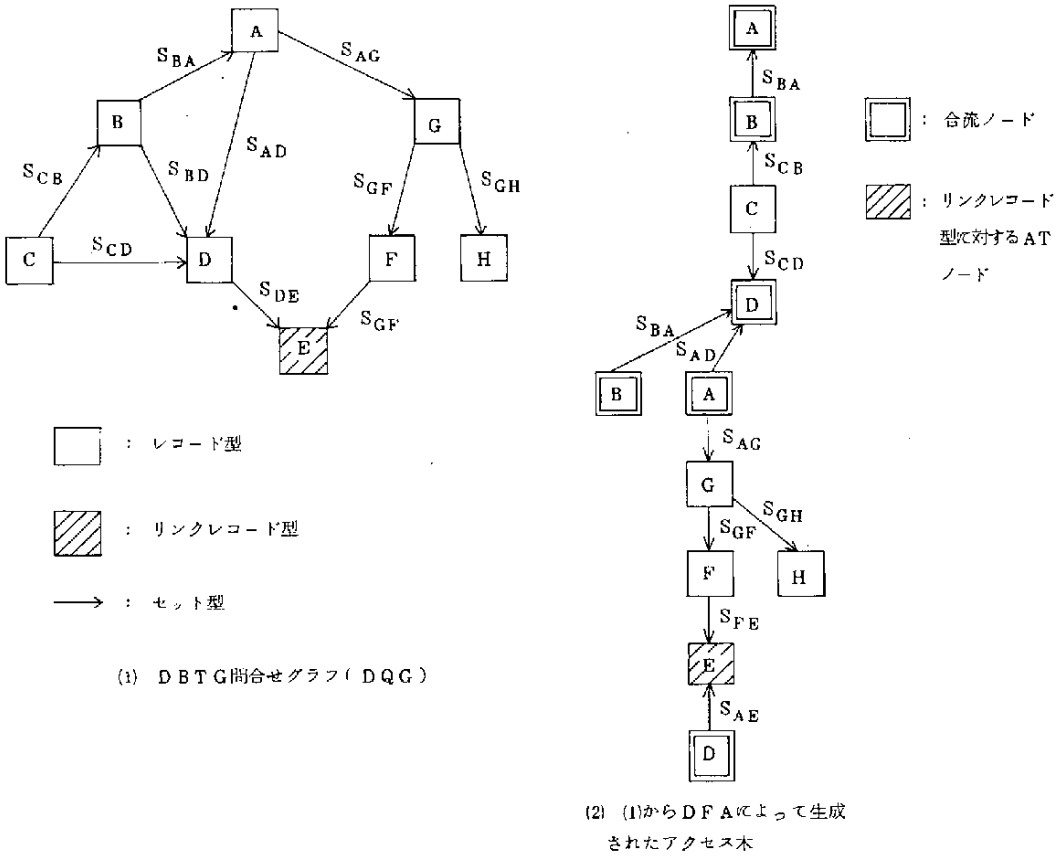


図5.2.1　アクセス木生成の例

図5.2.1は、(1)のＤＢＴＧ問合せグラフ（ＤＱＧ）から、ＤＱＧノードを、根ノードとした場合に、ＤＦＡによって生成されるアクセス木を示している。ＤＱＧノードＡは、対応するＡＴノードがアクセス木内に２個、ＤＱＧノードＢは２個、そしてＤＱＧノードＤは２個現れているので、この３つのＤＱＧノードは合流ノードとなる。

0〔初期化〕　ＳＴＢ（ｘ）を、ＤＱＧノードｘの隣接ノードリストとする。ＳＴＢ（ｘ）内には、ノードとアークとの対（y_1，a_1），……，（y_m，a_n）が、ＯＣＡy_i（ｉ＝１，２，…，ｎ）が小さい順にならんでいる。ここでy_iはｘの隣接ＤＱＧノード、a_iはｘとy_iの間のアーク（即ちセット型）を示している。ＯＣＡが同一の時は、ＯＣＳが小さい順に並べておく。即ち同一のＯＣＡを持つノードのうち、より小さな選択度を持っているものをより前におく。

ＤＱＧ内の全てのノードをＮＥＷとマークする。

push doun（A）；

〔ここで　ｘ，y_1，……，y_nはＤＱＧノードを表すとする。αをＤＱＧノード又はアークとすると、これに対応するＡＴノードとＡＴ枝は各々ＡＴ(α)と表わされる。〕

1.〔根ＡＴノードの決定〕

ＤＱＧ内のリンクレコード型に対応するもの以外の全てのノードの中で、ＯＣＡが一番小さいものを開始ノード、即ちアクセス木（ＡＴ）の根ノードとする。

もし、ＯＣＡが同一であれば、条件を満足するオーカランス数（ＯＣＳ）が一番少ないものを選ぶ。

ＤＱＧ内のノードを任意に番号づけ、n_iをｉ番目のノードとする。ｍを全ノード数とする。c_iをn_iのカーディナリティ、s_iをn_iのＣＡＬＣに関する等価制限式の選択度（なければ$s_i=1$）とする。開始ノードは$c_i \cdot s_i$が最少のものである。

2.〔ノードｘのマーク〕

push down（ｘ）；

ＤＱＧノードｘがＮＥＷとマークされていれば、ＤＱＧノードｘもＯＬＤとマークする。

ＤＱＧノードｘをＯＬＤとマークするとともに、これに対応するＡＴノードｘへのポインタをＤＱＧノードｘ内に格納する。

3.〔ＳＴＢのチェック〕

ＳＴＢ（ｘ）が空ならば、７へ。

（即ち、ＤＱＧノードｘは、ＡＴ枝としてアクセス木へ組み込まれた隣接ＤＱＧアークを１つも持っていない）

4.〔ＳＴＢ（ｘ）の先頭要素（y_i，a_i）を取り出す〕

ＳＴＢ（ｘ）の先頭要素（y_i，a_i）を取り出す。

ＤＱＧノードy_iとアークa_iに対応する。各々ＡＴノードと枝とをＡＴノードｘの子ＡＴノードとしてアクセス木へリンクする。

ＡＴノードｘが既に他の子ＡＴノードを持っていれば、y_iを一番右側の子ＡＴノードとしてアクセス木へリンクする。

図5.2.2　縦型アクセス木生成アルゴリズム（ＤＦＡ）(1)

5.〔ＤＢＴＧ問合せグラフ（ＤＱＧ）から、ＤＱＧアーク a_1 の除去〕

ＳＴＢ（ｘ）から、要素（y_1, a_1）を除去する。

同時にＳＴＢ（y_1）から、要素（x_1, a_1）を除去する。

6.〔ＤＱＧノードとＡＴノード y_1 のマーク〕

ＤＱＧノード y_1 がＯＬＤとする。

この時ＤＱＧノード y_1 をＣＯＮＦＬＵＥＮＴとマークする。ＡＴノード y_1 も、ＣＯＮＦＬＵＥＮＴとマークする。同時ＤＱＧノード y_1 内に格納されているポインタを用いて、最初にこのＤＱＧノード y_1 を参照したＡＴノードもCONFLUENTとマークする。 $x \leftarrow y_1$ として3へ。

7.〔ＳＴＢ（ｘ）が空の時〕

ｐｏｐｕｐ〔ｘ〕；

ｘが Λ ならば、アクセス木生成を終了する。

ｘが Λ でなければ3へ。

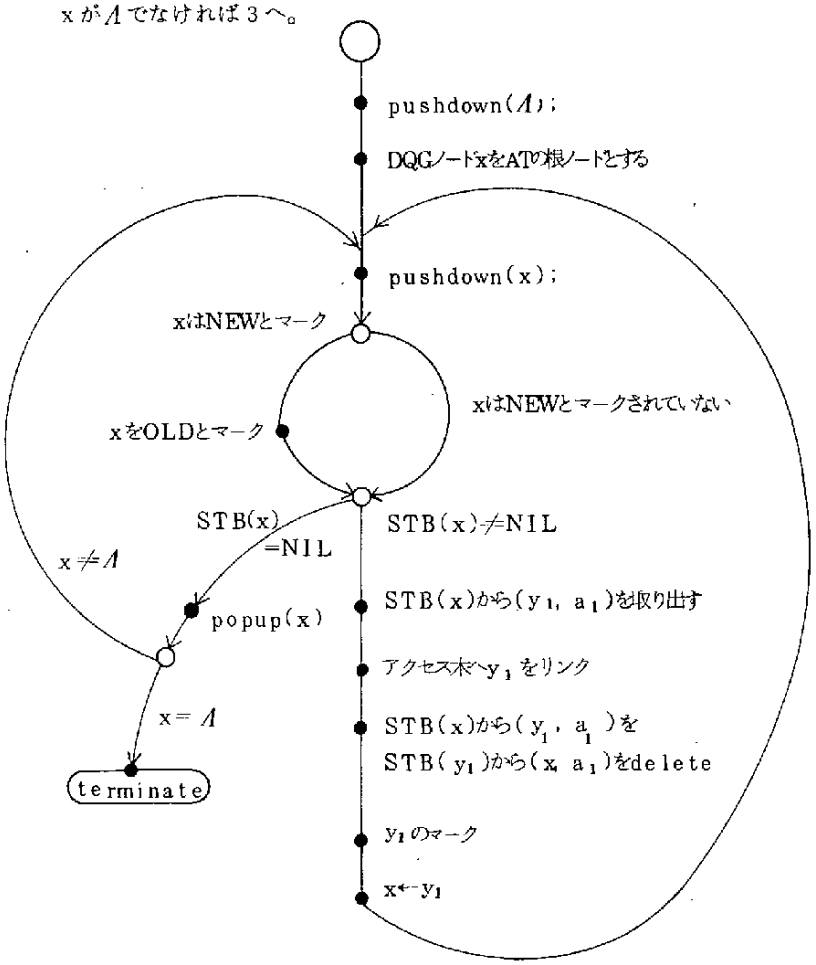


図 5.2.2　縦型アクセス木生成アルゴリズム（ＤＦＡ）(2)

あるアクセス木が形成された時、このなかのアクセスされるオカーランス数の期待値（OCA）は、5.4.2項のCで論じた結果を用いて計算できる。あるDBTG問合せグラフ（DQG）から生成可能な全てのアクセス木について、各々アクセスされるオカーランス数の期待値を求め、最も少ないアクセス木構造を得ることによって、アクセス時間について最適なアクセス木をもとめられる。しかし、これは多くの計算処理を必要としてしまう。このため、次の様なヒューリスティックスを用いた。即ち、次の子ATノードの決定は、可能なDQGノードのなかから、最もアクセスされるオカーランス数が小さいと思われるものを選ぶことによってなされる。このヒューリスティックは単純であるが、有効に不要なアクセス空間を縮小できる。

5.4.2項の式(4)が示す様に、あるノードのより右手の部分木へのアクセスは、左手の部分木群によって、このなかのアクセスされるオカーランス数を制限し得る。このため、アクセスされるオカーランス数が大きい部分木を、より右手におくことによって、この部分木内のアクセスされるオカーランス数を減少できる。よって我々のこの戦術は簡単であるが有効であるということができる。

リンクレコード型に対応するDQGノードは、アクセス木内の合流ノードとはならず、アクセス木内の単なる中間ノードとなる。リンクレコード型は、一般に、Rのオカーランスをユニークに識別できるデータ項目を持たない。同一のDQGノードに対応する合流ノードは、1つのアクセスパス内で同一のオカーランスがアクセスされねばならない。我々の手法では、オカーランスの同一性のチェックは、合流ノードのDQGノードの主キー属性の値を比較することによって行なっている。このため、リンクレコード型を合流ノードとすると、この比較を行なえなくなる。このためリンクレコード型のDQGノードを合流ノードとはしない。リンクレコード型は、一般に制限（ restriction ）を持たず、かつ2つのレコード型間のn：mの関係性を表わしているために根ノードとした時のアクセスされるオカーランス数はこれらのリンクレコード型よりも多いと考えられる。このため、多くの場合、リンクレコード型はアクセス木の根ノードとはならない。

縦型アクセス木生成アルゴリズム（DFA）によって生成されたアクセス木の特徴は、横型アクセス木生成アルゴリズム（BFA）のような合流ノードの処理のための多くの中間結果とこれらの処理のための集合演算を必要としない点である。この点は、我々のDFAの最も重要な長所である。これは、あるDQGノードに対する全ての合流ノードは、アクセス木内の最初の合流ノードを根ノードとする部分木内にふくまれることによっている。これは次の様にして示される。

あるDQGノードdのアクセス木内の合流ノードをCN_d^1, CN_d^2, …, CN_d^mとする。

ATノードR_0を根ノードとするアクセス木をpreorderでだとった場合のATノードのシーケンスは、R_0, …, CN_d^1, …CN_d^2…, CN_d^m であったとしよう。この時、CN_d^2,CN_d^3…, CN_d^mはCN_d^1を根ノードとする部分木内に全てふくまれることを示す。

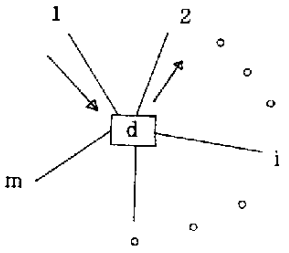


図5.23

ＤＱＧノードｄは図5.23のように、ｍ本のＤＱＧアークによって、ＤＢＴＧ問合せグラフに結合されているとしよう。ＤＦＡによって、ある時、１番目のアークを介してＤＱＧノードｄがアクセス木へ組みこまれたとする。ついで２番目のアークを通って、ｄの次のノードをたどりにいったとする。アーク２から生成され得るアクセス部分木が全てつくられると、DFAではＤＱＧノードｄへもどってくる。ここで、ｄに結合したリンクのなかでまだアクセス木のＡＴ枝としてアクセス木へ組み込まれていないものがあると、それを用いてアクセス部分木の生成を行なう。このようにしてｄからでているアークが全てアクセス木へ組みこまれるまで以上のことをくりかえす。ＤＱＧノードｄがあるループの中にあるとする。たとえばｄ→アーク２→……→アークｉ→ｄというループがあるとすると、アーク２から生成される部分木の葉のＡＴノードとしてｄがリンクされる。このｄは、合流ノードとなる。ループがある時は、ＤＱＧノードは、これから生成される部分木内に合流ノードを持つことになる。この時さらにこの合流ノードを根ノードとして、まだアクセス木へ組みこまれていないアークを用いて部分木をつくっていく。この様に、ＤＱＧノードｄの合流ノードはアクセス木生成ではじめにｄに対して生成されたＡＴノードを根ノードとする部分木の中に全てふくまれることになる。

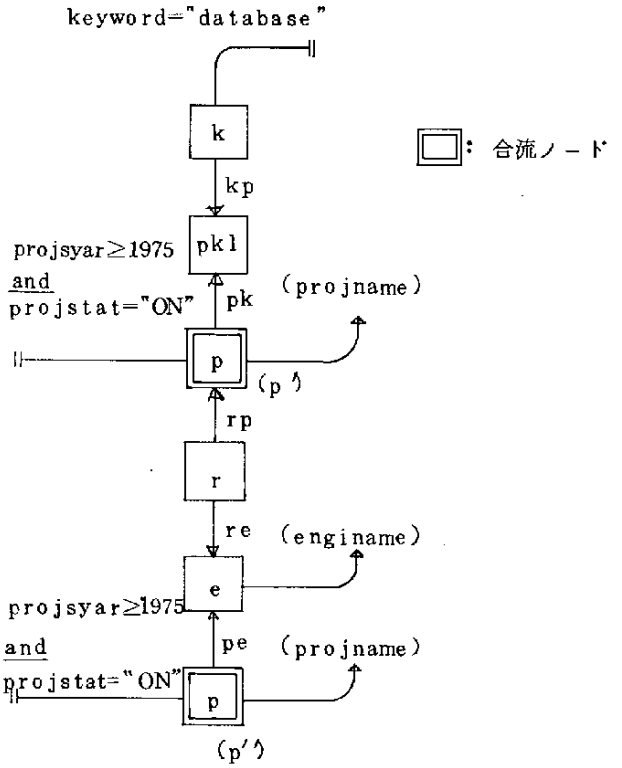


図5.24　図5.10から生成されるアクセス木〔ＤＦＡ〕

このことは、第２番目以降の合流ノードをアクセスする際に、第１の合流ノードのオカーランスの値と、今の合流ノードの値を比較し、同じものならばアクセスを続け、異なっていればアクセスをそこで打切ればよい。例えば、図5.24の例では、ＤＱＧノードｐについて２つの合流ノードが存在している。便宜上第１番目の合流ノードをｐ′、第２番目をｐ″と呼ぶことにする。ｐ′のアクセスの時、あるオカーランスが、この条件（ｐｒｏｊｓｙａｒ≥１９７５ａｎｄ ｐｒｏｊｓｔａ＝"ＯＮ"）を満足する

時、このオカーランスのキー属性projname（同時にこれは結果属性でもある）の値を結果リレーションに出力する。このオカーランスからリンクをたどってp″のアクセスをする時、p″のオカーランスのprojnameの値をもとめ、最初のpのprojnameの値と比較する。もし同じならば、このオカーランスは求めるものであり、異なればこのアクセスは失敗であり次のアクセスを考える。この様に、結果リレーション内に、AT内の合流ノードのキー属性を出力するための属性を設けるだけで、何等のこの他の中間結果もいらないことになる。第2番目以降の合流ノードは、その主キーの値を結果リレーション内のものと比較するだけでよい。このDFAによって生成されたアクセス木のこの性質は重要な長所である。

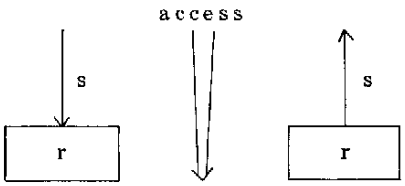
5.5 DMLの生成（DMLG）

アクセス木（AT）のATノードは、事象集合（ES）に対応する局所内部スキーマ（LIS）要素（e.g. DBTGモデルのレコード型、IMSモデルのセグメント型）を表わし、枝は事象集合（ES）間の関係性集合（RS）に対応するLIS要素（e.g. DBTGモデルのセット型、IMSモデルの階層パス）を表わしている。各データベースシステム（DBS）のLISレベルでのアクセスの基本単位は関係性集合（RS）とこれと関連した事象集合（ES）とに対応する各々の要素の対から成っている。例えばDBTGでは、セット型とこれに結びついたレコード型との対が、アクセスの基本単位となる。アクセスシーケンスは、よって、これらのアクセス単位（access unit or AU）のシーケンスとして表わせる。

5.5.1 アクセス木からアクセスシーケンスの生成

アクセス木（AT）から生成されるアクセスパスは、AT枝とATノードとをpreorder〔KNUTD73〕によってたどった場合のAT枝とこれに結ばれたATノードとの対のシーケンスである。例えば図5.19に示したアクセス木からは、アクセスシーケンス（（∧, k）(kp, pkl)　(pk,p)　(rp,r)　(re,e^{*})　(pe,e^{*})）が得られる。シーケンス内の各対の第1要素はセット型を第2要素はレコード型を表わしている。＊のついた変数は、これが合流ノードであることを示している。これの第1要素の∧は、セット型を介してアクセスされないことを示している。又、図5.24のアクセス木からは（（∧, k）（kp, pkl）（pk, p^{*}）（rp, r）（re,e）（pe, p^{*}））がアクセスシーケンスとして生成される。

シーケンス内のセット型とレコード型との対をアクセス単位（access unit or AU）と呼ぶ。アクセス単位は大別して2種類のものがある。アクセス単位を（s, r）と記すことにする。ここでsをセット型、rをsに結びついたレコード型とする。図5.25に示す様に、レコード型rは、セット型sを介してアクセスされる。レコード型rがセット型sの子レコード型の時、このアクセス単位（AU）をNEXTアクセス単位（NAU）と呼ぶ。又、rがsの親レコード型の時、このアクセス単位を親アクセス単位（OAU）と呼ぶ。



(1) nextアクセス単位(NAU)　　(2) ownerアクセス単位(OAU)

図5.25　2種類のアクセス単位(AU)

即ち、NEXTアクセス単位はアクシーケンス内のこれより1つ前のアクセス単位とは、1：Nの関係があることを示し、親アクセス単位は1：1の関係性を示している。

1つのセット型sのオカーランスによってリンクされたレコード型r内のオカーランスを、1つのアクセスリンク(access link)内にあるという。NEXTアクセス単位のアクセスリンク内にはn個のレコード型rのオカーランスがあり(n≧1)、親アクセス単位には、1つのオカーランスがある。

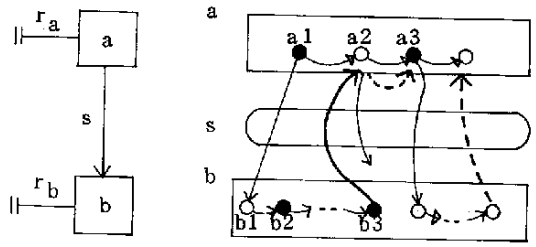
横型アクセス木生成(BFA)においては、あるDQGノードdに対する合流ノードは、DFAの様に同一の部分木内には存在しない場合がある。例として、図5.19を考えてみよう。DQGノードeに対するATノードの2つは合流ノードであり、互いに異なった部分木内にある。ATノードpの右手の部分木をT1、左手をT2と便宜上呼ぶことにする。T1でアクセスされ、T1の条件を満足するATノードとのオカーランスと、T2内のATノードとのオカーランスとは等しくなければならない。このためには、T1で生成された結果リレーションとT2で生成された結果リレーションとを、ATノードeのオカーランスについて等価結合(equi-join)を行なう必要がある。この演算を、先に述べたアクセスシーケンスに加えねばならない。よって、((∧, k) (kp, kpl) (pk, p)((rp, r) (re, e))(pe, e)J(e))をアクセスシーケンスとして得れる。ここでJ(e)は((rp, r) (re, e))のアクセス結果と、(pe, e)のアクセス結果との結合演算を行なうことを示している。

これに対して、縦型アクセス木生成(DFA)では、あるDQGノードに対する全ての合流ノードは同一の部分木内にあるので、BFAの様な結合演算を必要としない。

5.5.2 オカーランスレベルのシーケンス

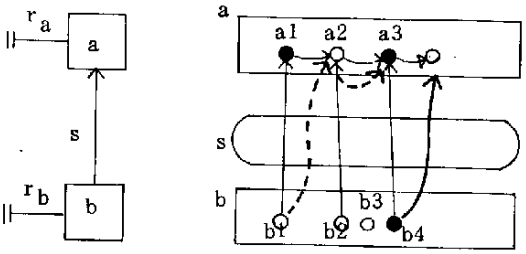
各レコード型は複数のオカーランスから構成され、これらの各々はセット型を介して他のレコード型のオカーランスとリンクされている。この様なオカーランスレベルのリンクでつくられた木をオカーランス木(occurrence tree)と呼ぶ。DBTGスキーマ上のアクセスは、このようなオカーランス木をたどっていくことである。本項では、オカーランス木のたどり(traverse)について考える。

A. 基 本 単 位



(a) アクセス木　　(b) オカーランス木

(1) (s, b)はNAU



(a) アクセス木　　(b) オカーランス木

(2) (s, b)はOAU

図 5.2 6　基本単位内のオカーランス

●：条件を満足するオカーランス

○：条件を満足しないオカーランス

r_a：レコード型aについての制限式

r_b：レコード型bについての制限式

⟶：成功

⇢：失敗

まず、アクセス単位内のオカーランスのリンクについて考えてみよう。図5.2 6は、レコード型aにアクセス単位（ＡＵ）(s, b)がリンクされている例である。(1)は、(s, b)がＮＥＸＴアクセス単位（ＮＡＵ）である場合を示している。図内で黒丸はそのレコード型についての条件、a内のオカーランス(a1)が即ち、制限式を満足するオカーランスを示し、白丸はそうでないものを示している。制限式 r_aを満足するならば、セット型sを介してリンクされているレコード型b内のn個(n≧1) のオカーランスをアクセスにいく。このn個内に条件式 (r_b)を満足するオカーランス (b2 と b3) がある時、(a1, b2), (a_1, b3) の２つの対は求める解となり、このリンク内のアクセスは成功したことになる(⟶)。次いで、レコード型aの a_1 の次のオカーランス a_2 がアクセスされる。これをレコード型aへの成功復帰と呼ぶ。b内のリンクされたオカーランスが１つも満足しなければこのリンクのアクセスは失敗したことになる。もちろん、aのオカーランス(a2)が条件 r_a を満足しなければ、この時も失敗となる。この２つの場合は図の太い点線(⇢)によって示してある。これを、レコード型bへの失敗復帰と呼ぶ。

図5.2 6の(2)は、(s, b)が親アクセス単位（ＯＡＵ）の場合を示している。レコード型a内の１つのオカーランスは、どんなに多くても、b内の１つのオカーランスとリンクされているだけである。図5.2 6の(2)内で条件を満足する解としては、(a3, b4) だけである。

B．アクセス単位（ＡＵ）のチェイン

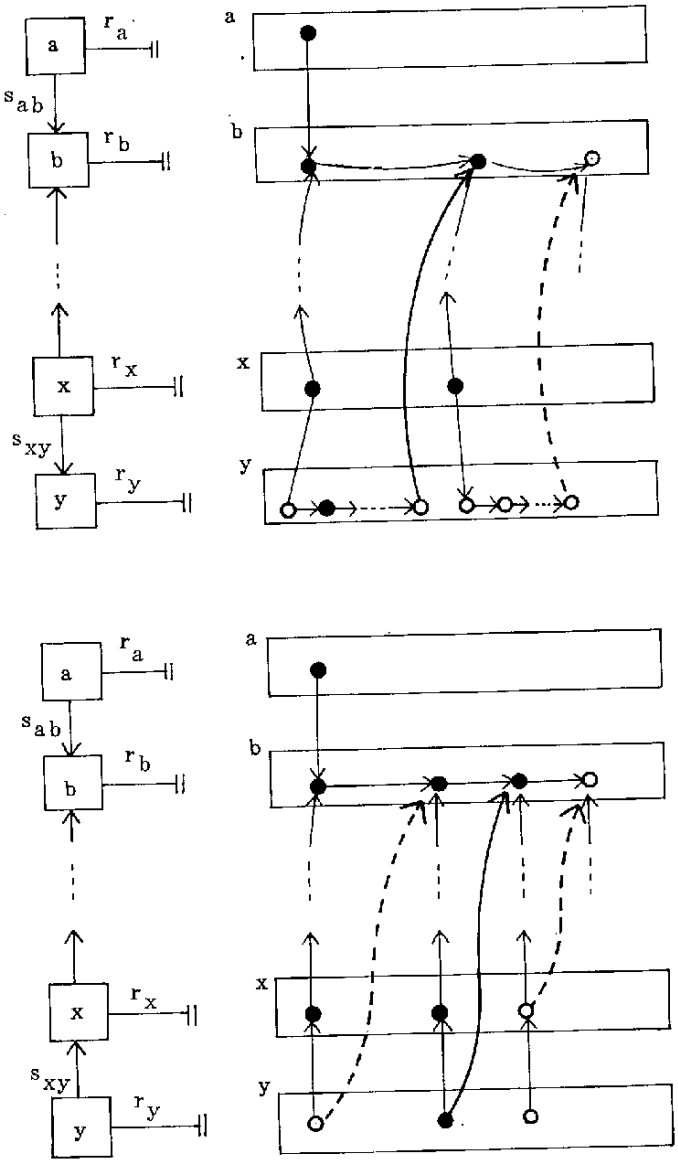


図5.27　アクセス単位のチェイン

次にアクセス単位が線型にチェインされている場合を考えてみよう。図5.27は（s_{ab}，b）がＮＥＸＴアクセス単位（ＮＡＵ）であり、以降ノードｘまでが親アクセス単位（ＯＡＵ）である場合を示している。(1)は最後のアクセス単位（s_{xy}，x）がＮＥＸＴアクセス単位で、(2)は親アクセス単位である場合を表わしている。Ａの時と同様に黒丸は条件を満足するオカーランスを、白丸はそうでないものを示している。太い線は、アクセスが成功した場合を示し、太い点線は失敗した場合を示している。この様に、あるアクセス単位（ＮＥＸＴでも親でも）内の一連のリンクされたオカーランスのアクセスが成功か失敗かによって終了した時、このアクセス単位の以前にあらわれた最後のＮＥＸＴアクセス単位内の次のオカーランスのアクセスをすることがわかる。このような最後に現われたＮＥＸＴアクセス単位をLast-appeared ＮＡＵ（ＬＮＡＵ）と呼ぶ。

C、アクセス単位（ＡＵ）の木

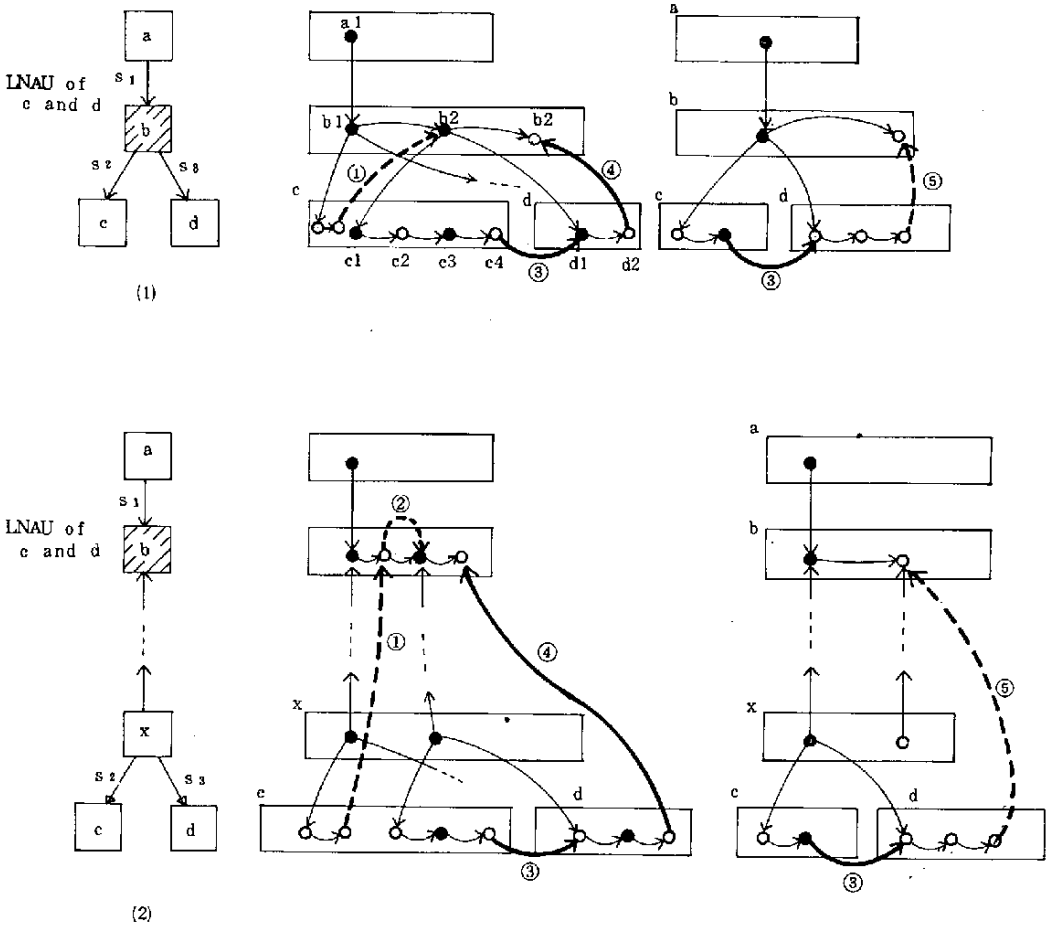


図５.２８　アクセス単位の木

次にアクセス単位が木を形成している場合を考えてみよう。図５.２８は、（ s_2, c ）（ s_3, d）を子ノードとして持つアクセス単位が存在する場合をしめしている。(1)と(2)では、ともにレコード型ｂがｃとｄにとってのＬＮＡＵ（ last−appeared ＮＡＵ）である。

この図より次のことがわかる。

(1) あるアクセス単位（ＡＵ）内で、アクセスに失敗した時（失敗復帰）には、このＡＵより上位のＡＵの中で最後に現われたＮＡＵ（ＬＮＡＵ）内の次のオカーランスへアクセスは帰る。図5.28の(1)と(2)をみよ（①と⑤はこの失敗復帰を示している）。

(2) あるアクセス単位内でアクセスに成功した時（成功復帰）には、もし右隣にこのアクセス単位の兄弟のアクセス単位があれば（ｃをみよ）、このアクセス単位内の同一の親オカーランスを持つオカーランスの先頭をアクセスする。

（③はこの成功復帰を示している。）

(3) あるアクセス単位内で、アクセス成功した時（成功復帰）、もし右隣にこのアクセス単位の兄弟のアクセス単位が何もなければ（ｄを見よ）、ＬＮＡＵ内の次のオカーランスへアクセスは帰る。（④はこれを示している。）

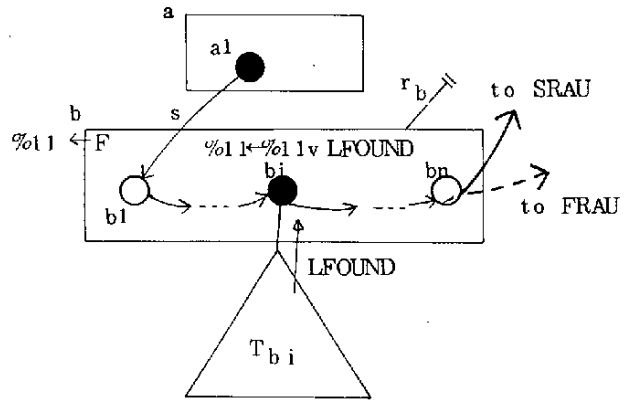
(4) あるアクセス単位で、あるオカーランスがこのアクセス単位の条件を満足しなければ、次のオカーランスをアクセスする（②はこれを示している）。

(1) の場合のアクセスが帰るアクセス単位を失敗復帰アクセス単位（ｆａｉｌｕｒｅ－ｒｅｔｕｒｎ ＡＵ ｏｒ ＦＲＡＵ）と呼ぶ。

(2) 及び (3) の場合のアクセスの帰るアクセス単位を成功復帰アクセス単位（success－return ＡＵ ｏｒ ＳＲＡＵ）と呼ぶ。

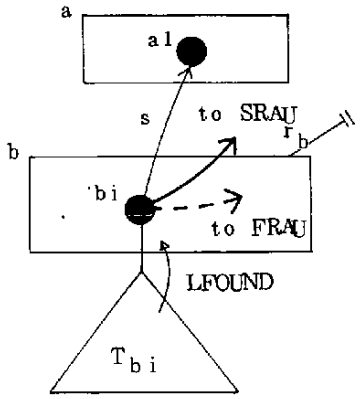
D. アクセスリンクの成功と失敗

アクセス木（ＡＴ）の中間ノードに対応するアクセス単位（ＡＵ）について考えてみる。アクセス単位のあるアクセスリンク内の１つのオカーランスは、一般に下方にオカーランスの部分木をもっている。このオカーランスへのアクセスが成功であるとは、このオカーランスの属するレコード型に関する制限式を満足するとともに、これのより下位のオカーランスの部分木のアクセスが成功であることを意味している。図5.28は、アクセス単位（ｓ，ｂ）（ここでｓはセット型、ｂはレコード型）内のあるアクセスリンク（ＡＬ）内のオカーランスｂｉについて示している。ｂｉは、オカーランスの部分木T_{bi}を持っているとする（ｉ＝1, 2,…, ｎ）。



(1) （s, b）はNAU

Ｔｂｉへアクセスに行くためには、まずｂｉは制限式ｒｂを満足せねばならない。Ｔｂｉのアクセスの結果は変数ＬＦＯＵＮＤにセットされてｂｉに帰ってくるとする。ＬＦＯＵＮＤがＴならばＴｂｉのアクセスは成功であり、Ｆならば失敗である。ｂが親アクセス単位ならば、ＬＦＯＵＮＤの値がＴならば成功復帰をし、Ｆならば失敗復



(2) (s, b)はOAU

図5.29　アクセスの成功と失敗

帰することになる。bがＮＥＸＴアクセス単位の場合には、アクセスリンクの最後のオカーランスbnにおいて復帰先の決定がなされる。biからbnのアクセスリンク内のどれかのオカーランスbiが制限式を満足し、かつT_{bi}のＬＦＯＵＮＤの値がＴであるものがあれば成功復帰をし、そうでなければ失敗復帰することになる。これは次のようにしてなされる。まず、アクセスリンクの最初のオカーランスbi へアクセスにはいる時に変数％１１にＦをセットする。bi が制限式r_bを満足しない時は、次のオカーランス(bih)をアクセスする。bi が制限式を満足すれば、T_{bi}へのアクセスにいく。Tbiのアクセスの後に、％１１とＬＦＯＵＮＤの論理和をとり、この結果を％１１に代入する。最後のオカーランスのアクセス終了時に％１１がＴならば成功復帰をＦならば失敗復帰をすることになる。このことを図5.30に示す。図中のＬＦＯＵＮＤと％１１は図5.34内に示した変数に対応している。

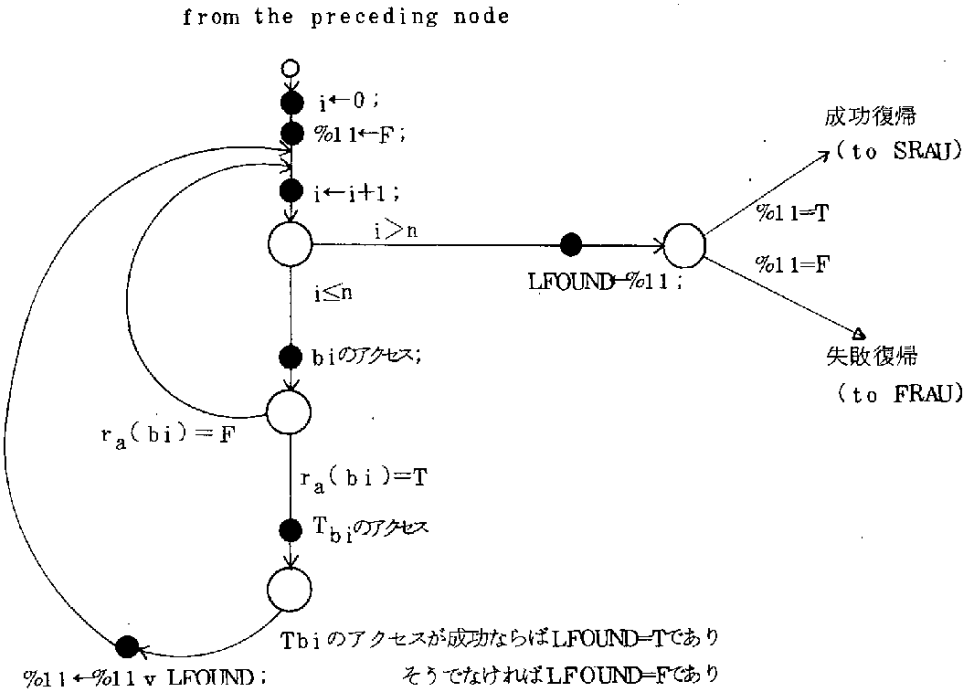


図5.30　アクセスリンク内のアクセス

5.5.3 アクセス単位の記述

以上のことより、アクセス単位を記述するためには、次の情報が必要になる。

1）セット型

2）レコード型及び親レコード型か子レコード型かの情報

3）制限式（ＣＡＬＣを参照する等価制限式とこの他の制限式）

4）結果属性リスト

5）合流ノードかどうか

6）開始ノードか中間ノードか葉ノードか

7）成功及び失敗の復帰先のＡＵ

アクセス単位の形式的記述を、図5.31に示す

```
AU::=(<unit-no>
        (SET <set-type>)
        (REC <record-type>
         [(RSLT <result-att-list>)]
         [(CEQR <calc-eq-restr>)]
         [(RSTR <non-calc-eq-restr>)]
        (UTYPE <set-rec-type><confluent-mode>
                                <AT-node-type>)
        (FRTRN <unit-no>)
        (SRTRN <unit-no>) )
  <set-rec-type>::=OWNER | MEMBER
<confluent-mode>::=CONFLUENT | NORMAL
 <AT-node-type>::=ROOT | INTERMEDIATE | LEAF
```

図5.31　アクセス単位（ＡＵ）の形式的記述

<unit-no> は、アクセス単位の識別番号であり、ＡＴから生成されるアクセスシーケンス内の順番を示している。<set-type> と<record-type> は、各々セット型とレコード型とを示している。レコード型かこのセット型の親である時、<set-rec-type>は、ＯＷＮＥＲに、子の時はＭＥＭＢＥＲになる。<result-att-list>は、結果属性リストを表わしている。ＣＥＱＲとＲＳＴＲは、このレコード型に関する制限式を示している。前者はＣＡＬＣを参照する等価制限式を、後者はこれ以外のものを示している。レコード型に関する制限式は、<calc-eq-restr>and <non-calc-eq-restr> となる。ともに対応する制限式がない時は、ＮＩＬ又は省略可である。このアクセス単位が合流ノード（ＣＮ）の時<confluent-mode>はＣＯＮＦＬＵＥＮＴに、そうでない時はＮＯＲＭＡＬとなる。

＜AT－node－type＞は、このアクセス単位がアクセス木内で根ノード（ＲＯＯＴ）、中間ノード（ＩＮＴＥＲＭＥＤＩＡＴＥ）、又は葉ノード（ＬＥＡＦ）かを示している。ＦＲＴＲＮとＳＲＴＲＮは、各々、失敗復帰アクセス単位と成功復帰アクセス単位の番号を示している。ＦＲＴＲＮとしては、このアクセス単位より上位のＮＡＵの中で最も近いもののアクセス単位番号がはいる。ＳＲＴＲＮとしては、このアクセス単位が右隣に兄弟のアクセス単位を持てばそのＡＵ番号であり、なければ、ＦＲＴＲＮと同じＡＵ番号がはいる。図5.３２は、図5.２３のアクセス木の記述を示している。

```
((1 (SET NIL)
    (REC KEYWORDS (CEQR (EQ KEYWORD (QUOTE DATABA
     SE))))
    (UTYPE NIL NORMAL ROOT)
    (FRTRN TERM)(SRTRN TERM))
 (2 (SET KEYW-PROJ)
    (REC KEYW-PROJ-LINK)
    (UTYPE MEMBER NORMAL INTERM)
    (FRTRN 1)(SRTRN 1))
 (3 (SET PROJ-KEYW)
    (REC PROJECT (RSLT(PROJNAME))
        (RSTR (AND (EQ RPOJSTAT (QUOTE ON))
                   (GE PROJSYAR 1975))))
    (UTYPE OWNER CONFLUENT INTERM)
    (FRTRN 2)(SRTRN 2))
 (4 (SET REPR-PROJ)
    (REC REPRESENTATIVE)(UTYPE OWNER NORMAL INTERM)
    (FRTRN 2)(SRTRN 2))
 (5 (SET REPR-ENGI)
    (REC ENGINEER (RSLT(ENGINAME))
    (UTYPE MEMBER NORMAL INTERM)
    (FRTRN 2)(SRTRN 2))
 (6 (SET PROJ-ENGI)
    (REC PROJECT(RSLT(PROJNAME))
        (RSTR (AND (EQ PROJSTAT(QUOTE ON))
                   (GE PROJSYAR 1975))))
    (UTYPE OWNER CONFLUENT LEAF)
    (FRTRN 5)(SRTRN 5)))
```

図5.３２　図5.２３の記述

5.5.4 ＤＭＬブロックの生成

各アクセス単位（ＡＵ）に対応して１つのＤＭＬのブロックを生成する。ＤＭＬブロックのシーケンスは、アクセスシーケンス内の対応するアクセス単位のシーケンスに等しい。アクセス単位は、図5.３３に示すような10個のパターンに分類でき、各パターンは対応するＤＭＬブロック記述をもっている。アクセスシーケンス内の各アクセス単位は、このパターンとの照合が行なわれ、照合すれば対応するＤＭＬブロックが出力される。

ＡＵのクラスは、ＦＩＲＳＴ，ＣＮ－ＦＩＲＳＴ，ＣＡＬＣ，ＯＷＮＥＲ，ＮＥＸＴ，ＣＮ－ＮＥＸＴ，ＬＡＳＴ－ＯＷＮＥＲ，ＬＡＳＴ－ＮＥＸＴ，Ｉ－ＦＩＲＳＴ，Ｉ－ＣＡＬＣの10個である。ＦＩＲＳＴ，ＣＮ－ＦＩＲＳＴ及びＣＡＬＣはＡＴの開始ノードに対応するアクセス単位（ＡＵ）である。ＣＡＬＣ－ＡＵは、ＣＡＬＣ項目を参照する等価制限式を保有するものであり、制限式内のＣＡＬＣ値を用いて直接に必要なオカーランスをアクセスできるものである。ＦＩＲＳＴ－ＡＵとＣＮ－ＦＩＲＳＴ－ＡＵとは、ＣＡＬＣ項目を参照する等価制限を持たないのであり、このＡＵ内のレコード型のオカーランスをシーケンシャルにアクセスせねばならないものである。ＣＮ－ＦＩＲＳＴは、さらに、合流ノードでもあることを示している。ＯＷＮＥＲ，ＮＥＸＴ及びＣＮ－ＮＥＸＴ－ＡＵは、ＡＴ内の中間ノードに対応するもので、各々ＯＡＵ，ＮＡＵ，ＮＡＵである。ＣＮ－ＮＥＸＴはこれが合流ノードに対応することを示している。ＬＡＳＴ－ＯＷＮＥＲとＬＡＳＴ－ＮＥＸＴ－ＡＵは各々ＡＴ内の葉ノードに対応するＯＡＵとＮＡＵである。Ｉ－ＦＩＲＳＴとＩ－ＣＡＬＣは、アクセスシーケンスがただ１つのＡＵから成っているものである。前者はＦＩＲＳＴ－ＡＵに、後者はＣＡＬＣ－ＡＵに対応している。アクセス単位（ＡＵ）の分類手順は図5.３３に示す。

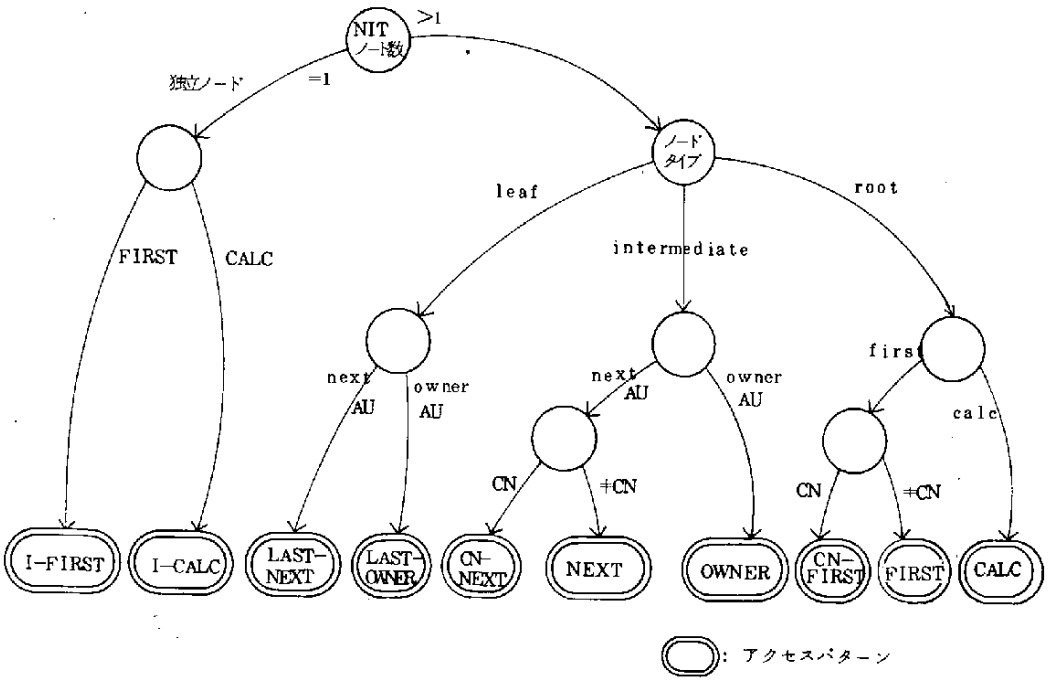


図5.33　アクセス単位（ＡＵ）の分類

図5.33の手順によって分類されたアクセスパターンから、次に図5.34を用いて各アクセス単位（ＡＵ）に対応するＤＭＬブロックを生成する。ブロック内の%のついた変数、e.g. %01, %02, %10, はブロック内の内部変数を示している。特に%0n（n＝1, 2, …, 9）はブロック内のラベル名を表わしている。%1n（n＝0, 1, …, 9）と%2n（n＝0, 1, …, 9）とはブロック内で用いるプログラム変数を表わしている。全ての%0は、プログラムの実行前にｆａｌｓｅとなっている。このうち、%2nは、このブロック内のアクセス対象のＤＢＴＧ要素のカランシー（ｃｕｒｒｅｎｃｙ）を保持するために用いられる。$0n（n＝0, 1, …, 9）は、アクセス単位（ＡＵ）から与えられるパラメータである。

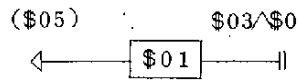
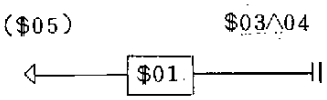
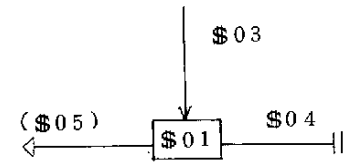
パターン	アクセス木	DML blocks
FIRST	 □ : ATノード → : 結果リンク —‖ : 制限リンク	LFOUND ←false. FIND FIRST $01. go to %φ2. %01. FIND NEXT $01. %02. if endset="YES", go to$06. GET $01. if~($03 and $04)go to%01. call RESULT($05).
CN-FIRST		LFOUND←false. FIND FIRST $01. go to %02. %01. FIND $01 DB-KEY IS %20. FIND NEXT $01. %02. if endset="YES",go to $06 ACCEPT %20 FROM CURR-ENCY. GET $01. if~($03 and $04),go to%01. call RESULT($05)。
CALC		LFOUND ←false. %01. call GNV(($03), val,mode) if mode="END", go to $06. MOVE val TO $08 IN $01. FIND ANY $01. if not found = "YES", go to $02. GET $01. if~($04), go to %01. call RESULT($05).

図5.34 アクセスパターンとDMLブロック(1)

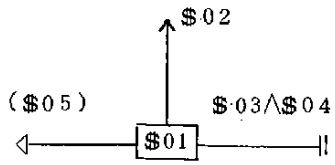
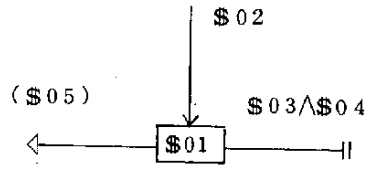
パターン	アクセス木	DML blocks
OWNER	 □：ATノード ↓：AT枝	%01. FIND OWNER WITHIN $02. if notfound≠"YES", go to %03. %02. LFOUND←false. go to $06. %03. GET $01. if ～($03 and $04),go to %02. LFOUND←true. call RESULT($05).
NEXT	 □：ATノード ↓：AT枝	%01. if %10 is true, go to %02. LFOUND←false. if $02 is enpty, go to $06. %10←true. %11←false. go to %03. %02. %11←%11 ∨ LFOUND. %03. FIND NEXT $01 WITHIN $02. if endset ≠ "YES",go to %04. LFOUND←%11. %10←false. if%11 is false, go to $06. go to $07. GET $01. if ～($03 and $04), go to %03. call RESULT ($05).

図5.34　アクセスパターンとDMLブロック(2)

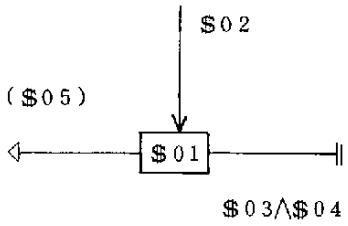
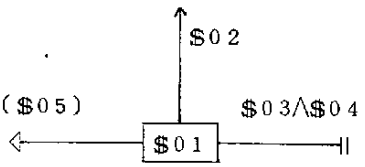
パターン	アクセス木	DML blocks
CN− NEXT		%01. if %10 is true, go to %02. LFOUND←false. if $02 is empty, go to $06. %10←true. %11←false. go to %03. * entry point from the other node %02. %11←%11∨LFOUND. FIND $01 DB−KEY is %20. %03. FIND NEXT $01 WITHIN $02. if endset≠"YES". go to %04. LFOUND←%11. %10←false. if %11 is false, go to $06. go to $07. %04. ACCEPT %20 FROM $02. CURRENCY. GET $01. if ～($03 and $04), go to %03.
LAST− OWNER		%01. FIND OWNER WITHIN $02. if notfound="no", go to %03. %02. LFOUND←false. go to $06. %03. GET $01. if ～($03 and $04), go to %02. LFOUND←true. call RESULT($05). go to $07.

図5.34　アクセスパターンとDMLブロック (3)

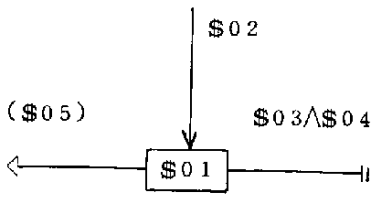
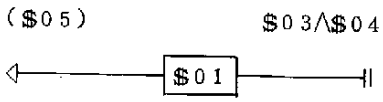
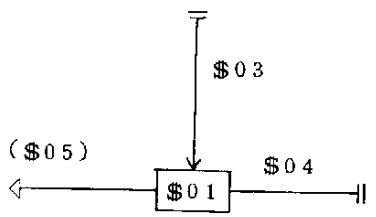
パターン	アクセス木	DML blocks
LAST-NEXT		%01. if %10 is true, go to %02. if $02 is empty, go to $06. %11←false. %10←true. %02. FIND NEXT $01 WITHIN $02. if endset="yes", go to %03. GET $01. if ～($03 and $04), go to %01. call RESULT($05). go to %02. %03. LEOUND←%11. %10←false. if %11 is false, go to $06. go to $07.
I-FIRST		FIND FIRST $01. go to %02. %01. FIND NEXT $01. %02. if endset="yes", go to $06. GET $01. if ～($03∧$04), go to %01. call RESULT($05). go to %01.
I-CALC		%01. call GNV(($03), val, mode) if mode="END", go to $06 MOVE val TO $08 IN $01. FIND ANY $01. if notfound="YES", go to %01. GET $01. if ～($04), go to %01. call RESULT ($05). go to %01.

図5.34 アクセスパターンとDMLブロック (4)

＄０１は、レコード型を、＄０２はセット型を表わしている。＄０３と＄０４は各々、ＣＡＬＣを参照する等価制限式とこれ以外の制限式とを表わしている。＄０５は、結果属性リストを表わしている。＄０６は、失敗復帰先ＡＵ番号の先頭ラベル,＄０７は成功復帰ＡＵ番号の先頭ラベルを示している。このラベルは、対応するＡＵ内の内部ラベル文％０１である。＄０８はレコード型のＣＡＬＣ項目名である。即ち、アクセスユニットは

```
(<unit-no>(SET $02)
          (REC $01 (RSLT $05)
                   (CEQR $03)
                   (RSTR $04))
          (UTYPE <a><b><c>)
          (FRTRN $06)(SRTRN $07)
```

である。対応するパラメータが、アクセス単位内でＮＩＬの時、そのパラメータがふくまれている文は出力されない。

例えば、図5.３２内の６番目のアクセス単位はＬＡＳＴ－ＯＷＮＥＲである。このアクセス単位からは図5.３５のようなＤＭＬブロックが生成される。

```
L0601. FIND OWNER WITHIN PROJ-ENGI.
       if notfound="no", go to L0603.
L0602. LFOUND←false.
                        go to L0501.
L0603. GET RPOJECT.
       if not(projstat="ON"and projsyar ≥1975),
                        go to L0602.
       LFOUND←true.
       call RESULT(projname).
                        go to L0501.
```

図5.３５　ＤＭＬブロックの例

合流ノード（ＣＮ）についてのアクセス単位であるＣＮ－ＦＩＲＳＴとＣＮ－ＮＥＸＴ、ＤＭＬブロックとではカランシ（currency）を保持するためのＡＣＣＥＰＴ文と、カランシをもとへ戻すＦＩＮＤとが、各々ＦＩＲＳＴとＮＥＸＴのＤＭＬブロックに加えられている。

合流ノードであるＤＱＧノード（レコード型）は、１つのアクセスパス内で合流ノードに対応した異なったセット型を介して複数回アクセスされる。又、アクセスパス内のＮＥＸＴノードは、１つのアクセスパス内でくり返してアクセスされる。このため、ある合流ノードをアクセスされた後、次の同一のＤＱＧノードに対する合流ノードがアクセスされると、前のアクセスされたオ

カーランスについてのカランシは失なわれてしまい、このＤＱＧノードのカランシは２回目のアクセスのオカーランスを指すことになる。前の合流ノードがＮＥＸＴノードであれば、アクセスパスは、当然、最初にアクセスされたオカーランスの次のオカーランス（ＮＥＸＴ）をアクセスにくるが、この時、カランシは既に２回目の合流ノードのアクセスオカーランスへ移ってしまっているので、必要なオカーランスをアクセスできなくなる。このため、ＣＮ－ＦＩＲＳＴとＣＮ－ＮＥＸＴとでは、このＮＩＴノードへのアクセスが終了した時、現在のセット型のカランシを％２０の領域へＡＣＣＥＰＴ文によって退避しておく。次のこのＮＩＴノードへのアクセス時には、まずカランシを％２０に退避されていた値へ、ＦＩＮＤ　$01 DB-KEY IS %02. によって復旧してから、ＦＩＮＤ　ＮＥＸＴ文でもとめるオカーランスをアクセスする。

ＲＥＳＵＬＴルーティンは、結果リレーションを出力するものである。パラメータに与えられた属性値を結果属性に出力することを表わしている。ＧＮＶルーティンは、ＣＡＬＣ等価結合式が、<v-att>=v_1 or <v-att>=v_2 or……or<v-att>=v_nのような形の時に、値のリスト（$v_1, v_2, \cdots; v_n$）から順に値を取り出すものである。

5.5.4 ＤＭＬプログラムの例

アクセス木をpreorderにたどりながら、各アクセス単位ごとに生成されるＤＭＬブロックの集合は、求めるＤＭＬプログラムの手続部（procedure division)となる。図5.4に示したＱＵＥＬで書かれたＬＣＳ問合せ（ＬＱ）から横型アクセス木生成アルゴリズム（ＢＦＡ）によって生成されるＤＭＬプログラムを図5.36に示し、縦型アクセス木生成アルゴリズム（ＤＦＡ）によるものを図5.37に示す。先述した様に、ＢＦＡは多くの中間結果と、これらの間の集合演算とを必要としている。しかし、ＤＦＡでは、ただ１つの結果リレーションだけが必要であり、ＢＦＡにおける様な集合演算は不要である。この点がＤＦＡの長所でもある。

図5.4と同様に、4.6.2項で述べたプロジェクトデータベースシステム（ＰＲＤＢＳ）の局所概念スキーマ（ＬＣＳ）〔4.6.2節〕に基づいたＬＣＳ問合せから、ＤＢＴＧ　ＤＭＬプログラムへの変換例について次に示す。

```
*---------- DBTG DML ----------

          MOVE FALSE TO LFOUND.
 L0101.   CALL  GET-NEXT-VALUE ((KEYWORD = "DATABASE" ),VAL,MODE).
          IF  MODE = 'END'  GO TO TERM.
          MOVE VAL TO KEYWORD IN KEYWORDS.
          FIND ANY KEYWORDS.
          IF  NOTFOUND = 'YES'  GO TO L0101.
          GET KEYWORDS.
*

 L0201.   IF  L0210 = TRUE  GO TO L0202.
          MOVE FALSE TO LFOUND.
          IF  KEYW-PROJ IS EMPTY  GO TO L0101.
          MOVE TRUE  TO L0210.
          MOVE FALSE TO L0211.        GO TO L0203.
 L0202.
          IF  L0211 = TRUE  OR  LFOUND = TRUE
                    MOVE TRUE  TO L0211
               ELSE MOVE FALSE TO L0211.
 L0203.   FIND NEXT PROJ-KEYW-LINK WITHIN KEYW-PROJ.
          IF  ENDSET NOT = 'YES'  GO TO L0204.
          MOVE L0211 TO LFOUND.
          MOVE FALSE TO L0210.
          IF  L0211 = FALSE  GO TO L0101.
                             GO TO L0101.
 L0204.   GET PROJ-KEYW-LINK.
*

 L0301.   FIND OWNER WITHIN PROJ-KEYW.
          IF  NOTFOUND NOT = 'YES'  GO TO L0303.
 L0302.   MOVE FALSE TO LFOUND.
          GO TO L0201.
 L0303.
          GET PROJECT.

          IF  NOT ( PROJSTAT = "ON" AND PROJSYAR GE  1975 )  GO TO
           L0302.
          MOVE TRUE TO LFOUND.
          CALL  RESULT (PROJNAME ).
```

図5.3 6　DMLプログラム（BFA）(1)

```
*
 L0401.  FIND OWNER WITHIN REPR-PROJ.
         IF  NOTFOUND NOT = 'YES'  GO TO L0403.
 L0402.  MOVE FALSE TO LFOUND.
         GO TO L0201.
 L0403.
         GET REPRESENTATIVE.
         MOVE TRUE TO LFOUND.
*
 L0601.  IF  L0610 = TRUE  GO TO L0602.
         IF  REPR-ENGI IS EMPTY  GO TO L0201.
         MOVE FALSE TO L0611.
         MOVE TRUE  TO L0610.
 L0602.
         FIND NEXT ENGINEER WITHIN REPR-ENGI.
         IF  ENDSET = 'YES'  GO TO L0603.
         GET ENGINEER.
         CALL  RESULT (ENGINAME ).
         GO TO L0602.
 L0603.
         MOVE L0611 TO LFOUND.
         MOVE FALSE TO L0610.
         IF  L0611 = FALSE  GO TO L0201.
                            GO TO L0501.

 L0501.  IF  L0510 = TRUE  GO TO L0502.
         IF  PROJ-ENGI IS EMPTY  GO TO L0201.
         MOVE FALSE TO L0511.
         MOVE TRUE  TO L0510.
 L0502.
         FIND NEXT ENGINEER WITHIN PROJ-ENGI.
         IF  ENDSET = 'YES'  GO TO L0503.
         GET ENGINEER.
         CALL  RESULT (ENGINAME ).
         GO TO L0502.
 L0503.
         MOVE L0511 TO LFOUND.
         MOVE FALSE TO L0510.
         IF  L0511 = FALSE  GO TO L0201.
                            GO TO L0201.

RESULT (REPRESENTATIVE  ) <-- JOIN RESULT (REPRESENTATIVE  )

AND RESULT (ENGINEER          ) ON ENGINEER          .ENGINAME
```

図5.3.6　DMLプログラム（BFA）(2)

```
GO;
     *---------- DBTG DML ----------

               MOVE FALSE TO LFOUND.
     L0101.    CALL  GET-NEXT-VALUE ((KEYWORD = "DATABASE" ),VAL,MODE).
               IF  MODE = 'END'  GO TO TERM.
               MOVE VAL TO KEYWORD IN KEYWORDS.
               FIND ANY KEYWORDS.
               IF  NOTFOUND = 'YES'  GO TO L0101.
               GET KEYWORDS.
     *

     L0201.    IF  L0210 = TRUE  GO TO L0202.
               MOVE FALSE TO LFOUND.
               IF  KEYW-PROJ IS EMPTY  GO TO L0101.
               MOVE TRUE  TO L0210.
               MOVE FALSE TO L0211.       GO TO L0203.
     L0202.
               IF  L0211 = TRUE  OR  LFOUND = TRUE
                         MOVE TRUE  TO L0211
                    ELSE MOVE FALSE TO L0211.
     L0203.    FIND NEXT PROJ-KEYW-LINK WITHIN KEYW-PROJ.
               IF  ENDSET NOT = 'YES'  GO TO L0204.
               MOVE L0211 TO LFOUND.
               MOVE FALSE TO L0210.
               IF  L0211 = FALSE  GO TO L0101.
                                  GO TO L0101.
     L0204.    GET PROJ-KEYW-LINK.
     *

     L0301.    FIND OWNER WITHIN PROJ-KEYW.
               IF  NOTFOUND NOT = 'YES'  GO TO L0303.
     L0302.    MOVE FALSE TO LFOUND.
               GO TO L0201.
     L0303.
               GET PROJECT.
               IF  NOT ( PROJSTAT = "ON" AND PROJSYAR GE  1975 )  GO TO
                L0302.
               MOVE TRUE TO LFOUND.
               CALL  RESULT (PROJNAME ).
     *

     L0401.    FIND OWNER WITHIN REPR-PROJ.
               IF  NOTFOUND NOT = 'YES'  GO TO L0403.
     L0402.    MOVE FALSE TO LFOUND.
               GO TO L0201.
     L0403.
               GET REPRESENTATIVE.
               MOVE TRUE TO LFOUND.
     *

     L0601.    IF  L0610 = TRUE  GO TO L0602.
               MOVE FALSE TO LFOUND.
               IF  REPR-ENGI IS EMPTY  GO TO L0201.
               MOVE TRUE  TO L0610.
               MOVE FALSE TO L0611.       GO TO L0603.
     L0602.
               IF  L0611 = TRUE  OR  LFOUND = TRUE
                         MOVE TRUE  TO L0611
                    ELSE MOVE FALSE TO L0611.
     L0603.    FIND NEXT ENGINEER WITHIN REPR-ENGI.
               IF  ENDSET NOT = 'YES'  GO TO L0604.
               MOVE L0611 TO LFOUND.
               MOVE FALSE TO L0610.
               IF  L0611 = FALSE  GO TO L0201.
                                  GO TO L0201.
     L0604.    GET ENGINEER.
               CALL  RESULT (ENGINAME ).
     *

     L0701.    FIND OWNER WITHIN PROJ-ENGI.
               IF  NOTFOUND = 'NO'  GO TO L0703.
     L0702.    MOVE FALSE TO LFOUND.
               GO TO L0601.
     L0703.
               GET PROJECT.
               IF  NOT ( PROJSTAT = "ON" AND PROJSYAR GE  1975 )  GO TO
                L0702.
               MOVE TRUE  TO LFOUND.
               CALL  RESULT (PROJNAME ).
               GO TO L0601.
```

図5.37　DMLプログラム（DFA）

```
QT;
RANGE OF ( P1, P2, P3, P4 ) ( PROJECT, PROJECT, PROJECT, PROJECT );
RANGE OF ( E1, E2, E3, E4 ) ( ENGINEER, ENGINEER, ENGINEER, ENGINEER);
RANGE OF ( RP1, RP2, RP3, RP4) (REPR-PROJ, REPR-PROJ, REPR-PROJ, REPR-PROJ);
RANGE OF ( PE1, PE2, PE3, PE4) (PROJ-ENGI, PROJ-ENGI, PROJ-ENGI, PROJ-ENGI);
RANGE OF ( RE1, RE2, RE3, RE4) (REPR-ENGI, REPR-ENGI, REPR-ENGI, REPR-ENGI);
RANGE OF ( PL1, PL2, PL3, PL4) (PROJ-LINK, PROJ-LINK, PROJ-LINK, PROJ-LINK);
RANGE OF ( R1, R2, R3, R4 ) ( REPRESENTATIVE, REPRESENTATIVE ,
                              REPRESENTATIVE, REPRESENTATIVE );
RANGE OF ( P5 , P6 ) ( PROJECT, PROJECT );


RETRIEVE INTO TEST52 ( RE1.ENGINAME, RP1.REPR-NO )
WHERE
     RE1.REPR-NO = RP1.REPR-NO   AND  RP1.PROJ-NO  = PE1.PROJ-NO   AND
     PE1.ENGINAME= RE1.ENGINAME
```

図5.3 8　テスト５２のＬＣＳ問合せ

図5.3 8のＬＣＳ問合せに対するリレーショナル問合せグラフ（ＲＱＧ）を図5.3 9(a)に、対応するＤＢＴＧ問合せグラフ（ＤＱＧ）を(b)に各々示す。これらの図からわかるように、この問合せは関係性集合リレーション（ＲＳＲ）のＲＥＰＲ−ＥＮＧＩ，ＰＲＯＪ−ＥＮＧＩ,ＲＥＰＲ−ＰＲＯＪのみを参照している。よってＤＢＴＧのレコード型ＲＥＰＲＥＳＥＮＴＡＴＩＶＥ，ＰＲＯＪＥＣＴ，ＥＮＧＩＮＥＥＲが隠れ構造になっている。これらの隠れ構造は異種性情報（ＨＩ）を用いて明らかにされ、図5.3 9(b)の様なＤＢＴＧ問合せグラフ（ＤＱＧ）が得られる。

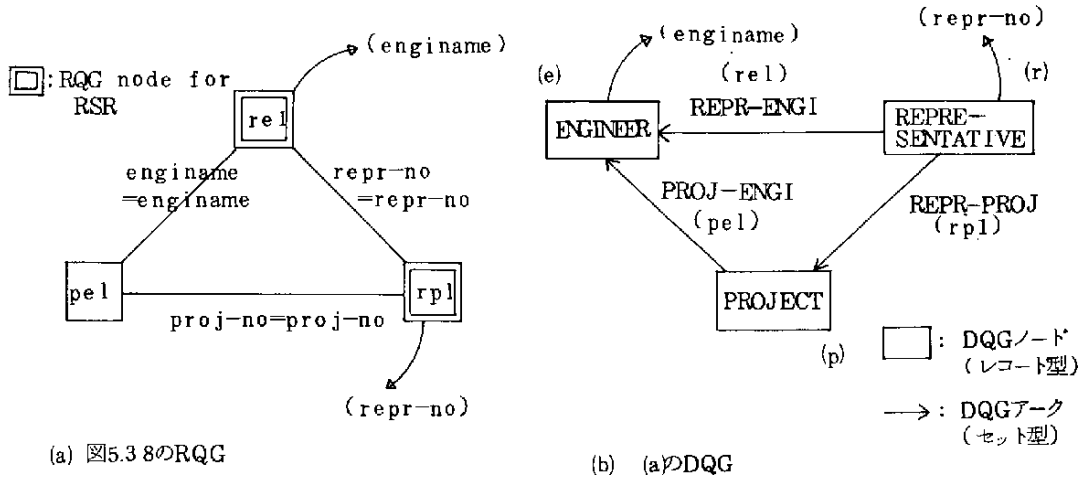


図5.3 9　図5.3 8のＲＱＧとＤＱＧ

この中でREPRESENTATIVEのカーディナリティが一番小さいので、DQGノードr（REPRESENTATIVE）がアクセス木の根ノードとして選ばれる。DFAによって生成されたアクセス木を図5.40に示す。

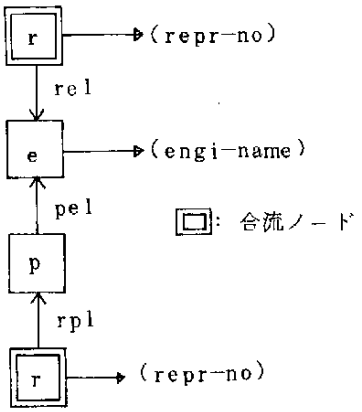


図5.40　アクセス木

DQGノードrは、アクセス木内に合流ノードを2つもつことがわかる。最初のATノードrは、合流ノードでかつCALC等価制限式を持たないので、これはCN-FIRSTアクセス単位となる。このアクセス単位では、DQGノードr（即ち、レコード型REPRESENTATIVE）をアクセスするたびに、現在のカランシを保持し、次のアクセス時には、このセーブされたカランシを復帰してその次のオカーランスをアクセスする必要がある。

```
GO;
      *---------- DBTG DML ----------

              MOVE FALSE TO LFOUND.
              FIND FIRST REPRESENTATIVE.
              GO TO L0102.
       L0101.
              FIND REPRESENTATIVE DB-KEY IS L0120.
              FIND NEXT REPRESENTATIVE.
       L0102. IF  ENDSET = 'YES'  GO TO TERM.
              ACCEPT L0120 FROM CURRENCY.
              GET REPRESENTATIVE.
      *
       L0201. IF  L0210 = TRUE  GO TO L0202.
              MOVE FALSE TO LFOUND.
              IF  REPR-ENGI IS EMPTY  GO TO L0101.
              MOVE TRUE  TO L0210.
              MOVE FALSE TO L0211.        GO TO L0203.
       L0202.
              IF  L0211 = TRUE  OR  LFOUND = TRUE
                        MOVE TRUE  TO L0211
                   ELSE MOVE FALSE TO L0211.
       L0203. FIND NEXT ENGINEER WITHIN REPR-ENGI.
              IF  ENDSET NOT = 'YES'  GO TO L0204.
              MOVE L0211 TO LFOUND.
              MOVE FALSE TO L0210.
              IF  L0211 = FALSE  GO TO L0101.
                                 GO TO L0101.
       L0204. GET ENGINEER.
      *
       L0401. FIND OWNER WITHIN PROJ-ENGI.
              IF  NOTFOUND NOT = 'YES'  GO TO L0403.
       L0402. MOVE FALSE TO LFOUND.
              GO TO L0201.
       L0403.
              GET PROJECT.
              MOVE TRUE TO LFOUND.
      *
       L0501. FIND OWNER WITHIN REPR-PROJ.
              IF  NOTFOUND = 'NO'  GO TO L0503.

       L0502. MOVE FALSE TO LFOUND.
              GO TO L0201.
       L0503.
              GET REPRESENTATIVE.
              MOVE TRUE  TO LFOUND.
              GO TO L0201.
```

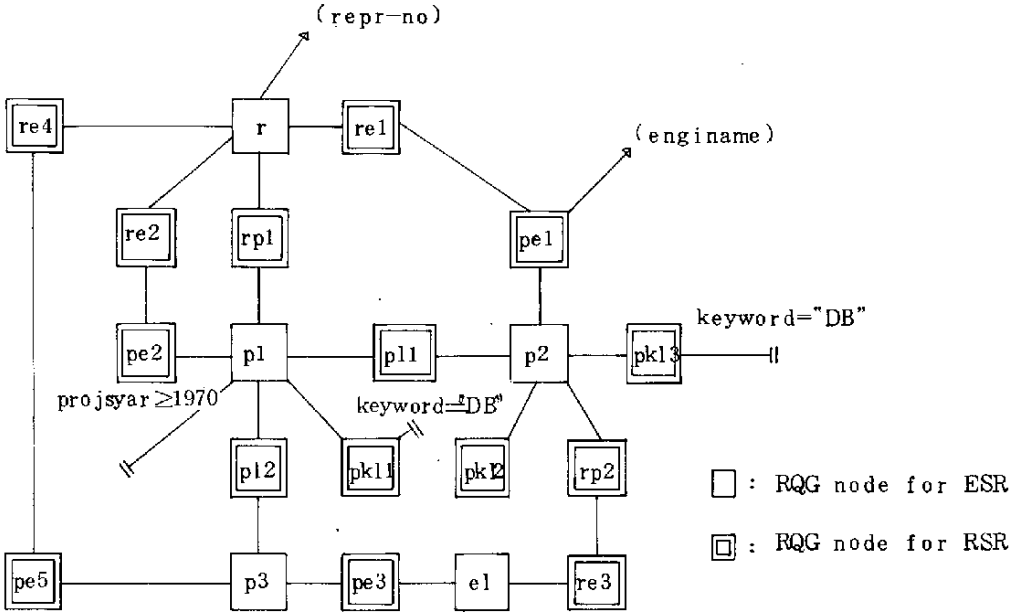
図5.41　DMLプログラム

2) テストＲＲ

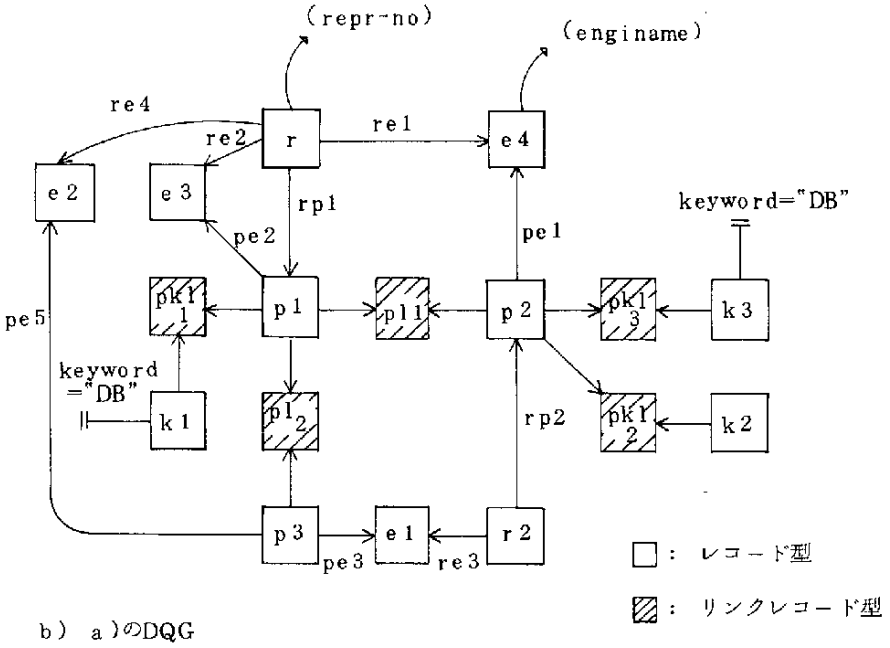
```
@T;
RANGE OF ( P1, P2, P3, P4 ) ( PROJECT, PROJECT, PROJECT, PROJECT );
RANGE OF (RP1, RP2) (REPR-PROJ,REPR-PROJ);
RANGE OF ( PKL1  , PKL2   ) ( PROJ-KEYW-LINK  , PROJ-KEYW-LINK   );
RANGE OF ( PKL3  , PKL4   ) ( PROJ-KEYW-LINK  , PROJ-KEYW-LINK   );
RANGE OF ( PL1, PL2, PL3, PL4) ( PROJ-LINK, PROJ-LINK, PROJ-LINK, PROJ-LINK);
RANGE OF ( R, E ) ( REPRESENTATIVE, ENGINEER );
RANGE OF ( PE1, PE2, PE3)  ( PROJ-ENGI, PROJ-ENGI, PROJ-ENGI);
RANGE OF ( RE1, RE2, RE3 )  ( REPR-ENGI, REPR-ENGI, REPR-ENGI);
RANGE OF ( S, PSL )  ( SPONSOR , PROJ-SPON-LINK );
RANGE OF ( RE4 )   ( REPR-ENGI )  ;
RANGE OF ( PE5 )   ( PROJ-ENGI )  ;
RANGE OF ( E1  )   ( ENGINEER  )  ;

RETRIEVE INTO RR ( R.REPR-NO, PE1.ENGINAME )
WHERE
 R.REPR-NO   = RE2.REPR-NO     AND R.REPR-NO  = RP1.REPR-NO  AND
 R.REPR-NO   = RE1.REPR-NO     AND
RE2.ENGINAME= PE2.ENGINAME     AND PE2.PROJ-NO= P1.PROJ-NO    AND
RP1.PROJ-NO = P1.PROJ-NO       AND P1.PROJ-NO = PKL1.PROJ-NO AND
P1.PROJ-NO  = PL1.PROJ-NO      AND
RE1.ENGINAME= PE1.ENGINAME     AND PE1.PROJ-NO= P2.PROJ-NO    AND
PKL1.KEYWORD= "DATABASE"       AND PKL2.PROJ-NO=P2.PROJ-NO    AND
PL1.PROJ-NO = P2.PROJ-NO       AND P2.PROJ-NO = PKL3.PROJ-NO AND
PKL3.KEYWORD= "DATABASE"       AND P1.PROJSYAR GE 1970        AND
P2.PROJSTAT = "ON"             AND
RE4.REPR-NO = R.REPR-NO        AND RE4.ENGINAME= PE5.ENGINAME AND
PE5.PROJ-NO = P3.PROJ-NO       AND P3.PROJ-NO  = PL2.PROJ-NO  AND
PL2.PRIOR-NO= P1.PROJ-NO       AND
P3.PROJ-NO  = PE3.PROJ-NO      AND PE3.ENGINAME= E1.ENGINAME  AND
E1.ENGINAME = RE3.ENGINAME     AND RE3.REPR-NO = RP2.REPR-NO  AND
RP2.PROJ-NO = P2.PROJ-NO
```

図5.42　テストＲＲのＬＣＳ問合せ

図5.42のＬＣＳ問合せに対するリレーショナル問合せグラフ（ＲＱＧ）とＤＢＴＧ問合せグラフ（ＤＱＧ）とを、各々図5.43の(a)と(b)とに示す。この問合せは、２０個の組変数を参照するとともにいくつかの隠れ構造を持っている。ＤＦＡによって生成されたアクセス木を図5.43のc）に、又最終的に生成されるＤＭＬプログラムを図5.44に示す。

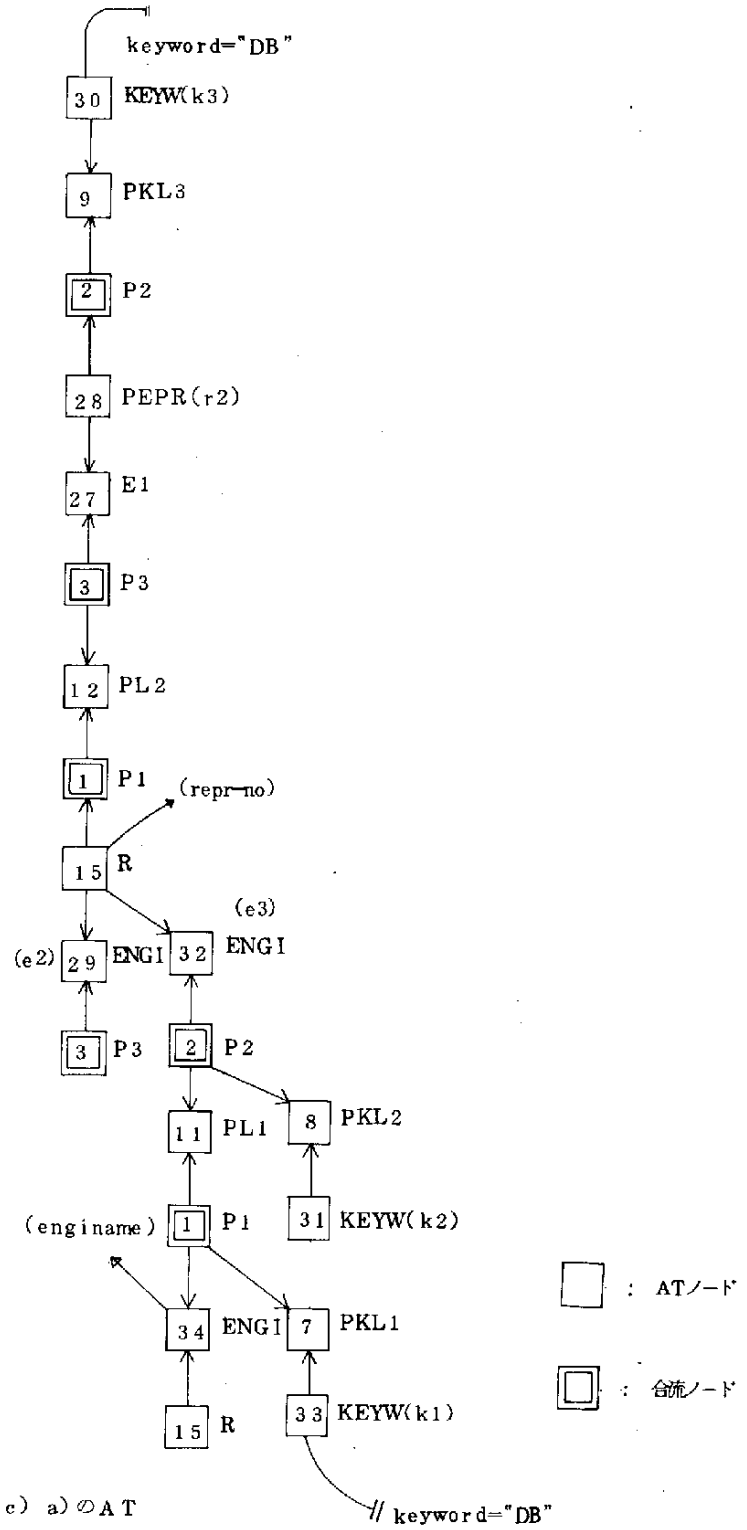


a) RQG



b) a)のDQG

図5.4.3 テストRRのRQG、DQGとAT (1)



c) a)のAT

図5.43 テストRRのRQG，DQGとAT(2)

```
GO;
      *---------- DBTG DML ----------

                MOVE FALSE TO LFOUND.
      L0101.    CALL  GET-NEXT-VALUE ((KEYWORD = "DATABASE" ),VAL,MODE).
                IF  MODE = 'END'  GO TO TERM.
                MOVE VAL TO KEYWORD IN KEYWORDS.
                FIND ANY KEYWORDS.
                IF  NOTFOUND = 'YES'  GO TO L0101.
                GET KEYWORDS.
      *
      L0201.    IF  L0210 = TRUE  GO TO L0202.
                MOVE FALSE TO LFOUND.
                IF  KEYW-PROJ IS EMPTY  GO TO L0101.
                MOVE TRUE  TO L0210.
                MOVE FALSE TO L0211.        GO TO L0203.
      L0202.
                IF  L0211 = TRUE  OR  LFOUND = TRUE
                          MOVE TRUE  TO L0211
                     ELSE MOVE FALSE TO L0211.
      L0203.    FIND NEXT PROJ-KEYW-LINK WITHIN KEYW-PROJ.
                IF  ENDSET NOT = 'YES'  GO TO L0204.
                MOVE L0211 TO LFOUND.
                MOVE FALSE TO L0210.
                IF  L0211 = FALSE  GO TO L0101.
                                   GO TO L0101.
      L0204.    GET PROJ-KEYW-LINK.
      *
      L0301.    FIND OWNER WITHIN PROJ-KEYW.
                IF  NOTFOUND NOT = 'YES'  GO TO L0303.
      L0302.    MOVE FALSE TO LFOUND.
                GO TO L0201.
      L0303.
                GET PROJECT.
                IF  NOT ( PROJSTAT = "ON" )  GO TO L0302.
                MOVE TRUE TO LFOUND.
      *
      L0401.    FIND OWNER WITHIN REPR-PROJ.
                IF  NOTFOUND NOT = 'YES'  GO TO L0403.
      L0402.    MOVE FALSE TO LFOUND.
                GO TO L0201.
      L0403.
                GET REPRESENTATIVE.
                MOVE TRUE TO LFOUND.
      *
      L0801.    IF  L0810 = TRUE  GO TO L0802.
                MOVE FALSE TO LFOUND.
                IF  REPR-ENGI IS EMPTY  GO TO L0201.
                MOVE TRUE  TO L0810.
                MOVE FALSE TO L0611.        GO TO L0803.
      L0802.
                IF  L0811 = TRUE  OR  LFOUND = TRUE
                          MOVE TRUE  TO L0811
                     ELSE MOVE FALSE TO L0811.
      L0803.    FIND NEXT ENGINEER WITHIN REPR-ENGI.
                IF  ENDSET NOT = 'YES'  GO TO L0804.
                MOVE L0811 TO LFOUND.
                MOVE FALSE TO L0810.
                IF  L0811 = FALSE  GO TO L0201.
                                   GO TO L0201.
      L0804.    GET ENGINEER.
```

図5.4.4　RRのDMLプログラム (1)

```
*
L0901.  FIND OWNER WITHIN PROJ-ENGI.
        IF  NOTFOUND NOT = 'YES'  GO TO L0903.
L0902.  MOVE FALSE TO LFOUND.
        GO TO L0801.
L0903.
        GET PROJECT.
        MOVE TRUE TO LFOUND.
*
L1001.  IF  L1010 = TRUE  GO TO L1002.
        MOVE FALSE TO LFOUND.
        IF  PRIOR-PROJ IS EMPTY  GO TO L0801.
        MOVE TRUE  TO L1010.
        MOVE FALSE TO L1011.        GO TO L1003.
L1002.
        IF  L1011 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L1011
            ELSE MOVE FALSE TO L1011.
L1003.  FIND NEXT PROJ-LINK WITHIN PRIOR-PROJ.
        IF  ENDSET NOT = 'YES'  GO TO L1004.
        MOVE L1011 TO LFOUND.
        MOVE FALSE TO L1010.
        IF  L1011 = FALSE  GO TO L0801.
                           GO TO L0801.
L1004.  GET PROJ-LINK.
*
L1201.  FIND OWNER WITHIN LAST-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L1203.
L1202.  MOVE FALSE TO LFOUND.
        GO TO L1001.
L1203.
        GET PROJECT.
        IF  NOT ( PROJSYAR GE  1970 )  GO TO L1202.
        MOVE TRUE TO LFOUND.
*
L1301.  FIND OWNER WITHIN REPR-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L1303.
L1302.  MOVE FALSE TO LFOUND.
        GO TO L1001.
L1303.
        GET REPRESENTATIVE.
        MOVE TRUE TO LFOUND.
        CALL  RESULT (REPR-NO ).
*
L1701.  IF  L1710 = TRUE  GO TO L1702.
        MOVE FALSE TO LFOUND.
        IF  REPR-ENGI IS EMPTY  GO TO L1001.
        MOVE TRUE  TO L1710.
        MOVE FALSE TO L1711.        GO TO L1703.
L1702.
        IF  L1711 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L1711
            ELSE MOVE FALSE TO L1711.
L1703.  FIND NEXT ENGINEER WITHIN REPR-ENGI.
        IF  ENDSET NOT = 'YES'  GO TO L1704.
        MOVE L1711 TO LFOUND.
        MOVE FALSE TO L1710.
        IF  L1711 = FALSE  GO TO L1001.
                           GO TO L2101.
L1704.  GET ENGINEER.
*
L2001.  FIND OWNER WITHIN PROJ-ENGI.
        IF  NOTFOUND = 'NO'  GO TO L2003.
L2002.  MOVE FALSE TO LFOUND.
        GO TO L1701.
L2003.
        GET PROJECT.
        MOVE TRUE  TO LFOUND.
        GO TO L1701.
```

図5.4.4　RRのDMLプログラム(2)

```
L2101.  IF  L2110 = TRUE  GO TO L2102.
        MOVE FALSE TO LFOUND.
        IF  REPR-ENGI IS EMPTY  GO TO L1001.
        MOVE TRUE  TO L2110.
        MOVE FALSE TO L2111.        GO TO L2103.

L2102.
        IF  L2111 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L2111
            ELSE MOVE FALSE TO L2111.
L2103.  FIND NEXT ENGINEER WITHIN REPR-ENGI.
        IF  ENDSET NOT = 'YES'  GO TO L2104.
        MOVE L2111 TO LFOUND.
        MOVE FALSE TO L2110.
        IF  L2111 = FALSE  GO TO L1001.
                           GO TO L1001.
L2104.  GET ENGINEER.
*
L2301.  FIND OWNER WITHIN PROJ-ENGI.
        IF  NOTFOUND NOT = 'YES'  GO TO L2303.
L2302.  MOVE FALSE TO LFOUND.
        GO TO L2101.
L2303.
        GET PROJECT.
        IF  NOT ( PROJSTAT = "EN" )  GO TO L2302.
        MOVE TRUE TO LFOUND.
*
L2401.  IF  L2410 = TRUE  GO TO L2402.
        MOVE FALSE TO LFOUND.
        IF  PRIOR-PROJ IS EMPTY  GO TO L2101.
        MOVE TRUE  TO L2410.
        MOVE FALSE TO L2411.        GO TO L2403.
L2402.
        IF  L2411 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L2411
            ELSE MOVE FALSE TO L2411.
L2403.  FIND NEXT PROJ-LINK WITHIN PRIOR-PROJ.
        IF  ENDSET NOT = 'YES'  GO TO L2404.
        MOVE L2411 TO LFOUND.
        MOVE FALSE TO L2410.
        IF  L2411 = FALSE  GO TO L2101.
                           GO TO L3201.
L2404.  GET PROJ-LINK.
*
L2601.  FIND OWNER WITHIN LAST-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L2603.
L2602.  MOVE FALSE TO LFOUND.
        GO TO L2401.
L2603.
        GET PROJECT.
        IF  NOT ( PROJSYAR GE  1970 )  GO TO L260
        MOVE TRUE TO LFOUND.
*
L2701.  IF  L2710 = TRUE  GO TO L2702.
        MOVE FALSE TO LFOUND.
        IF  PROJ-ENGI IS EMPTY  GO TO L2401.
        MOVE TRUE  TO L2710.
        MOVE FALSE TO L2711.        GO TO L2703.
L2702.
        IF  L2711 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L2711
            ELSE MOVE FALSE TO L2711.
```

図 5.4.4　RRのDMLプログラム(3)

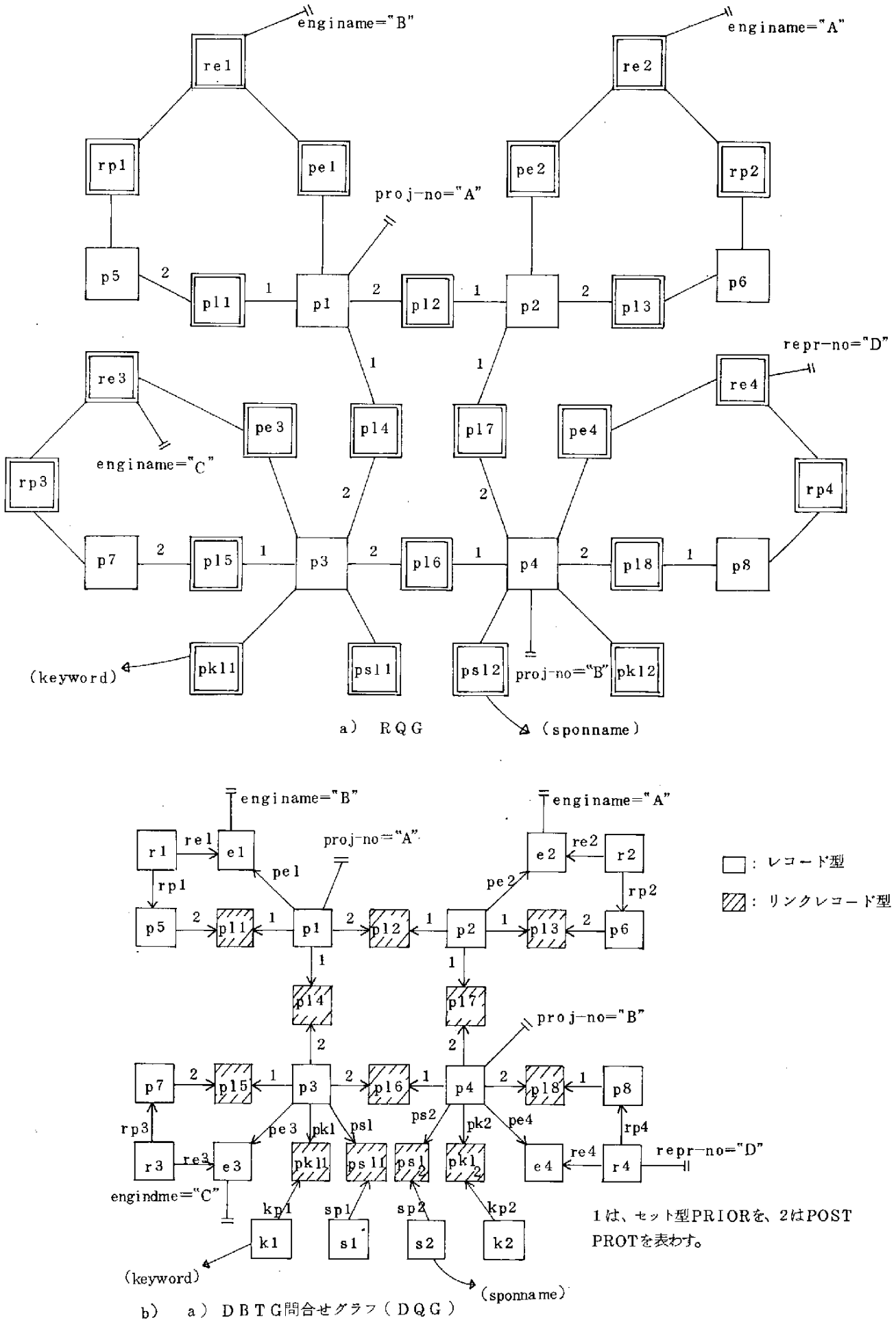
```
L2703.  FIND NEXT ENGINEER WITHIN PROJ-ENGI.
        IF  ENDSET NOT = 'YES'  GO TO L2704.
        MOVE L2711 TO LFOUND.
        MOVE FALSE TO L2710.
        IF  L2711 = FALSE  GO TO L2401.
                           GO TO L3001.
L2704.  GET ENGINEER.
*
L2901.  FIND OWNER WITHIN REPR-ENGI.
        IF  NOTFOUND = 'NO'  GO TO L2903.
L2902.  MOVE FALSE TO LFOUND.
        GO TO L2701.
L2903.
        GET REPRESENTATIVE.
        MOVE TRUE  TO LFOUND.
        CALL  RESULT (REPR-NO ).
        GO TO L2701.

L3001.  IF  L3010 = TRUE  GO TO L3002.
        MOVE FALSE TO LFOUND.
        IF  PROJ-KEYW IS EMPTY  GO TO L2401.
        MOVE TRUE  TO L3010.
        MOVE FALSE TO L3011.        GO TO L3003.
L3002.
        IF  L3011 = TRUE  OR  LFOUND = TRUE
                  MOVE TRUE  TO L3011
            ELSE MOVE FALSE TO L3011.
L3003.  FIND NEXT PROJ-KEYW-LINK WITHIN PROJ-KEYW.
        IF  ENDSET NOT = 'YES'  GO TO L3004.
        MOVE L3011 TO LFOUND.
        MOVE FALSE TO L3010.
        IF  L3011 = FALSE  GO TO L2401.
                           GO TO L2401.
L3004.  GET PROJ-KEYW-LINK.
*
L3101.  FIND OWNER WITHIN KEYW-PROJ.
        IF  NOTFOUND = 'NO'  GO TO L3103.
L3102.  MOVE FALSE TO LFOUND.
        GO TO L3001.
L3103.
        GET KEYWORDS.
        IF  NOT (KEYWORD = "DATABASE" )  GO TO L3102.
        MOVE TRUE  TO LFOUND.
        GO TO L3001.

L3201.  IF  L3210 = TRUE  GO TO L3202.
        MOVE FALSE TO LFOUND.
        IF  PROJ-KEYW IS EMPTY  GO TO L2101.
        MOVE TRUE  TO L3210.
        MOVE FALSE TO L3211.        GO TO L3203.
L3202.
        IF  L3211 = TRUE  OR  LFOUND = TRUE
                  MOVE TRUE  TO L3211
            ELSE MOVE FALSE TO L3211.
L3203.  FIND NEXT PROJ-KEYW-LINK WITHIN PROJ-KEYW.
        IF  ENDSET NOT = 'YES'  GO TO L3204.
        MOVE L3211 TO LFOUND.
        MOVE FALSE TO L3210.

        IF  L3211 = FALSE  GO TO L2101.
                           GO TO L2101.
L3204.  GET PROJ-KEYW-LINK.
*
L3301.  FIND OWNER WITHIN KEYW-PROJ.
        IF  NOTFOUND = 'NO'  GO TO L3303.
L3302.  MOVE FALSE TO LFOUND.
        GO TO L3201.
L3303.
        GET KEYWORDS.
        MOVE TRUE  TO LFOUND.
        GO TO L3201.
```

図5.4.4　RRのDMLプログラム(4)

3) テスト９１

```
@T;
RANGE OF ( PSL1, PSL2 ) ( PROJ-SPON-LINK, PROJ-SPON-LINK );
RANGE OF ( PKL1, PKL2 ) ( PROJ-KEYW-LINK, PROJ-KEYW-LINK );
RANGE OF ( P1 )  ( PROJECT );
RANGE OF ( P2 )  ( PROJECT );
RANGE OF ( P3 )  ( PROJECT );
RANGE OF ( P4 )  ( PROJECT );
RANGE OF ( P5 )  ( PROJECT );
RANGE OF ( P6 )  ( PROJECT );
RANGE OF ( P7 )  ( PROJECT );
RANGE OF ( P8 )  ( PROJECT );
RANGE OF ( PL1)  ( PROJ-LINK);
RANGE OF ( PL2)  ( PROJ-LINK);
RANGE OF ( PL3)  ( PROJ-LINK);
RANGE OF ( PL4)  ( PROJ-LINK);
RANGE OF ( PL5)  ( PROJ-LINK);
RANGE OF ( PL6)  ( PROJ-LINK);
RANGE OF ( PL7)  ( PROJ-LINK);
RANGE OF ( PL8)  ( PROJ-LINK);
RANGE OF ( PE1 )   ( PROJ-ENGI );
RANGE OF ( PE2 )   ( PROJ-ENGI );
RANGE OF ( PE3 )   ( PROJ-ENGI );
RANGE OF ( PE4 )   ( PROJ-ENGI );
RANGE OF ( RP1 )   ( REPR-PROJ );
RANGE OF ( RP2 )   ( REPR-PROJ );
RANGE OF ( RP3 )   ( REPR-PROJ );
RANGE OF ( RP4 )   ( REPR-PROJ );
RANGE OF ( RE1 )   ( REPR-ENGI );
RANGE OF ( RE2 )   ( REPR-ENGI );
RANGE OF ( RE3 )   ( REPR-ENGI );
RANGE OF ( RE4 )   ( REPR-ENGI );

RETRIEVE INTO TEST91 ( PKL1.KEYWORD, PSL2.SPONNAME )
WHERE
     RE1.REPR-NO = RP1.REPR-NO     AND  RP1.PROJ-NO   = P5.PROJ-NO      AND
     RE1.ENGINAME= PE1.ENGINAME    AND  P5.PROJ-NO    = PL1.PRIOR-NO    AND
     PL1.PROJ-NO = P1.PROJ-NO      AND  P1.PROJ-NO    = PE1.PROJ-NO     AND
     RE1.ENGINAME= "B"             AND
     RE2.ENGINAME= PE2.ENGINAME    AND  PE2.PROJ-NO   = P2.PROJ-NO      AND
     P2.PROJ-NO  = PL3.PRIOR-NO    AND  PL3.PROJ-NO   = P6.PROJ-NO      AND
     P6.PROJ-NO  = RP2.PROJ-NO     AND  RP2.REPR-NO   = RE2.REPR-NO     AND
     RE2.ENGINAME= "A"             AND
     RE3.ENGINAME= PE3.ENGINAME    AND  PE3.PROJ-NO   = P3.PROJ-NO      AND
     P3.PROJ-NO  = PL5.PROJ-NO     AND  PL5.PRIOR-NO  = P7.PROJ-NO      AND
     P7.PROJ-NO  = RP3.PROJ-NO     AND  RP3.REPR-NO   = RE3.REPR-NO     AND
     RE3.ENGINAME= "C"             AND
     RE4.REPR-NO = RP4.REPR-NO     AND  RP4.PROJ-NO   = P8.PROJ-NO      AND
     P8.PROJ-NO  = PL8.PROJ-NO     AND  PL8.PRIOR-NO  = P4.PROJ-NO      AND
     P4.PROJ-NO  = PE4.PROJ-NO     AND  PE4.ENGINAME  = RE4.ENGINAME    AND
     RE4.REPR-NO = "D"             AND
     P4.PROJ-NO  = "B"             AND
     P3.PROJ-NO  = PKL1.PROJ-NO    AND
     P3.PROJ-NO  = PSL1.PROJ-NO    AND
     P4.PROJ-NO  = PKL2.PROJ-NO    AND
     P4.PROJ-NO  = PSL2.PROJ-NO    AND
     P1.PROJ-NO  = PL2.PRIOR-NO    AND  PL2.PROJ-NO   = P2.PROJ-NO      AND
     P1.PROJ-NO  = PL4.PRIOR-NO    AND  PL4.PROJ-NO   = P3.PROJ-NO      AND
     P3.PROJ-NO  = PL6.PRIOR-NO    AND  PL6.PROJ-NO   = P4.PROJ-NO      AND
     P4.PROJ-NO  = PL7.PRIOR-NO    AND  PL7.PROJ-NO   = P2.PROJ-NO
     ;
```

図5.45　テスト９１のＬＣＳ問合せ



a） RQG

b） a） DBTG問合せグラフ（DQG）

図5.46

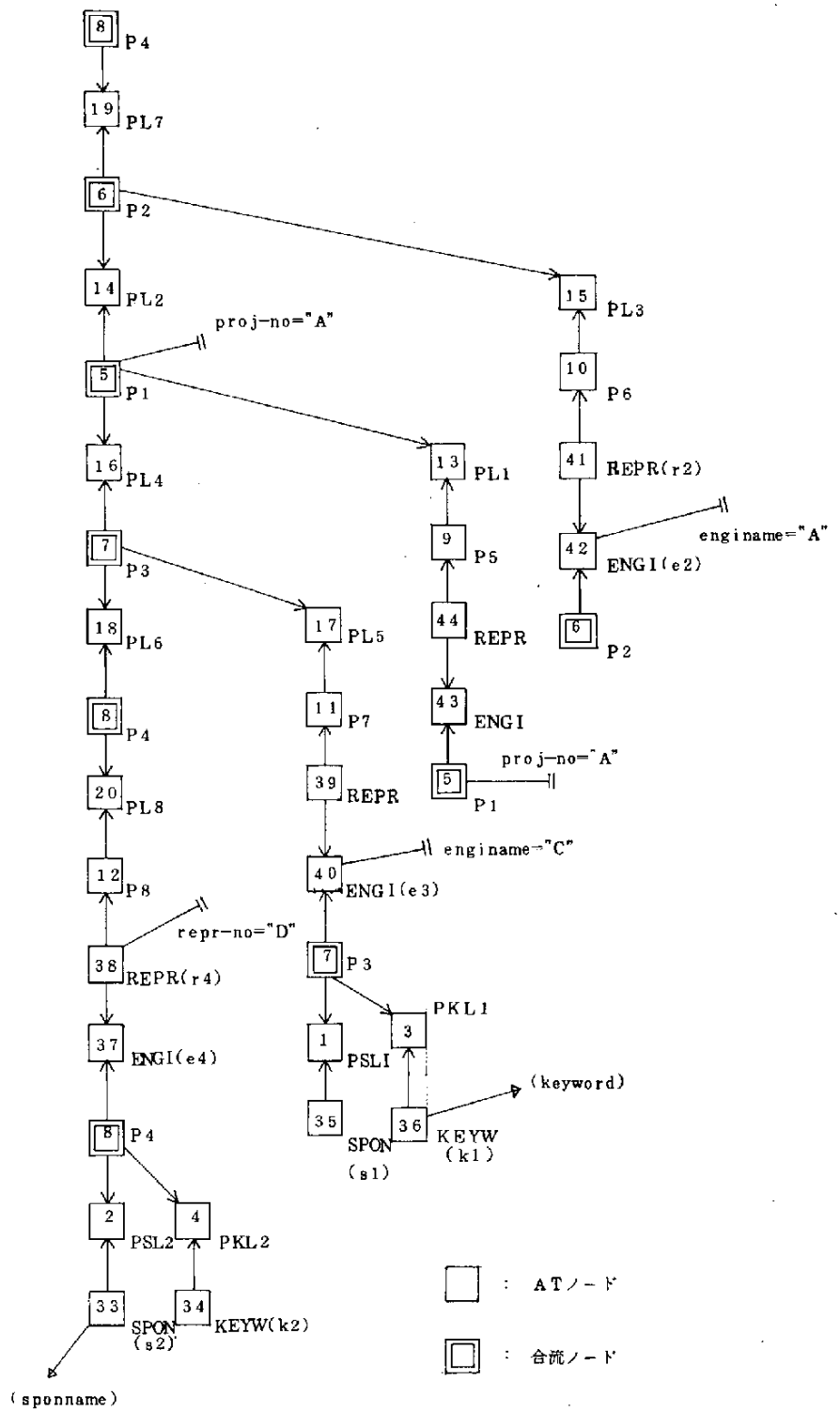


図5.47　91のアクセス木

```
*---------- DBTG DML ----------
          MOVE FALSE TO LFOUND.
 L0101.   CALL  GET-NEXT-VALUE ((PROJ-NO = "B" ),VAL,MODE).
          IF  MODE = 'END'  GO TO TERM.
          MOVE VAL TO PROJ-NO IN PROJECT.
          FIND ANY PROJECT.
          IF  NOTFOUND = 'YES'  GO TO L0101.
          GET PROJECT.
*
 L0201.   IF  L0210 = TRUE  GO TO L0202.
          MOVE FALSE TO LFOUND.
          IF  LAST-PROJ IS EMPTY  GO TO L0101.
          MOVE TRUE  TO L0210.
          MOVE FALSE TO L0211.        GO TO L0203.
 L0202.
          IF  L0211 = TRUE  OR  LFOUND = TRUE
                    MOVE TRUE  TO L0211
              ELSE MOVE FALSE TO L0211.
 L0203.   FIND NEXT PROJ-LINK WITHIN LAST-PROJ.
          IF  ENDSET NOT = 'YES'  GO TO L0204.

          MOVE L0211 TO LFOUND.
          MOVE FALSE TO L0210.
          IF  L0211 = FALSE  GO TO L0101.
                             GO TO L0101.
 L0204.   GET PROJ-LINK.
*
 L0801.   FIND OWNER WITHIN PRIOR-PROJ.
          IF  NOTFOUND NOT = 'YES'  GO TO L0803.
 L0802.   MOVE FALSE TO LFOUND.
          GO TO L0201.
 L0803.
          GET PROJECT.
          MOVE TRUE TO LFOUND.
*
 L0901.   IF  L0910 = TRUE  GO TO L0902.
          MOVE FALSE TO LFOUND.
          IF  PRIOR-PROJ IS EMPTY  GO TO L0201.
          MOVE TRUE  TO L0910.
          MOVE FALSE TO L0911.        GO TO L0903.
 L0902.
          IF  L0911 = TRUE  OR  LFOUND = TRUE
                    MOVE TRUE  TO L0911
              ELSE MOVE FALSE TO L0911.
 L0903.   FIND NEXT PROJ-LINK WITHIN PRIOR-PROJ.
          IF  ENDSET NOT = 'YES'  GO TO L0904.
          MOVE L0911 TO LFOUND.
          MOVE FALSE TO L0910.
          IF  L0911 = FALSE  GO TO L0201.
                             GO TO L5501.
 L0904.   GET PROJ-LINK.
*
 L1201.   FIND OWNER WITHIN LAST-PROJ.
          IF  NOTFOUND NOT = 'YES'  GO TO L1203.
 L1202.   MOVE FALSE TO LFOUND.
          GO TO L0901.
 L1203.
          GET PROJECT.
          MOVE TRUE TO LFOUND.
```

図5.4.8　DMLプログラム(1)

```
L1301.  IF  L1310 = TRUE  GO TO L1302.
        MOVE FALSE TO LFOUND.
        IF  LAST-PROJ IS EMPTY  GO TO L0901.
        MOVE TRUE  TO L1310.
        MOVE FALSE TO L1311.        GO TO L1303.
L1302.
        IF  L1311 = TRUE  OR  LFOUND = TRUE
                    MOVE TRUE  TO L1311
              ELSE MOVE FALSE TO L1311.
L1303.  FIND NEXT PROJ-LINK WITHIN LAST-PROJ.
        IF  ENDSET NOT = 'YES'  GO TO L1304.
        MOVE L1311 TO LFOUND.
        MOVE FALSE TO L1310.
        IF  L1311 = FALSE  GO TO L0901.
                           GO TO L4901.
L1304.  GET PROJ-LINK.

L1601.  FIND OWNER WITHIN PRIOR-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L1603.
L1602.  MOVE FALSE TO LFOUND.
        GO TO L1301.
L1603.
        GET PROJECT.
        MOVE TRUE TO LFOUND.
*

L1701.  IF  L1710 = TRUE  GO TO L1702.
        MOVE FALSE TO LFOUND.
        IF  LAST-PROJ IS EMPTY  GO TO L1301.
        MOVE TRUE  TO L1710.
        MOVE FALSE TO L1711.        GO TO L1703.
L1702.
        IF  L1711 = TRUE  OR  LFOUND = TRUE
                    MOVE TRUE  TO L1711
              ELSE MOVE FALSE TO L1711.
L1703.  FIND NEXT PROJ-LINK WITHIN LAST-PROJ.
        IF  ENDSET NOT = 'YES'  GO TO L1704.
        MOVE L1711 TO LFOUND.
        MOVE FALSE TO L1710.
        IF  L1711 = FALSE  GO TO L1301.
                           GO TO L3601.
L1704.  GET PROJ-LINK.
*

L2201.  FIND OWNER WITHIN PRIOR-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L2203.
L2202.  MOVE FALSE TO LFOUND.
        GO TO L1701.
L2203.
        GET PROJECT.
        IF  NOT (PROJ-NO = "B". )  GO TO L2202.
        MOVE TRUE TO LFOUND.
*

L2301.  IF  L2310 = TRUE  GO TO L2302.
        MOVE FALSE TO LFOUND.
        IF  LAST-PROJ IS EMPTY  GO TO L1701.
        MOVE TRUE  TO L2310.
        MOVE FALSE TO L2311.        GO TO L2303.
L2302.
        IF  L2311 = TRUE  OR  LFOUND = TRUE
                    MOVE TRUE  TO L2311
              ELSE MOVE FALSE TO L2311.
L2303.  FIND NEXT PROJ-LINK WITHIN LAST-PROJ.
        IF  ENDSET NOT = 'YES'  GO TO L2304.
        MOVE L2311 TO LFOUND.
        MOVE FALSE TO L2310.
        IF  L2311 = FALSE  GO TO L1701.
                           GO TO L1701.
L2304.  GET PROJ-LINK.
*
```

図5.48　DMLプログラム(2)

```
L2701.  FIND OWNER WITHIN PRIOR-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L2703.
L2702.  MOVE FALSE TO LFOUND.
        GO TO L2301.

L2703.
        GET PROJECT.
        MOVE TRUE TO LFOUND.
*
L2801.  FIND OWNER WITHIN REPR-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L2803.
L2802.  MOVE FALSE TO LFOUND.
        GO TO L2301.
L2803.
        GET REPRESENTATIVE.
        IF  NOT (REPR-NO = "D"  )  GO TO L2802.
        MOVE TRUE TO LFOUND.
*
L2901.  IF  L2910 = TRUE  GO TO L2902.
        MOVE FALSE TO LFOUND.
        IF  REPR-ENGI IS EMPTY  GO TO L2301.
        MOVE TRUE  TO L2910.
        MOVE FALSE TO L2911.        GO TO L2903.
L2902.
        IF  L2911 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L2911
          ELSE MOVE FALSE TO L2911.
L2903.  FIND NEXT ENGINEER WITHIN REPR-ENGI.
        IF  ENDSET NOT = 'YES'  GO TO L2904.
        MOVE L2911 TO LFOUND.
        MOVE FALSE TO L2910.
        IF  L2911 = FALSE  GO TO L2301.
                           GO TO L2301.
L2904.  GET ENGINEER.
*
L3001.  FIND OWNER WITHIN PROJ-ENGI.
        IF  NOTFOUND NOT = 'YES'  GO TO L3003.
L3002.  MOVE FALSE TO LFOUND.
        GO TO L2901.
L3003.
        GET PROJECT.
        IF  NOT (PROJ-NO = "B"  )  GO TO L3002.
        MOVE TRUE TO LFOUND.
*
L3101.  IF  L3110 = TRUE  GO TO L3102.
        MOVE FALSE TO LFOUND.
        IF  PROJ-SPON IS EMPTY  GO TO L2901.
        MOVE TRUE  TO L3110.
        MOVE FALSE TO L3111.        GO TO L3103.
L3102.
        IF  L3111 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L3111
          ELSE MOVE FALSE TO L3111.
L3103.  FIND NEXT PROJ-SPON-LINK WITHIN PROJ-SPON.
        IF  ENDSET NOT = 'YES'  GO TO L3104.
        MOVE L3111 TO LFOUND.
        MOVE FALSE TO L3110.
        IF  L3111 = FALSE  GO TO L2901.
                           GO TO L3401.
L3104.  GET PROJ-SPON-LINK.
        CALL  RESULT (SPONNAME ).

L3301.  FIND OWNER WITHIN SPON-PROJ.
        IF  NOTFOUND = 'NO'  GO TO L3303.
L3302.  MOVE FALSE TO LFOUND.
        GO TO L3101.
L3303.
        GET SPONSOR.
        MOVE TRUE  TO LFOUND.
        GO TO L3101.
```

図5.4.8　DMLプログラム(3)

```
L3401.  IF  L3410 = TRUE  GO TO L3402.
        MOVE FALSE TO LFOUND.
        IF  PROJ-KEYw IS EMPTY  GO TO L2901.
        MOVE TRUE  TO L3410.
        MOVE FALSE TO L3411.        GO TO L3403.
L3402.
        IF  L3411 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L3411
            ELSE MOVE FALSE TO L3411.
L3403.  FIND NEXT PROJ-KEYw-LINK WITHIN PROJ-KEYw.
        IF  ENDSET NOT = 'YES'  GO TO L3404.
        MOVE L3411 TO LFOUND.
        MOVE FALSE TO L3410.
        IF  L3411 = FALSE  GO TO L2901.
                           GO TO L2901.
L3404.  GET PROJ-KEYw-LINK.
*
L3501.  FIND OWNER WITHIN KEYw-PROJ.
        IF  NOTFOUND = 'NO'  GO TO L3503.
L3502.  MOVE FALSE TO LFOUND.
        GO TO L3401.
L3503.
        GET KEYWORDS.
        MOVE TRUE  TO LFOUND.
        GO TO L3401.

L3601.  IF  L3610 = TRUE  GO TO L3602.
        MOVE FALSE TO LFOUND.
        IF  PRIOR-PROJ IS EMPTY  GO TO L1301.
        MOVE TRUE  TO L3610.
        MOVE FALSE TO L3611.        GO TO L3603.
L3602.
        IF  L3611 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L3611
            ELSE MOVE FALSE TO L3611.
L3603.  FIND NEXT PROJ-LINK WITHIN PRIOR-PROJ.
        IF  ENDSET NOT = 'YES'  GO TO L3604.
        MOVE L3611 TO LFOUND.
        MOVE FALSE TO L3610.
        IF  L3611 = FALSE  GO TO L1301.
                           GO TO L1301.
L3604.  GET PROJ-LINK.
*
L4001.  FIND OWNER WITHIN LAST-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L4003.
L4002.  MOVE FALSE TO LFOUND.
        GO TO L3601.

L4003.
        GET PROJECT.
        MOVE TRUE TO LFOUND.
*
L4101.  FIND OWNER WITHIN REPR-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L4103.
L4102.  MOVE FALSE TO LFOUND.
        GO TO L3601.
L4103.
        GET REPRESENTATIVE.
        MOVE TRUE TO LFOUND.
*
L4201.  IF  L4210 = TRUE  GO TO L4202.
        MOVE FALSE TO LFOUND.
        IF  REPR-ENST IS EMPTY  GO TO L3601.
        MOVE TRUE  TO L4210.
        MOVE FALSE TO L4211.        GO TO L4203.
L4202.
        IF  L4211 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L4211
            ELSE MOVE FALSE TO L4211.
```

図5.4.8　DMLプログラム(4)

```
L4203.  FIND NEXT ENGINEER WITHIN REPR-ENGI.
        IF  ENDSET NOT = 'YES'  GO TO L4204.
        MOVE L4211 TO LFOUND.
        MOVE FALSE TO L4210.
        IF  L4211 = FALSE  GO TO L3601.
                           GO TO L3601.
L4204.  GET ENGINEER.
*
L4301.  FIND OWNER WITHIN PROJ-ENGI.
        IF  NOTFOUND NOT = 'YES'  GO TO L4303.
L4302.  MOVE FALSE TO LFOUND.
        GO TO L4201.
L4303.
        GET PROJECT.
        MOVE TRUE TO LFOUND.
*
L4401.  IF  L4410 = TRUE  GO TO L4402.
        MOVE FALSE TO LFOUND.
        IF  PROJ-SPON IS EMPTY  GO TO L4201.
        MOVE TRUE  TO L4410.
        MOVE FALSE TO L4411.        GO TO L4403.
L4402.
        IF  L4411 = TRUE  OR  LFOUND = TRUE
                  MOVE TRUE  TO L4411
              ELSE MOVE FALSE TO L4411.
L4403.  FIND NEXT PROJ-SPON-LINK WITHIN PROJ-SPON.
        IF  ENDSET NOT = 'YES'  GO TO L4404.
        MOVE L4411 TO LFOUND.
        MOVE FALSE TO L4410.
        IF  L4411 = FALSE  GO TO L4201.
                           GO TO L4701.
L4404.  GET PROJ-SPON-LINK.
*
L4601.  FIND OWNER WITHIN SPON-PROJ.

        IF  NOTFOUND = 'NO'  GO TO L4603.
L4602.  MOVE FALSE TO LFOUND.
        GO TO L4401.
L4603.
        GET SPONSOR.
        MOVE TRUE  TO LFOUND.
        GO TO L4401.

L4701.  IF  L4710 = TRUE  GO TO L4702.
        MOVE FALSE TO LFOUND.
        IF  PROJ-KEYW IS EMPTY  GO TO L4201.
        MOVE TRUE  TO L4710.
        MOVE FALSE TO L4711.        GO TO L4703.
L4702.
        IF  L4711 = TRUE  OR  LFOUND = TRUE
                  MOVE TRUE  TO L4711
              ELSE MOVE FALSE TO L4711.
L4703.  FIND NEXT PROJ-KEYW-LINK WITHIN PROJ-KEYW.
        IF  ENDSET NOT = 'YES'  GO TO L4704.
        MOVE L4711 TO LFOUND.
        MOVE FALSE TO L4710.
        IF  L4711 = FALSE  GO TO L4201.
                           GO TO L4201.
L4704.  GET PROJ-KEYW-LINK.
        CALL  RESULT (KEYWORD ).
*
L4801.  FIND OWNER WITHIN KEYW-PROJ.
        IF  NOTFOUND = 'NO'  GO TO L4803.
L4802.  MOVE FALSE TO LFOUND.
        GO TO L4701.
L4803.
        GET KEYWORDS.
        MOVE TRUE  TO LFOUND.
        GO TO L4701.
```

図 5.4 8　DMLプログラム(5)

```
L4901.  IF  L4910 = TRUE  GO TO L4902.
        MOVE FALSE TO LFOUND.
        IF  PRIOR-PROJ IS EMPTY  GO TO L0901.
        MOVE TRUE  TO L4910.
        MOVE FALSE TO L4911.        GO TO L4903.
L4902.
        IF  L4911 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L4911
            ELSE MOVE FALSE TO L4911.
L4903.  FIND NEXT PROJ-LINK WITHIN PRIOR-PROJ.
        IF  ENDSET NOT = 'YES'  GO TO L4904.
        MOVE L4911 TO LFOUND.
        MOVE FALSE TO L4910.
        IF  L4911 = FALSE  GO TO L0901.
                           GO TO L0901.
L4904.  GET PROJ-LINK.
*
L5101.  FIND OWNER WITHIN LAST-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L5103.
L5102.  MOVE FALSE TO LFOUND.
        GO TO L4901.
L5103.
        GET PROJECT.

        MOVE TRUE TO LFOUND.
*
L5201.  FIND OWNER WITHIN REPR-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L5203.
L5202.  MOVE FALSE TO LFOUND.
        GO TO L4901.
L5203.
        GET REPRESENTATIVE.
        MOVE TRUE TO LFOUND.
*
L5301.  IF  L5310 = TRUE  GO TO L5302.
        MOVE FALSE TO LFOUND.
        IF  REPR-ENGI IS EMPTY  GO TO L4901.
        MOVE TRUE  TO L5310.
        MOVE FALSE TO L5311.        GO TO L5303.
L5302.
        IF  L5311 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L5311
            ELSE MOVE FALSE TO L5311.
L5303.  FIND NEXT ENGINEER WITHIN REPR-ENGI.
        IF  ENDSET NOT = 'YES'  GO TO L5304.
        MOVE L5311 TO LFOUND.
        MOVE FALSE TO L5310.
        IF  L5311 = FALSE  GO TO L4901.
                           GO TO L4901.
L5304.  GET ENGINEER.
*
L5401.  FIND OWNER WITHIN PROJ-ENGI.
        IF  NOTFOUND = 'NO'  GO TO L5403.
L5402.  MOVE FALSE TO LFOUND.
        GO TO L5301.
L5403.
        GET PROJECT.
        MOVE TRUE  TO LFOUND.
        GO TO L5301.
```

図5.48　DMLプログラム(6)

```
L5501.  IF  L5510 = TRUE  GO TO L5502.
        MOVE FALSE TO LFOUND.
        IF  LAST-PROJ IS EMPTY  GO TO L0201.
        MOVE TRUE  TO L5510.
        MOVE FALSE TO L5511.        GO TO L5503.
L5502.
        IF  L5511 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L5511
            ELSE MOVE FALSE TO L5511.
L5503.  FIND NEXT PROJ-LINK WITHIN LAST-PROJ.
        IF  ENDSET NOT = 'YES'  GO TO L5504.
        MOVE L5511 TO LFOUND.
        MOVE FALSE TO L5510.
        IF  L5511 = FALSE  GO TO L0201.
                           GO TO L0201.
L5504.  GET PROJ-LINK.
*
L5701.  FIND OWNER WITHIN PRIOR-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L5703.
L5702.  MOVE FALSE TO LFOUND.

        GO TO L5501.
L5703.
        GET PROJECT.
        MOVE TRUE TO LFOUND.
*
L5801.  FIND OWNER WITHIN REPR-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L5803.
L5802.  MOVE FALSE TO LFOUND.
        GO TO L5501.
L5803.
        GET REPRESENTATIVE.
        MOVE TRUE TO LFOUND.
*
L5901.  IF  L5910 = TRUE  GO TO L5902.
        MOVE FALSE TO LFOUND.
        IF  REPR-ENGI IS EMPTY  GO TO L5501.
        MOVE TRUE  TO L5910.
        MOVE FALSE TO L5911.        GO TO L5903.
L5902.
        IF  L5911 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L5911
            ELSE MOVE FALSE TO L5911.
L5903.  FIND NEXT ENGINEER WITHIN REPR-ENGI.
        IF  ENDSET NOT = 'YES'  GO TO L5904.
        MOVE L5911 TO LFOUND.
        MOVE FALSE TO L5910.
        IF  L5911 = FALSE  GO TO L5501.
                           GO TO L5501.
L5904.  GET ENGINEER.
*
L6001.  FIND OWNER WITHIN PROJ-ENGI.
        IF  NOTFOUND = 'NO'  GO TO L6003.
L6002.  MOVE FALSE TO LFOUND.
        GO TO L5901.
L6003.
        GET PROJECT.
        MOVE TRUE  TO LFOUND.
        GO TO L5901.
```

図5.4.8　DMLプログラム(7)

4) テスト104

```
@T;
RANGE OF ( P1 )      ( PROJECT        );
RANGE OF ( P2 )      ( PROJECT        );
RANGE OF ( P3 )      ( PROJECT        );
RANGE OF ( R1 )      ( REPRESENTATIVE );
RANGE OF ( R2 )      ( REPRESENTATIVE );
RANGE OF ( E1 )      ( ENGINEER       );
RANGE OF ( RP1, RP2, RP3, RP4) (REPR-PROJ, REPR-PROJ, REPR-PROJ, REPR-PROJ);
RANGE OF ( PE1, PE2, PE3, PE4) (PROJ-ENGI, PROJ-ENGI, PROJ-ENGI, PROJ-ENGI);
RANGE OF ( RE1, RE2, RE3, RE4) (REPR-ENGI, REPR-ENGI, REPR-ENGI, REPR-ENGI);
RANGE OF ( PL1, PL2, PL3, PL4) (PROJ-LINK, PROJ-LINK, PROJ-LINK, PROJ-LINK);
RANGE OF ( RP5, RP6 )          ( REPR-PROJ, REPR-PROJ     );
RANGE OF ( RP7 )     ( REPR-PROJ      );
RANGE OF ( RP8 )     ( REPR-PROJ      );
RANGE OF ( RP9 )     ( REPR-PROJ      );
RANGE OF ( RP10)     ( REPR-PROJ      );
RANGE OF ( PL5 )     ( PROJ-LINK      );
RANGE OF ( PL6 )     ( PROJ-LINK      );


RETRIEVE INTO TEST104 ( P1.PROJ-NO )
WHERE
     R1.REPR-NO   = RP3.REPR-NO    AND  RP3.PROJ-NO   = P2.PROJ-NO    AND
     E1.ENGINAME  = PE3.ENGINAME   AND  PE3.PROJ-NO   = P1.PROJ-NO    AND
     R1.REPR-NO   = RP3.REPR-NO    AND  RP3.PROJ-NO   = P2.PROJ-NO    AND
     R2.REPR-NO   = RP4.REPR-NO    AND  RP4.PROJ-NO   = P2.PROJ-NO    AND
     R1.REPR-NO   = RE1.REPR-NO    AND  RE1.ENGINAME  = E1.ENGINAME   AND
     R1.REPR-NO   = RP1.REPR-NO    AND  RP1.PROJ-NO   = P1.PROJ-NO    AND
     P1.PROJ-NO   = PL1.PRIOR-NO   AND  PL1.PROJ-NO   = P2.PROJ-NO    AND
     P2.PROJ-NO   = PL2.PRIOR-NO   AND  PL2.PROJ-NO   = P3.PROJ-NO    AND
     P3.PROJ-NO   = RP2.PROJ-NO    AND  PE1.PROJ-NO   = P2.PROJ-NO    AND
     E1.ENGINAME  = PE1.ENGINAME   AND  RP2.REPR-NO   = R2.REPR-NO    AND
     E1.ENGINAME  = RE2.ENGINAME   AND  RE2.REPR-NO   = R2.REPR-NO
                                   AND
     R1.REPR-NO   = RE3.REPR-NO    AND  RE3.ENGINAME  = E1.ENGINAME   AND
     R1.REPR-NO   = RP5.REPR-NO    AND  RP5.PROJ-NO   = P1.PROJ-NO    AND
     P1.PROJ-NO   = PL3.PRIOR-NO   AND  PL3.PROJ-NO   = P2.PROJ-NO    AND
     P2.PROJ-NO   = PL4.PRIOR-NO   AND  PL4.PROJ-NO   = P3.PROJ-NO    AND
     R2.REPR-NO   = RP6.REPR-NO    AND  RP6.PROJ-NO   = P3.PROJ-NO    AND
     R2.REPR-NO   = RE4.REPR-NO    AND  RE4.ENGINAME  = E1.ENGINAME
                                   AND
     R1.REPR-NO   = RP7.REPR-NO    AND  RP7.PROJ-NO   = P3.PROJ-NO    AND
     R1.REPR-NO   = RP8.REPR-NO    AND  RP8.PROJ-NO   = P3.PROJ-NO    AND
     R2.REPR-NO   = RP9.REPR-NO    AND  RP9.PROJ-NO   = P1.PROJ-NO    AND
     R2.REPR-NO   = RP10.REPR-NO   AND  RP10.PROJ-NO  = P1.PROJ-NO    AND
     P1.PROJ-NO   = PL5.PRIOR-NO   AND  PL5.PROJ-NO   = P3.PROJ-NO    AND
     P1.PROJ-NO   = PL6.PRIOR-NO   AND  PL6.PROJ-NO   = P3.PROJ-NO
     ;
```

図6.49 テスト104のLCS問合せ

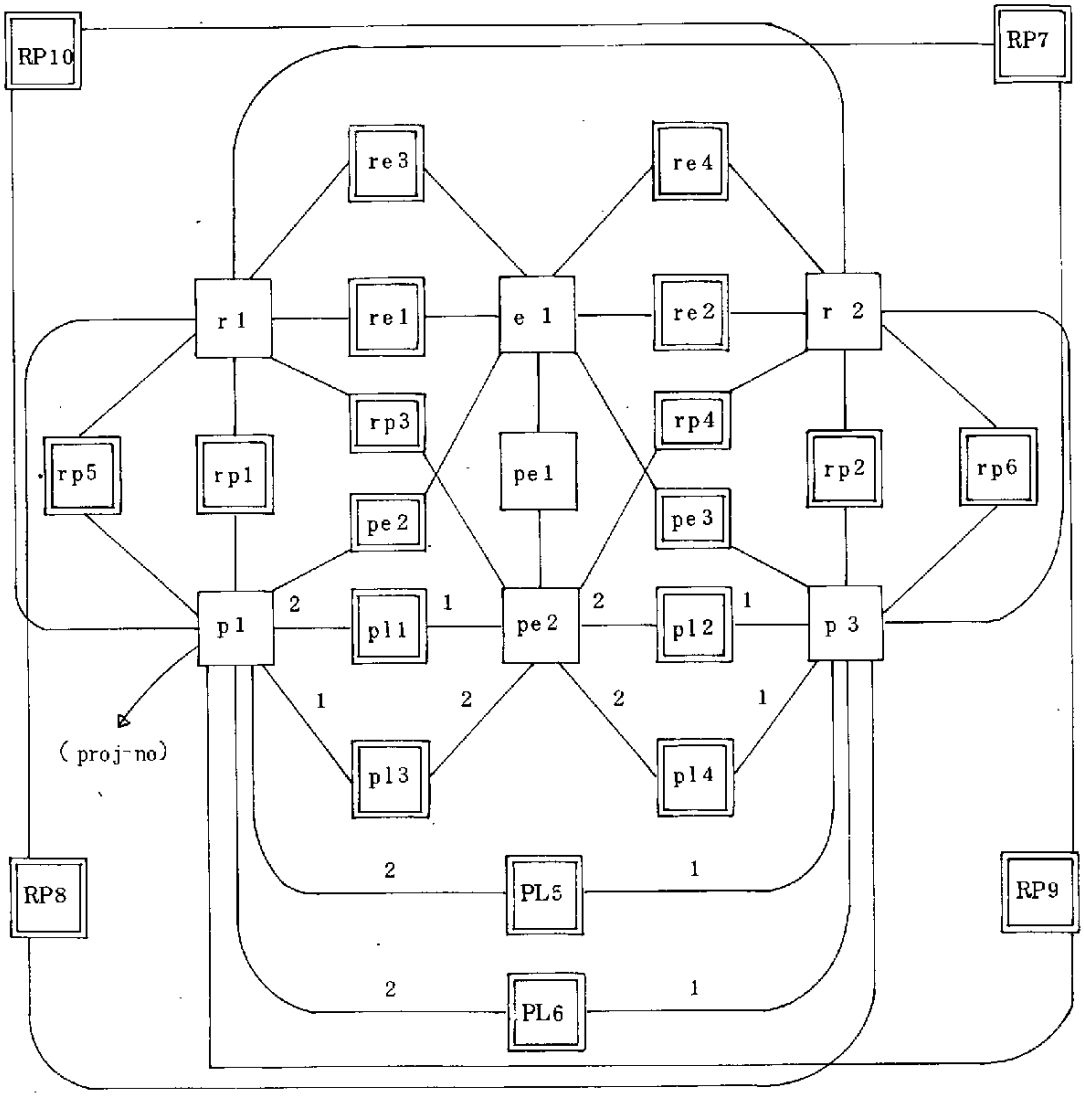


図5.50　テスト104のRQG

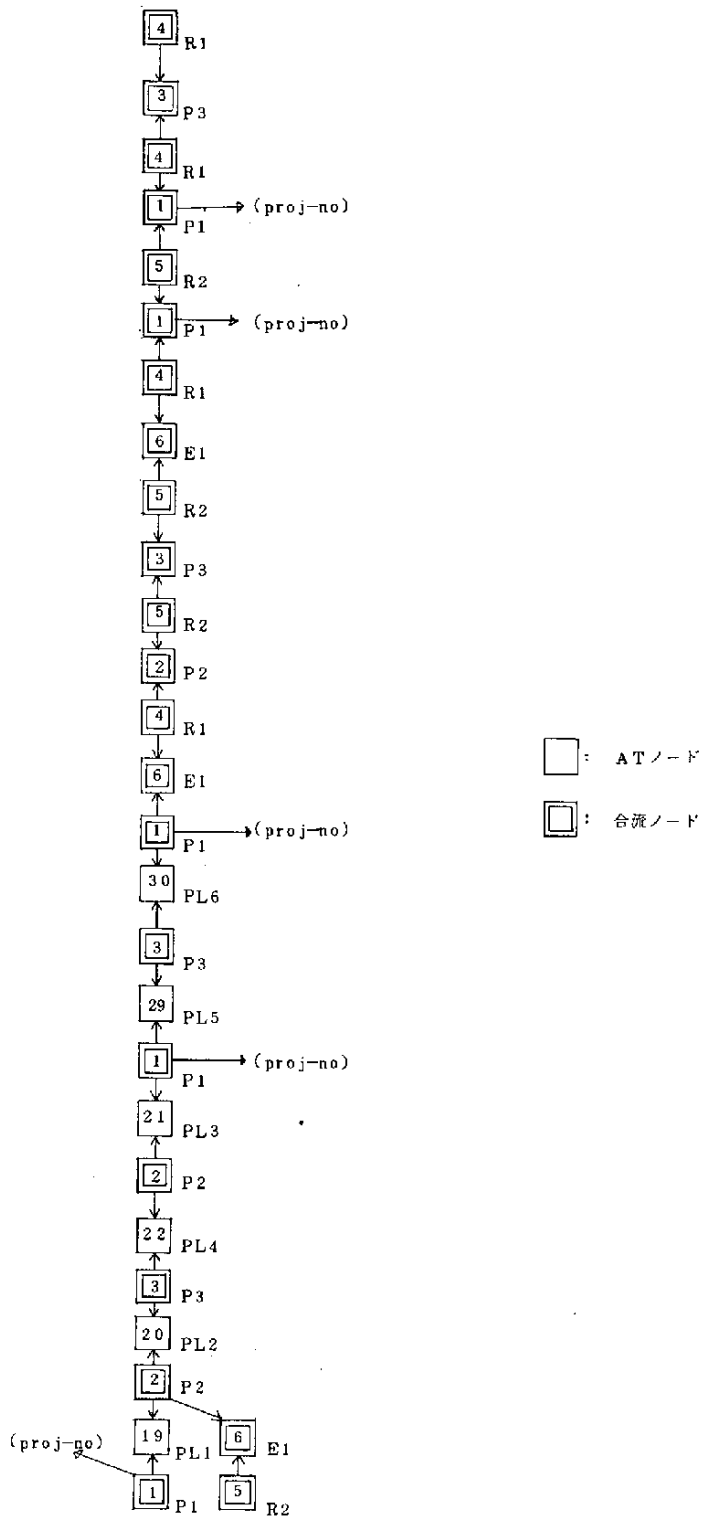


図5.5.1　テスト104のアクセス木

```
GO;
      *---------- DBTG DML ----------
              MOVE FALSE TO LFOUND.
              FIND FIRST REPRESENTATIVE.
              GO TO L0102.
 L0101.
              FIND REPRESENTATIVE DB-KEY IS L0120.
              FIND NEXT REPRESENTATIVE.
 L0102.  IF  ENDSET = 'YES'  GO TO TERM.
              ACCEPT L0120 FROM CURRENCY.

              GET REPRESENTATIVE.
      *
 L0201.  IF  L0210 = TRUE  GO TO L0202.
              MOVE FALSE TO LFOUND.
              IF  REPR-PROJ IS EMPTY  GO TO L0101.
              MOVE TRUE  TO L0210.
              MOVE FALSE TO L0211.        GO TO L0203.
 L0202.
              IF  L0211 = TRUE  OR  LFOUND = TRUE
                        MOVE TRUE  TO L0211
                   ELSE MOVE FALSE TO L0211.
              FIND PROJECT DB-KEY IS L0220.
 L0203.
              FIND NEXT PROJECT WITHIN REPR-PROJ.
              IF  ENDSET NOT = 'END'  GO TO L0204.
              MOVE L0211 TO LFOUND.
              MOVE FALSE TO L0210.
              IF  L0211 = FALSE  GO TO L0101.
                                 GO TO L0101.
 L0204.
              ACCEPT L0220 FROM REPR-PROJ CURRENCY.
              GET PROJECT.
      *
 L0901.  FIND OWNER WITHIN REPR-PROJ.
              IF  NOTFOUND NOT = 'YES'  GO TO L0903.
 L0902.  MOVE FALSE TO LFOUND.
              GO TO L0201.
 L0903.
              GET REPRESENTATIVE.
              MOVE TRUE TO LFOUND.
      *
 L1601.  IF  L1610 = TRUE  GO TO L1602.
              MOVE FALSE TO LFOUND.
              IF  REPR-PROJ IS EMPTY  GO TO L0201.
              MOVE TRUE  TO L1610.
              MOVE FALSE TO L1611.        GO TO L1603.
 L1602.
              IF  L1611 = TRUE  OR  LFOUND = TRUE
                        MOVE TRUE  TO L1611
                   ELSE MOVE FALSE TO L1611.
              FIND PROJECT DB-KEY IS L1620.
 L1603.
              FIND NEXT PROJECT WITHIN REPR-PROJ.
              IF  ENDSET NOT = 'END'  GO TO L1604.
              MOVE L1611 TO LFOUND.
              MOVE FALSE TO L1610.
              IF  L1611 = FALSE  GO TO L0201.
                                 GO TO L0201.
 L1604.
              ACCEPT L1620 FROM REPR-PROJ CURRENCY.
              GET PROJECT.
              CALL  RESULT (PROJ-NO ).
      *
 L2101.  FIND OWNER WITHIN REPR-PROJ.
              IF  NOTFOUND NOT = 'YES'  GO TO L2103.
 L2102.  MOVE FALSE TO LFOUND.
```

図 5.5 2　DML プログラム(1)

```
           GO TO L1601.
 L2103.
           GET REPRESENTATIVE.
           MOVE TRUE TO LFOUND.
 *
 L2901.    IF  L2910 = TRUE  GO TO L2902.
           MOVE FALSE TO LFOUND.
           IF  REPR-PROJ IS EMPTY  GO TO L1601.
           MOVE TRUE  TO L2910.
           MOVE FALSE TO L2911.        GO TO L2903.
 L2902.
           IF  L2911 = TRUE  OR  LFOUND = TRUE
                     MOVE TRUE  TO L2911
                ELSE MOVE FALSE TO L2911.
           FIND PROJECT DB-KEY IS L2920.
 L2903.
           FIND NEXT PROJECT WITHIN REPR-PROJ.
           IF  ENDSET NOT = 'END'  GO TO L2904.
           MOVE L2911 TO LFOUND.
           MOVE FALSE TO L2910.
           IF  L2911 = FALSE  GO TO L1601.
                              GO TO L1601.
 L2904.
           ACCEPT L2920 FROM REPR-PROJ CURRENCY.
           GET PROJECT.
           CALL  RESULT (PROJ-NO ).
 *
 L3501.    FIND OWNER WITHIN REPR-PROJ.
           IF  NOTFOUND NOT = 'YES'  GO TO L3503.
 L3502.    MOVE FALSE TO LFOUND.
           GO TO L2901.
 L3503.
           GET REPRESENTATIVE.
           MOVE TRUE TO LFOUND.
 *
 L4101.    IF  L4110 = TRUE  GO TO L4102.
           MOVE FALSE TO LFOUND.
           IF  REPR-ENGI IS EMPTY  GO TO L2901.
           MOVE TRUE  TO L4110.
           MOVE FALSE TO L4111.        GO TO L4103.
 L4102.
           IF  L4111 = TRUE  OR  LFOUND = TRUE
                     MOVE TRUE  TO L4111
                ELSE MOVE FALSE TO L4111.
           FIND ENGINEER DB-KEY IS L4120.
 L4103.
           FIND NEXT ENGINEER WITHIN REPR-ENGI.
           IF  ENDSET NOT = 'END'  GO TO L4104.
           MOVE L4111 TO LFOUND.
           MOVE FALSE TO L4110.
           IF  L4111 = FALSE  GO TO L2901.
                              GO TO L2901.
 L4104.
           ACCEPT L4120 FROM REPR-ENGI CURRENCY.
           GET ENGINEER.
 *
 L4401.    FIND OWNER WITHIN REPR-ENGI.
           IF  NOTFOUND NOT = 'YES'  GO TO L4403.
 L4402.    MOVE FALSE TO LFOUND.
           GO TO L4101.
 L4403.
           GET REPRESENTATIVE.
           MOVE TRUE TO LFOUND.
 *
```

図5.5 2　DMLプログラム(2)

```
L4901.  IF  L4910 = TRUE  GO TO L4902.
        MOVE FALSE TO LFOUND.
        IF  REPR-PROJ IS EMPTY  GO TO L4101.
        MOVE TRUE  TO L4910.
        MOVE FALSE TO L4911.        GO TO L4903.
L4902.
        IF  L4911 = TRUE  OR  LFOUND = TRUE
                    MOVE TRUE  TO L4911
              ELSE MOVE FALSE TO L4911.
        FIND PROJECT DB-KEY IS L4920.
L4903.
        FIND NEXT PROJECT WITHIN REPR-PROJ.
        IF  ENDSET NOT = 'END'  GO TO L4904.
        MOVE L4911 TO LFOUND.
        MOVE FALSE TO L4910.
        IF  L4911 = FALSE  GO TO L4101.
                           GO TO L4101.
L4904.
        ACCEPT L4920 FROM REPR-PROJ CURRENCY.
        GET PROJECT.
*
L5301.  FIND OWNER WITHIN REPR-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L5303.
L5302.  MOVE FALSE TO LFOUND.
        GO TO L4901.
L5303.
        GET REPRESENTATIVE.
        MOVE TRUE TO LFOUND.
*
L5801.  IF  L5810 = TRUE  GO TO L5802.
        MOVE FALSE TO LFOUND.
        IF  REPR-PROJ IS EMPTY  GO TO L4901.
        MOVE TRUE  TO L5810.
        MOVE FALSE TO L5811.        GO TO L5803.
L5802.
        IF  L5811 = TRUE  OR  LFOUND = TRUE
                    MOVE TRUE  TO L5811
              ELSE MOVE FALSE TO L5811.
        FIND PROJECT DB-KEY IS L5820.
L5803.
        FIND NEXT PROJECT WITHIN REPR-PROJ.
        IF  ENDSET NOT = 'END'  GO TO L5804.
        MOVE L5811 TO LFOUND.
        MOVE FALSE TO L5810.
        IF  L5811 = FALSE  GO TO L4901.
                           GO TO L4901.
L5804.
        ACCEPT L5820 FROM REPR-PROJ CURRENCY.
        GET PROJECT.
*
L6001.  FIND OWNER WITHIN REPR-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L6003.
L6002.  MOVE FALSE TO LFOUND.
        GO TO L5801.
L6003.
        GET REPRESENTATIVE.
        MOVE TRUE TO LFOUND.
*
L6601.  IF  L6610 = TRUE  GO TO L6602.
        MOVE FALSE TO LFOUND.
        IF  REPR-ENGI IS EMPTY  GO TO L5801.
        MOVE TRUE  TO L6610.
        MOVE FALSE TO L6611.        GO TO L6603.
L6602.
        IF  L6611 = TRUE  OR  LFOUND = TRUE
                    MOVE TRUE  TO L6611
              ELSE MOVE FALSE TO L6611.
        FIND ENGINEER DB-KEY IS L6620.
```

図5.52　DMLプログラム(3)

```
L6603.
        FIND NEXT ENGINEER WITHIN REPR-ENGI.
        IF  ENDSET NOT = 'END'  GO TO L6604.
        MOVE L6611 TO LFOUND.
        MOVE FALSE TO L6610.
        IF  L6611 = FALSE  GO TO L5801.
                           GO TO L5801.
L6604.
        ACCEPT L6620 FROM REPR-ENGI CURRENCY.
        GET ENGINEER.
*
L6701.  FIND OWNER WITHIN PROJ-ENGI.
        IF  NOTFOUND NOT = 'YES'  GO TO L6703.
L6702.  MOVE FALSE TO LFOUND.
        GO TO L6601.
L6703.
        GET PROJECT.
        MOVE TRUE TO LFOUND.
        CALL  RESULT (PROJ-NO ).
*
L7001.  IF  L7010 = TRUE  GO TO L7002.
        MOVE FALSE TO LFOUND.
        IF  LAST-PROJ IS EMPTY  GO TO L6601.
        MOVE TRUE  TO L7010.
        MOVE FALSE TO L7011.        GO TO L7003.
L7002.
        IF  L7011 = TRUE  OR  LFOUND = TRUE
                  MOVE TRUE  TO L7011
              ELSE MOVE FALSE TO L7011.
L7003.  FIND NEXT PROJ-LINK WITHIN LAST-PROJ.
        IF  ENDSET NOT = 'YES'  GO TO L7004.
        MOVE L7011 TO LFOUND.
        MOVE FALSE TO L7010.
        IF  L7011 = FALSE  GO TO L6601.
                           GO TO L6601.
L7004.  GET PROJ-LINK.
*
L7401.  FIND OWNER WITHIN PRIOR-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L7403.
L7402.  MOVE FALSE TO LFOUND.
        GO TO L7001.
L7403.
        GET PROJECT.
        MOVE TRUE TO LFOUND.
*
L7501.  IF  L7510 = TRUE  GO TO L7502.
        MOVE FALSE TO LFOUND.
        IF  PRIOR-PROJ IS EMPTY  GO TO L7001.
        MOVE TRUE  TO L7510.
        MOVE FALSE TO L7511.        GO TO L7503.
L7502.
        IF  L7511 = TRUE  OR  LFOUND = TRUE
                  MOVE TRUE  TO L7511
              ELSE MOVE FALSE TO L7511.
L7503.  FIND NEXT PROJ-LINK WITHIN PRIOR-PROJ.
        IF  ENDSET NOT = 'YES'  GO TO L7504.
        MOVE L7511 TO LFOUND.
        MOVE FALSE TO L7510.
        IF  L7511 = FALSE  GO TO L7001.
                           GO TO L7001.
L7504.  GET PROJ-LINK.
*
L7801.  FIND OWNER WITHIN LAST-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L7803.
L7802.  MOVE FALSE TO LFOUND.
        GO TO L7501.
L7803.
        GET PROJECT.
        MOVE TRUE TO LFOUND.
        CALL  RESULT (PROJ-NO ).
*
```

図 5.5 2　DMLプログラム(4)

```
L7901.  IF  L7910 = TRUE  GO TO L7902.
        MOVE FALSE TO LFOUND.
        IF  LAST-PROJ IS EMPTY  GO TO L7501.
        MOVE TRUE  TO L7910.
        MOVE FALSE TO L7911.        GO TO L7903.
L7902.
        IF  L7911 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L7911
            ELSE MOVE FALSE TO L7911.
L7903.  FIND NEXT PROJ-LINK WITHIN LAST-PROJ.
        IF  ENDSET NOT = 'YES'  GO TO L7904.
        MOVE L7911 TO LFOUND.
        MOVE FALSE TO L7910.
        IF  L7911 = FALSE  GO TO L7501.
                           GO TO L7501.
L7904.  GET PROJ-LINK.
*
L8101.  FIND OWNER WITHIN PRIOR-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L8103.
L8102.  MOVE FALSE TO LFOUND.
        GO TO L7901.

L8103.
        GET PROJECT.
        MOVE TRUE TO LFOUND.
*
L8201.  IF  L8210 = TRUE  GO TO L8202.
        MOVE FALSE TO LFOUND.
        IF  LAST-PROJ IS EMPTY  GO TO L7901.
        MOVE TRUE  TO L8210.
        MOVE FALSE TO L8211.        GO TO L8203.
L8202.
        IF  L8211 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L8211
            ELSE MOVE FALSE TO L8211.
L8203.  FIND NEXT PROJ-LINK WITHIN LAST-PROJ.
        IF  ENDSET NOT = 'YES'  GO TO L8204.
        MOVE L8211 TO LFOUND.
        MOVE FALSE TO L8210.
        IF  L8211 = FALSE  GO TO L7901.
                           GO TO L7901.
L8204.  GET PROJ-LINK.
*
L8601.  FIND OWNER WITHIN PRIOR-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L8603.
L8602.  MOVE FALSE TO LFOUND.
        GO TO L8201.
L8603.
        GET PROJECT.
        MOVE TRUE TO LFOUND.
*
L8701.  IF  L8710 = TRUE  GO TO L8702.
        MOVE FALSE TO LFOUND.
        IF  PRIOR-PROJ IS EMPTY  GO TO L8201.
        MOVE TRUE  TO L8710.
        MOVE FALSE TO L8711.        GO TO L8703.
L8702.
        IF  L8711 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L8711
            ELSE MOVE FALSE TO L8711.
L8703.  FIND NEXT PROJ-LINK WITHIN PRIOR-PROJ.
        IF  ENDSET NOT = 'YES'  GO TO L8704.
        MOVE L8711 TO LFOUND.
        MOVE FALSE TO L8710.
        IF  L8711 = FALSE  GO TO L8201.
                           GO TO L8201.
L8704.  GET PROJ-LINK.
*
```

図 5.5 2　DMLプログラム(5)

```
L8801.  FIND OWNER WITHIN LAST-PROJ.
        IF  NOTFOUND NOT = 'YES'  GO TO L8803.
L8802.  MOVE FALSE TO LFOUND.
        GO TO L8701.
L8803.
        GET PROJECT.
        MOVE TRUE TO LFOUND.
*
L8901.  IF  L8910 = TRUE  GO TO L8902.

        MOVE FALSE TO LFOUND.
        IF  PRIOR-PROJ IS EMPTY  GO TO L8701.
        MOVE TRUE  TO L8910.
        MOVE FALSE TO L8911.        GO TO L8903.
L8902.
        IF  L8911 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L8911
            ELSE MOVE FALSE TO L8911.
L8903.  FIND NEXT PROJ-LINK WITHIN PRIOR-PROJ.
        IF  ENDSET NOT = 'YES'  GO TO L8904.
        MOVE L8911 TO LFOUND.
        MOVE FALSE TO L8910.
        IF  L8911 = FALSE  GO TO L8701.
                           GO TO L9201.
L8904.  GET PROJ-LINK.
*
L9101.  FIND OWNER WITHIN LAST-PROJ.
        IF  NOTFOUND = 'NO'  GO TO L9103.
L9102.  MOVE FALSE TO LFOUND.
        GO TO L8901.
L9103.
        GET PROJECT.
        MOVE TRUE  TO LFOUND.
        CALL  RESULT (PROJ-NO ).
        GO TO L8901.

L9201.  IF  L9210 = TRUE  GO TO L9202.
        MOVE FALSE TO LFOUND.
        IF  PROJ-ENGI IS EMPTY  GO TO L8701.
        MOVE TRUE  TO L9210.
        MOVE FALSE TO L9211.        GO TO L9203.
L9202.
        IF  L9211 = TRUE  OR  LFOUND = TRUE
                MOVE TRUE  TO L9211
            ELSE MOVE FALSE TO L9211.
        FIND ENGINEER DB-KEY IS L9220.
L9203.
        FIND NEXT ENGINEER WITHIN PROJ-ENGI.
        IF  ENDSET NOT = 'END'  GO TO L9204.
        MOVE L9211 TO LFOUND.
        MOVE FALSE TO L9210.
        IF  L9211 = FALSE  GO TO L8701.
                           GO TO L8701.
L9204.
        ACCEPT L9220 FROM PROJ-ENGI CURRENCY.
        GET ENGINEER.
*
L9301.  FIND OWNER WITHIN REPR-ENGI.
        IF  NOTFOUND = 'NO'  GO TO L9303.
L9302.  MOVE FALSE TO LFOUND.
        GO TO L9201.
L9303.
        GET REPRESENTATIVE.
        MOVE TRUE  TO LFOUND.
        GO TO L9201.
```

図5.5.2　DMLプログラム(6)

5.6 QTP

QTP（query translation processor）は、これまで述べてきた問合せ変換アルゴリズムに基づいて、インプリメントされたシステムである。これの詳細は8.2節において述べられている。QTPは、QUELによって書かれたLCS問合せ（LQ）を入力として、DBTG DMLプログラムを出力するシステムである〔図5.5.3〕。QTPの必要とする異種性情報(HI)〔4.5を参照〕は、HIPS〔4.7又は8.1を参照〕によって生成されたものである。

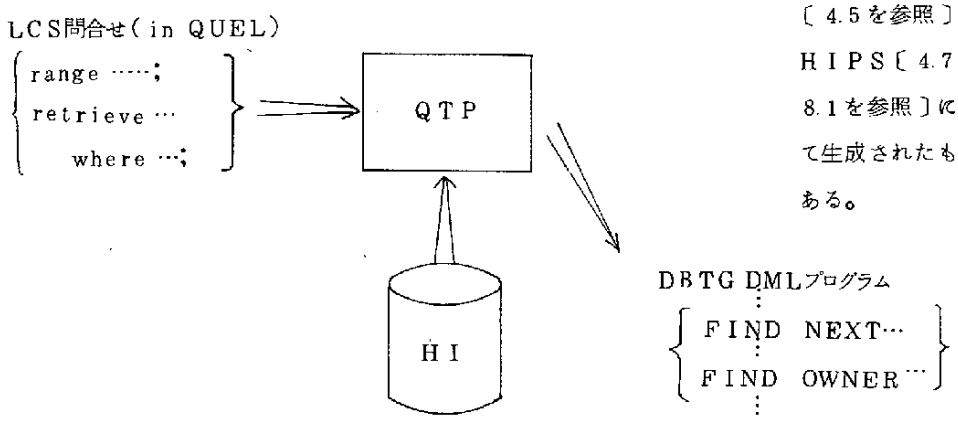


図5.5.3　QTPの概要

QTPは、本章で論じてきた様に3つの主要モヂュールからなっている。これらは問合せの構造変換、最適化（アクセス木生成）、DML生成を行なう3つのモヂュールである。第1の問合せの構造変換では、問合せをグラフ表現して、その変形を行なうことによって目的とするDBTG構造による問合せ表現（DBTG問合せグラフ）を得ると述べた〔5.3〕。DBTG問合せグラフ（DQG）は、RTBとSTBと呼ばれる2つのテーブル（リレーション）によって表わされる。さらにこれらは総称してQDBTGRと呼ばれる。RTBは各DQGノードについて、対応するレコード型、それに関する制限式（restriction formula）結果属性カーディナリティ及び制限式の選択度等の情報を維持している。一方STBは各DQGアークに対応するセット型と、これの親と子レコード型か何かを示している。さらに問合せの条件式と結果属性のリストとは、2分木によって内部表現される。

例えば、ESR A(a1, a2, a3) RSR R(a1, b1)　に対する次のようなLCS問
B(b1, b2)
合せ（LQ）を考えてみよう。

```
        range   (a, b, r)   (A, B, R);
LQ:     retrieve into   R   (a1, a3)
        where   a.a1 = r.a1   and   r.b1 = b.b1   and   a.a2 = " A ";
```

これは、図5.5.4のような2分木によって内部表現される。LQに対応するDBTG問合せグラフ

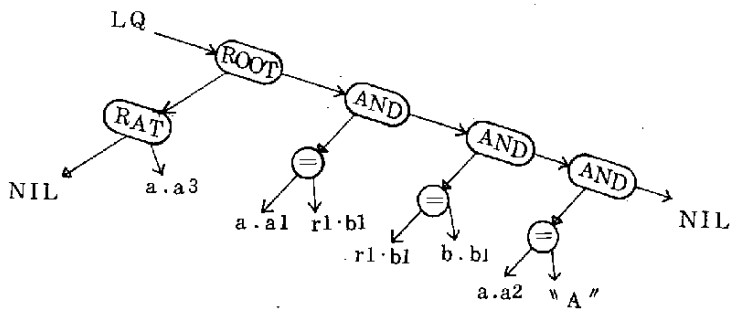


図5.5 4　　問合せの2分木表現

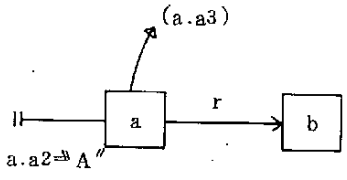


図5.5 5　図5.5 4のDQG

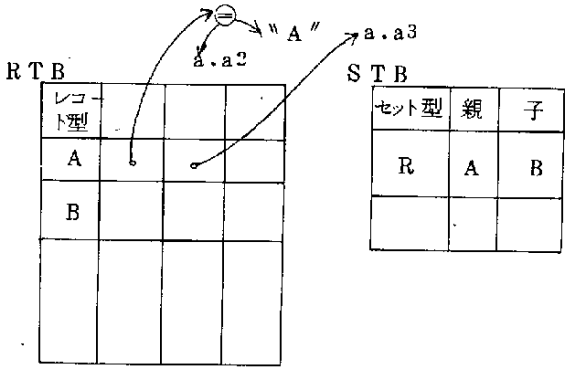


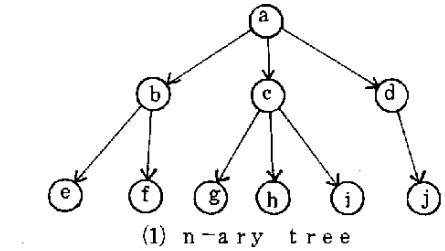
図5.5 6　DQGのRTBとSTBによる表現（QDBTGR）

は図5.5 5のようになるとする。このDQGは、QTP内で図5.5 6のようなSTBとRTBという2つのテーブルによって表わされる。RTBにはLCS問合せ（LQ）が参照するレコード型とこれに関する制限式と結果属性等の 情報が格納されている。レコード型Aは制限式a.a2=〃A〃と結果属性a.a3とをもっていることを示している。STBには、LQ内の等価結合式に対応するセット型とそれの親子レコード型の情報が格納されている。セット型Rの親レコード型はAであり、子レコード型はBであることを示している。

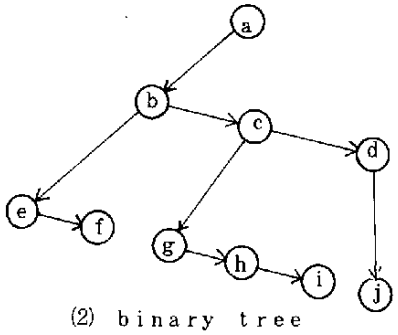
この様なDBTG問合せグラフの内部表現、即ちQDBTGRをもとに、次のアクセス木生成では、アクセス木をDFA又はBFAによって生成する。アクセス木は、NITと呼ばれる領域に2分木によって表現されて格納される。アクセス木は一般にn分木（n≧1）であるが、これを図5.5 7の様に2分木によって表現している。これの各ノード（NITノード）には対応するレコード型、AT枝に対応するセット型、ノードの種類（合流ノードかどうか、根ノードか、葉ノードか、中間ノードか）等の情報がふくまれている。即ち、このNITノードはアクセス単位を表わしている。

最後のDML生成では、このアクセス木の2分木表現を入力として、preorder に各NITノード（アクセス木のノード）をたどりながら、このNITノードに対応するDMLブロックを生成する。各DMLブロックはHIの一部として各サイトに保持されている。

QTPのインプリメンテーションには、PL／Iを用いた。しかし、これまで論じてきた様に、問合せ変換では、問合せの条件式の内部表現としての2分木の操作、DBTG問合せグラフのようなグラフ表現の操作が多く必要となる。PL／Iはこのような構造の表現、操作機能、即ち記



(1) n−ary tree



(2) binary tree

図5.57　n分木と2分木

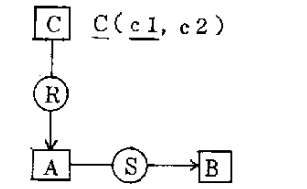
号処理機能をもたない。このため、スタック、ポインタによる木表現、木表現のため自由セルのもり集め等の機能を新たにインプリメントする必要があつた。多くのプログラミング努力がこれらの基本的記号処理能力をつくるために払われねばならなかつたことを考えると、ＬＩＳＰの様な記号処理言語によるインプリメントの方がプログラミング上は望ましかつたように思う。今後、ＬＩＳＰによるインプリメントの可能性も検討していくべきであると考える。

5.7　ＱＴＰの議論

本章ではこれまで論じてきた問合せ変換（ＱＴ）方法及びシステムについて、その能力とともに問題点について論じる。ＱＴＰは、ＰＬ／ＩでＭ－１６０上にインプリメントされている。ＱＴＰのプログラムＰＬ／Ｉ文数は約３０００　、オブジェクトサイズは１３０ＫＢである。ここでは、アクセス木生成方法として縦型方法（ＤＦＡ）についてのみ考える。Ｍ－１６０のＰＬ／Ｉは、ｃｐｕ のアカウントをとる機能がないために、実行時間についての測定はできなかつた。まず、問合せ構造変換、アクセス木生成、ＤＭＬ生成の各々について論じ、最後にＱＴＰ全体について考えることにする。

5.7.1　問合せ構造変換（ＳＴ）

結合リンクの一方のノードは必ず関係性集合リレーション（ＲＳＲ）でなければならない。この条件は、ＬＣＳ問合せ（ＬＱ）は局所内部スキーマ（ＬＩＳ）に記述されたアクセスパスに沿つて問合せが発せられねばならないことを意味している。図5.58ａ）の様なＬＩＳがあつたとしよう。このＬＩＳに対する局所概念スキーマ（ＬＣＳ）は図5.40ｂ）の様である。図5.58のｂ）に示したＬＣＳに対する問合せとして図5.59に示す様なパターンは許される。何故ならこれらの中のどの結合リンクも必ず一方には関係性集合リレーション（ＲＳＲ）を持つているとともに、その他方はＬＩＳ内にアクセスパスが存在しているものだけであるからである。しかし、図5.60に示した様



A(a1, a2)　B(b1, b2)

a）ＬＩＳの例

ＥＳＲ：A(a1、a2)

B(b1、b2)

C(c1、c2)

ＲＳＲ：S(a1、b1)

R(c1、a1)

b）　a）のＬＣＳ

図5.58　ＬＩＳとＬＣＳの例

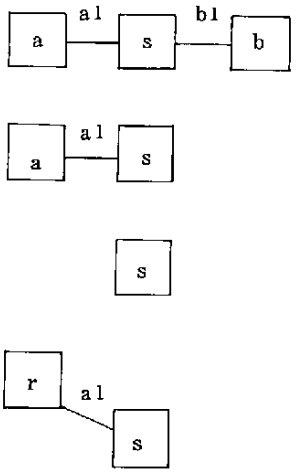


図5.59　許される問合せグラフ

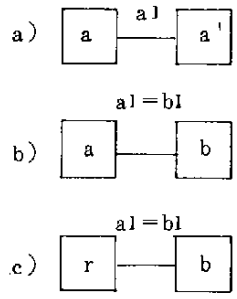


図5.60　許されない問合せグラフ

な問合せパターンは許されない。この図のａ）とｂ)とは互いに事象集合リレーション（ＥＳＲ）同志であるので許されない。これは組変数ｒはＲＳＲを表わしているが、これはレコード型Ａが隠れ構造となっているだけでｂ）と同じである。ｂ）では、レコード型Ａの属性ａ１とレコード型Ｂの属性ｂ１とが等しいということを表わすアクセスパスはＬＩＳ内には存在しない。このためこのパターンも許されない。図5.60に示した様な問合せを許すならば、我々はＬＩＳに示されているＤＢＴＧモデルが提供するアクセスパスの他に、これとは独立な集合演算機能〔ＭＵＲＯＫ８０〕が必要となってしまう。このような集合演算機能を提供することは必要になろうが、無制限にＬＣＳのリレーション間で結合演算を行なわせることには問題があるように考える。このため我々は、関係性集合リレーション（ＲＳＲ）によって結合可能なリレーションの対を示している。現在のところ、この結合可能性は２つの（事象集合）リレーション間にＤＢＴＧのアクセスパス、即ちセット型又はリンクレコード型が存在していることを意味している。

関係性リレーションから出される結合リンクは、主キー属性の各々に対してユニークに存在するだけである。即ち、図5.58において関係性集合リレーションS(a1、b1)　を考えてみよう。

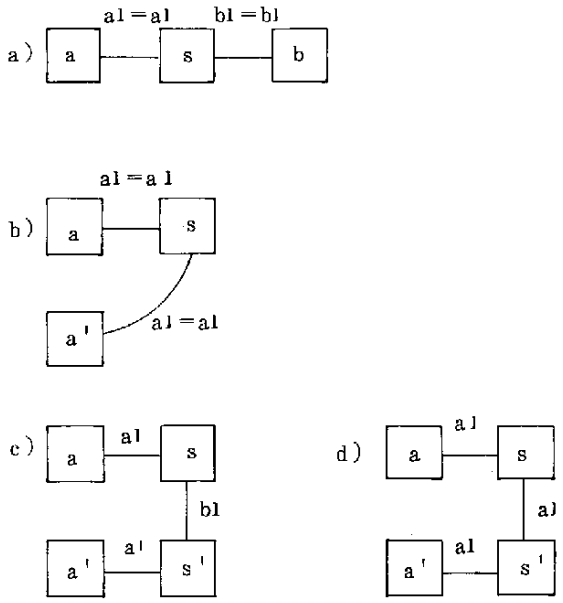


図5.6.1　　ＲＳＲの制限

Ｓに対する問合せパターンとして、図5.6.1のａ）とｃ）とは許されるが、ｂ）とｄ）とは許されない。ｂ）とｄ）との問題は図5.6.1のａ）と同じである。ｂ）とｄ）の様なパターンを許すためには前述した様に集合演算機能が必要になる。

結合述語としては等価結合（equi-join）のみが許される。不等価結合を行なうためには、やはり集合演算機能が必要となる。

この様に、ＬＣＳのＲＤに基づいたリレーショナル問合せの能力は、ＬＩＳに基づいたＤＭＬと同じものということができる。これ以上の能力をＬＣＳ問合せが持つためには、ＬＩＳの親言語によって集合演算機能をインプリメントする必要があるだろう。しかし、我々のＱＴＰをＤＢＴＧ　ＤＢＭＳへのリレーショナルインタフェースと位置づけた時、ユーザの多くの問合せは関係性集合リレーションに基づいて自分の必要な事象リレーションをたどり（navigate）ながらアクセスしていくと考えられる。又結合もその多くは等価結合であると考えられる。この意味で現在のＱＴＰでもユーザの問合せの多くの部分をカバーできるであろう。

5.7.2　アクセス木の生成

5.4.1で述べた様に、問合せ変換の目標は次の２点である。

1）　中間結果の数と量とを最少化する。

2）　アクセスされるオカーランス数を最少化する。

5.4.5で論じた縦型アクセス木生成アルゴリズム（ＤＦＡ）は、この目標を達成しようとするものである。5.4節で述べた様に、ＤＱＧからのアクセス木生成方法としては縦型アクセス木生成アルゴリズム（ＤＦＡ）が優れている。これはＤＦＡによって生成されたアクセス木は、ただ１つの結果リレーションを必要とするだけである点である。一方ＢＦＡでは合流ノードに対応して多くの中間結果リレーションを必要とし、さらにこれらのリレーション間の集合演算を必要としている。ＤＦＡによるアクセス木が、あるＤＱＧノードに対する全ての合流ノードをアクセス木内で最初に現れた合流ノードであるＡＴノードを根ノードとする部分木内に含まれるという性質を持つことは、我々の提案するＤＦＡの最大の長所である。よって我々のＤＦＡは中間結果の数を最少化できる。

中間結果が複数ある場合には、これらに対して集合演算を行なう必要がある。こうした集合演算を行なうためには、さらにこの演算のための中間結果を必要とする。更に、いくつかの中間結果内で最終的な結果となるデータはその一部である。このことは、一般に中間結果として不要なデータを格納していることになる。この様に一般的に中間結果の数を減らすことは中間結果の量を減らすことになる。DFAでは結果リレーション内に,アクセス木に基づいたアクセスパスによってオカーランスがアクセスされ条件に合うもののみが、即ち、必要なデータのみが格納される。従って、我々のDFAは中間結果の量も多くの場合最少化できる。

次に、アクセスされるオカーランス数について考えてみよう。我々のDFAは最少のアクセスされるオカーランス数を持つアクセス木を必ず生成するものではない。DBTG問合せグラフから、各DQGノードのカーディナリティ、制限式の選択度、DQGマークの結合度を用いて、可能なアクセス木群のなかから最少のアクセスオカーランス数を持つものを見つけ出すための計算量は、DQGのノードとアークの増大にともなって指数関数的に増大する。従って我々は次のようなヒューリスティックスを用いた。即ち、次のATノードを決める場合、可能なATノードのなかで最少の結合度を持つDQGアークによって隣接するATノードを選ぶ。又同一結合度を持つ複数のDQGノードがあればこの中で条件を満足するオカーランス数(OCS)が最少のものを選ぶ。このことは、1レベルの探索によって次の手を決定することである。この様に我々のヒューリスティックスは、深いレベルまでサーチして手を決定しないので極めて簡単なものである。

このヒューリスティックスによって生成されるアクセス木のあるノード(a)が複数の部分木(T1、…Tn)を持つとしよう。この時この部分木(Ti, i=1, …, n)は、部分木の根ノード内(ai)のアクセスされるオカーランス数が小さい順に右から左へと並んでいる。よって、アクセスされるオカーランス数がより小さいものをより先にアクセスすることになる。このことはaからai へ直接アクセスする時のアクセスオカーランス数が大きかったものが、アクセスパスの中でTiより左手の部分木Tj(j=1, 2,…, i−1)のアクセスによってアクセスされるオカーランス数を制限できる〔5.4.2〕。このことは、同一ノードに対する部分木間の関係においてだけではなく、同一部分木内の異なったレベルにあるATノード間においてもいえる。即ち、より小さなアクセスオカーランス数を持つものをよりアクセス木の上位におくことによってアクセスされるオカーランス数を制限できる。

以上のことより我々のDFAに用いられたヒューリスティックスは、簡単であるとともに有効であると結論できる。

我々が選択度、結合度を用いてより複雑なアクセス木生成を行なわないもう1つの重要な理由がある。それは選択度(selectivity)、結合度(connectivity)といったDBTGデータベース内のオカーランスについての統計情報がどれだけ実際のアクセスを予測できるかという点である。選択度はあるリレーション(レコード型)内ある属性(データ項目)の値がこのリレーション内でユニークであるものの数をこのリレーションのカーディナリティとの比として表わされる。

これには属性値がこのリレーション内で均等に分散（ｅｖｅｎ　ｄｉｓｔｒｉｂｕｔｉｏｎ）していることが前提となっている。しかし、実際の値はこの様に分散しているであろうか。このことは結合度に対しても同様に言える。即ち、ＤＱＧアークの結合度とはこのアークが１方のＤＱＧノード内の１つのオカーランスに対して、他方のＤＱＧノード内の平均何個のオカーランスをリンクしているかである。我々はこの様に定義された選択度と結合度はアクセスの１つの目安として用いられるだけであると考える。これらをより正確にする方法としては、よく用いる値についての選択度、結合度についての統計情報をディレクトリ（即ち異種性情報）内に蓄積していくものと考えられる。異種性情報（ＨＩ）の格納オーバヘッド、管理等とからめて、これらの統計情報の扱いについて今後検討していきたい。

5.7.3　ＤＭＬの生成について

我々が生成したＤＭＬプログラムはＣＯＢＯＬ　ＤＭＬプログラムの手続部（ｐｒｏｃｅｄｕｒｅ　ｄｉｖｉｓｉｏｎ）の生成である。ＤＭＬプログラムはＤＢＴＧモデルの局所内部スキーマ（ＬＩＳ）上のアクセスシーケンスを表わしている。我々の目標は、第1にリレーショナル計算に基づいた非手続的なＬＣＳ問合せ（ＬＱ）を、ＤＢＴＧ　ＤＭＬの シーケンスによって手続的に表現することであった。この点が問合せ変換（ＱＴ）における最も重要な点と考えている。この意味で、我々はこの目標を達成出来たと考えている。次の問題は、実際のＤＢＴＧデータベースシステム上でＣＯＢＯＬ　ＤＭＬプログラムを実動させることである。完全なＣＯＢＯＬ　ＤＭＬプログラムは図5.62〔ＣＡＲＤＡ７９〕のような構成を持っている。我々のＤＭＬプログラムを実動可能とするためにはまず、データ部（ｄａｔａ　ｄｉｖｉｓｉｏｎ）の定義を付加する必要がある。データ部ではこのｒｕｎ－ｕｎｉｔ内でローカルに使用する変数を定義する。ローカルな変数としては、次の様なものがある。

1）　ＤＭＬブロック内の変数％10 と％11に対応する変数の宣言。及び％10 変数の初期値としてｆａｌｓｅ をセットする。

2）　全ＤＭＬブロックに共通な変数ＬＦＯＵＮＤの宣言。

3）　ＤＭＬブロック内の％20 に対応する変数をＵＳＡＧＥ　ＩＳ　ＤＢ－ＫＥＹ節を付加しての宣言。

4）　結果リレーション及び結果属性に対応した各々レコード型とデータ項目の定義。

1）と3）とはＤＭＬプログラムを生成時に、アクセス木をたどりながら対応する変数の定義文を生成する。合流ノードに対しては％20 変数を生成する。4）も同じくアクセス木をたどっている時、結果属性がＡＴノードにあればそれを結果データ項目として定義する。結果リレーション名は、ＬＳＣ問合せと木構造表現した時に用いるＲＲＴＢＬ〔8.2〕を用いる。

5.5.2で述べた様に、アクセス木が木構造をしているために変数％10 、％11及びLFOUNDを用いて複雑な成功及び失敗復帰の制御を行なっている。この制御は木構造を持ったアクセス木に対しては必要である。しかし、線型なアクセス木に対しては不要であり、多くの問合せは線型であ

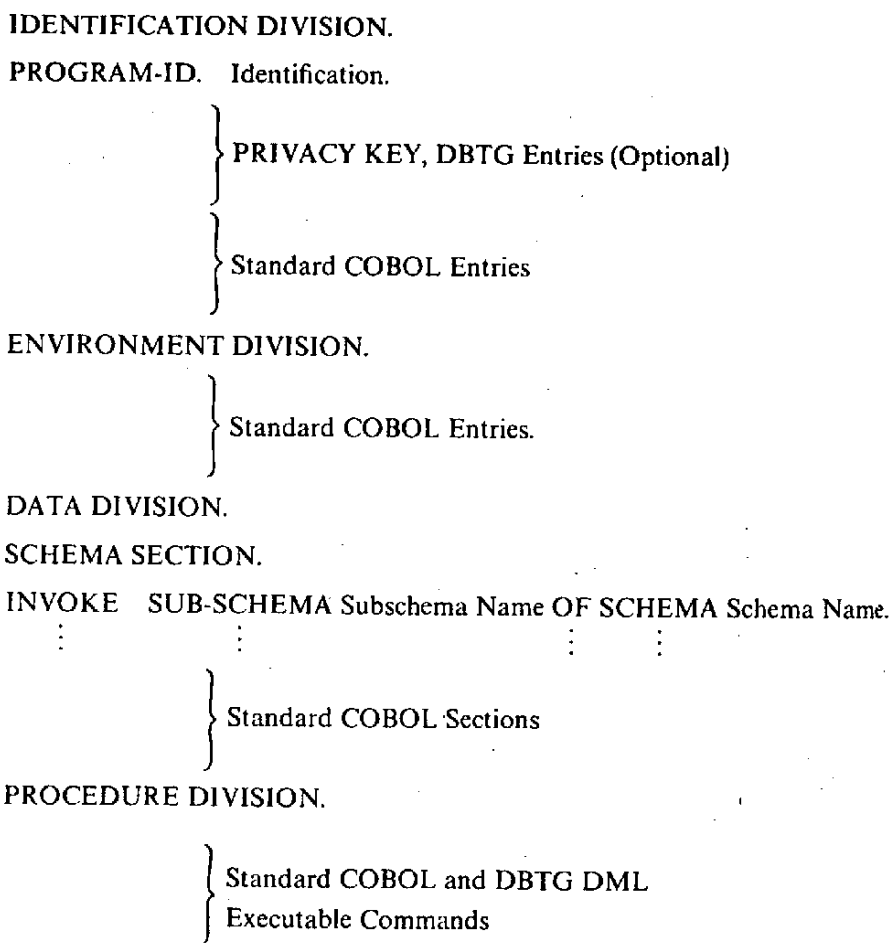


図5.62　COBOL　DBTG　DMLプログラムの構成

ろうと考えられる。このため、線型なアクセス木用のＤＭＬブロックの定義が今後とも必要になると考える。この線型アクセス木用ＤＭＬブロックは、％１０、％１１、ＬＦＯＵＮＤに関する文を除くことによって容易に得られる。これは図5.34に示したＤＭＬブロックごとに各々定義できる。

次に生成されたＤＭＬプログラムの最適化がある。図5.64の例を考えてみよう。我々のＤＭＬブロック生成法では図のｂ）のようなプログラムが生成される。ここでレコード型ＡのオカーランスをＧＥＴする以前に、このオカーランスがセット型Ｓを介してレコード型Ｂ内にリンクされたオカーランスをもっているかどうかチェックする方が有効である。このようなＤＭＬブロック間での文の組みかえによる最適化も考えられる。

現在の各ＤＭＬブロックは必ずＧＥＴ＄０１，を生成している。毎回ＧＥＴすることはパフォーマンスが損なわれる。結果属性も制限式もないものに、ＧＥＴ文は不要である。よって＄０３、＄０４、＄０５がＮＩＬの場合はＧＥＴ文を出力しないようにする。

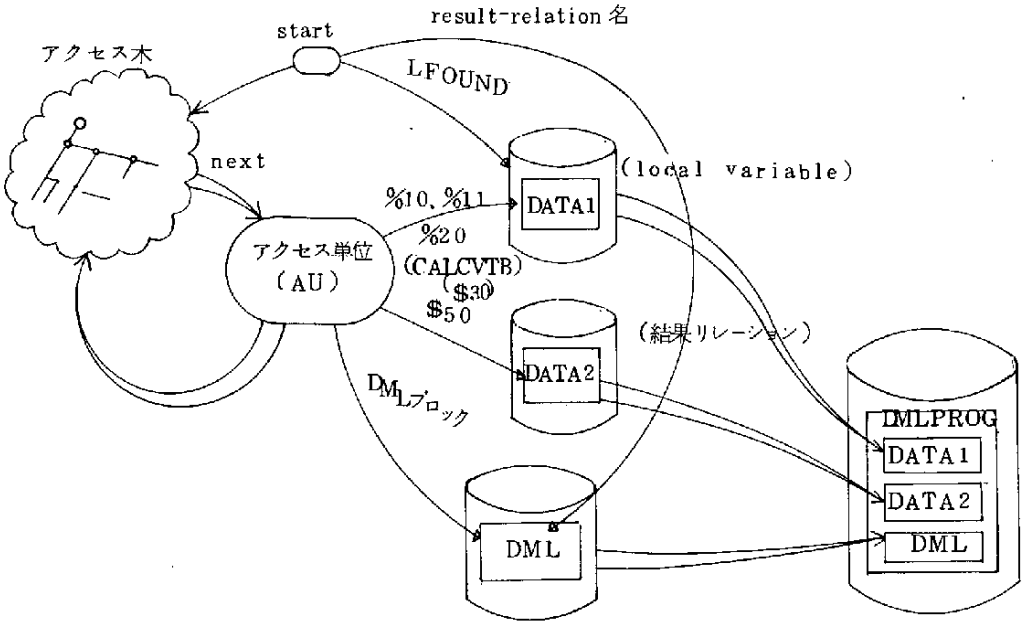
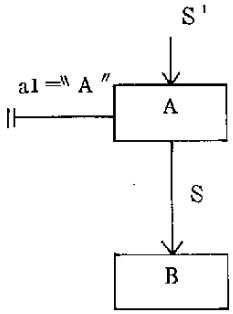


図5.6.3　COBOL DML プログラムの作成



a) アクセス木

```
L1. FIND NEXT A WITHIN S'
    GET A.
    if a1 = " A ", go to L1.
    if S is empty, go to L1.
    FIND NEXT B WITHIN S.
```

b) DML プログラム

```
L1. FIND NEXT A WITHIN S.'
    if S is empty, go to L1.
    GET A.
    if a1 = " A ", go to L1.
    FIND NEXT B WITHIN S.
```

c) 最適化DMLプログラム

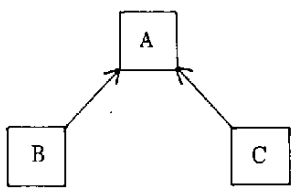
図5.6.4　DMLプログラムの最適化

DMLブロック内には、RESULTとGNVという2つのサブルーチィンがある。RESULTはそのレコード型の結果属性の値を結果リレーションに出力するためのサブルーチィンである。GNVはCALCアクセス単位においてCALC等価制限式が、CALC等価制限述語の論理和である時、今回順番に1つづつdisjunctの値をとり出すためのサブルーチィンである。まずGNVサブルーチィンについて考えてみよう。CALC等価制限式a.a1="v1" and a.a1="v2" and …and a.a1="vn"について考えよう。アクセス単位がCALCタイプである時、CALCVTBLと呼ばれるテーブルの先頭から順に値v1、v2、…、vn を格納する。GNVは、これが呼び出(call)されるたびにCALCVTBLの先頭から値を1つづつ取り出し、valパラメータに返す。このためCALC DMLブロック内のcall GNV((＄03), val, mode).は、DATA1ファイル〔図5.6.3〕にCALCVTBLを生成する(同時にテーブル内の現在指示変数CCALCVTBLの値を0とする)とともに、call GNV(val, mode)をDMLファイルに出力する。この様にGNVルーチィンは比較的容易にCOBOLを用いてつくれる。

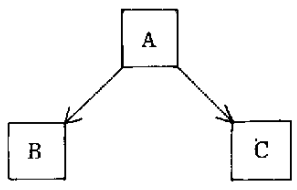
次にRESULTサブルーチィンについて示す。RESULTルーチィンの呼び出し形式は次の様である。

RESULT(ratt-no, ratt-value)

即ち、これはある結果属性(ratt-noによって示される)の値を結果リレーションの所定の場所に格納する。原則的にRESULTルーチィンは、呼び出されるたび、結果リレーションの現在の組(tuple) 内の結果属性に値を格納していくものである。これは図5.6.5 a)のようなアクセス木において正しい。即ち、部分木の根ノードAとこの2つの子ノードとは互いに1：1の関係性がある。即ちAの1つのオカーランスa1はBのただ1つのオカーランスb1とCのただ1つのオカーランスc1と関係しているだけである。これを表で表わしたものを図5.6.5のc)に示す。しかし、図5.6.5のb)の場合を考えてみよう。これはAとBとは1：



a) 1：1



b) 1：n

図5.6.5 アクセス木と結果リレーション

R

A	B	C
a1	b1	c1
a2	b2	c2

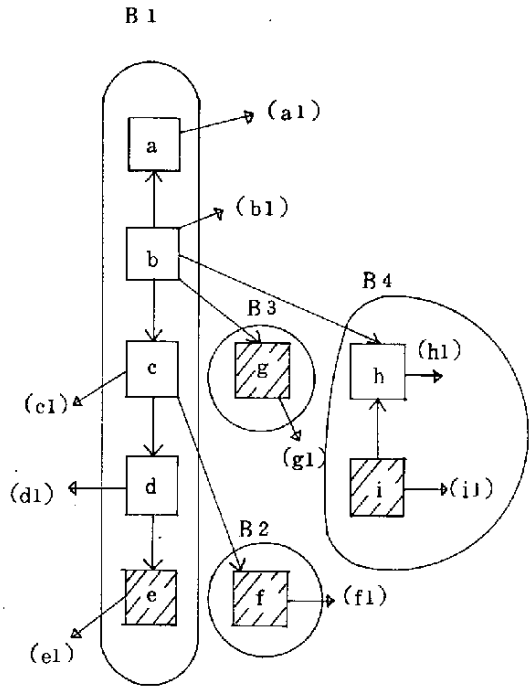
c) a)の結果リレーション

A	B	C
a1	b1	c1
a1	b2	c1
a1	b1	c2
a1	b2	c2
a2	b3	c3
a2	b4	c3
a2	b3	c4
a2	b4	c4

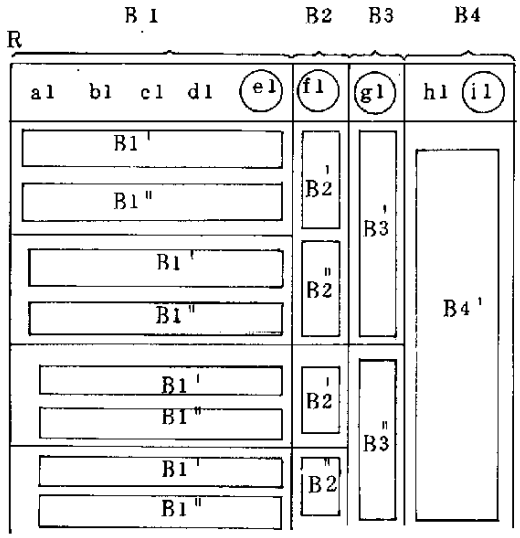
d) b)の結果リレーション

n、AとCとも1：nの関係性があり、しかしBとCは独立である場合である。これは関係モデルにおける多値従属、即ちA→→B｜Cに対応している。b）を表で表わしたものをd）に示す。b）のようなアクセス木からd）のような結果リレーションを得るためには、AとBとの結果リレーションとAとCとの結果リレーションとをAについて結合を行なうことによって得られる。しかし、このような結合演算を行なうために中間結果を図5.6.5のb）の様なパターンの各部分木ごとに作ることは、5.4.5で述べた縦型アクセス木生成アルゴリズム（DFA）の長所を失なわせてしまう。このため我々は、1つのアクセスパス内で必要な結果を得るたびごとに逐次結果リレーションへ格納しながら結合演算を行なっていく方法を考えた。この方法について以下に述べる。

まず最右端ノード（most-right node or MRN）を次の様に定義する。MRNはアクセス木において、右手に兄弟ノードを持たず、かつ下方に子ノードを持たないもの又は、NEXTノードで子ノードを持たないものである。図5.6.6の例はe,f,g,iがMRNである。アクセス木から得られるアクセスシーケンスにおいて、あるMRN ATノードから次のMRN ATノードまでのATノードのシーケンスをブロックと呼ぶ。このブロックを左から順に番号づけ、i番目のブロックをBiと記す。図5.6.6では、（a、b、c、d、e）（f）（g）（h,i）は各々B1、B2、B3、B4 に属する。nをアクセスシーケンス内の全ブロック数とする。結果リレーションをRとし、R内の結果属性の並びは、アクセスシーケンスの並びと同一とする。mをRの次数（degree）とし、rati （i=1、2、…、m）をR内のi番目の結果属性とする。まず、最初のブロック（B1）について考える。aから出発し、b、c、d、eとアクセスし、eでのアクセスが終了するまでアクセスするとB1'の組の集合が得られる。ついでeが終了するとdはNEXTノードなのでd1 の異なった値に対して、組集合B2'が得られる。ここでa、b、c、d、eのアクセスが全て終了したとしよう。次にfのアクセスを行なう。この時f1の1つの値はB1" とB2"とに対応する。よって、fが2つの値B2' とB2"を持つとB2" に対してB1' とB2'がコピーされる。これはB1 の結果とB2 の結果をノードc（属性c1）について結合を行なったことと等しい。この様にしてB3、B4 についても同様にして結果リレーションがつくられる。



a） アクセス木



b） 結果リレーション

図5.6.6

RESULTサブルーチィンの概要を図5.6.7に示す。ここでcstは現在のブロック番号、ratt-noは結果属性番号である。BNO(ratt-no)は、この結果属性の属するブロック番号である。BNO(m+1)=n+1である。H(cst)とT(cst)、cst番目のブロック(Bcst)の最初の1連の組集合の各々先頭と最後の組番号を示している。TH(cst)とTT(cst)とはBcstの現在のある値が格納される各々先頭及び最後の組番号を示している。即ち、TT(cst)-TH(cst)=T(cst-1)-H(cst-1)である。

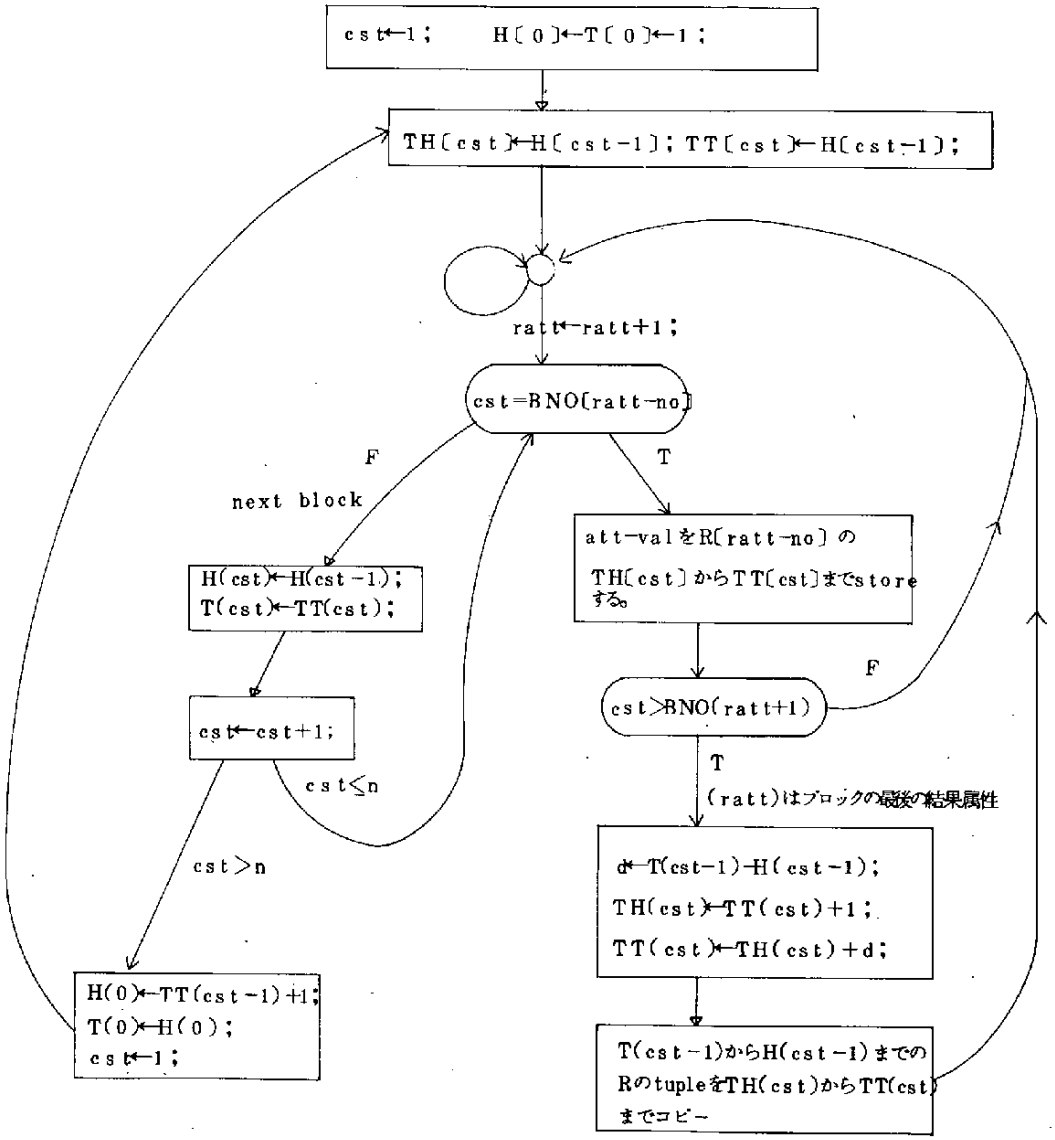


図5.67　　RESULT サブルーチィンの概要

5. 8　まとめと問題点

本章では同種化の逆過程としての問合せ変換（ＱＴ）問題を論じた。全体概念スキーマ（ＧＣＳ）に基づいたＧＣＳ問合せ（ＧＱ）は、第７章で論じる問合せ分割（ＱＤ）によってまず各サイト単位のＬＣＳ問合せ（ＬＱ）に分割される。ＬＣＳ問合せは、ＧＣＳ問合せと同一の問合せ言語ＱＵＥＬによって記述されている。各サイトはそのサイトのデータベースシステムによって実行可能なアクセス言語（これをＬＩＳ　ＤＭＬと呼ぶ）は、一般にＱＵＥＬとは異なる。このため問合せ変換は、そのサイトの局所概念スキーマ（ＬＣＳ）に基づき、かつＱＵＥＬで記述されたＬＣＳ問合せを、そのサイトの局所内部スキーマ（ＬＩＳ）に基づきかつＬＩＳ　ＤＭＬによって記述されたプログラムへの変換を行なう。本章では特に、ＬＩＳとしてＤＢＴＧスキーマを,ＬＩＳ　ＤＭＬとしてＤＢＴＧ　ＤＭＬ（ＣＯＢＯＬ）を設定した場合について論じた。ＱＵＥＬ　ＬＱは非手続的であるが、ＤＢＴＧＬ　ＤＭＬプログラムは手続的である。

よって問合せ変換では、次の２つのことをやらねばならない。

１）局所概念スキーマ（ＬＣＳ）要素を局所内部スキーマ（ＬＩＳ）要素への対応づけ。

２）非手続的ＬＣＳ問合せから手続の生成。

前者は構造変換（ＳＴ）と呼び、後者を最適化（ＯＰＴ）と呼ぶ。ＳＴでは、異種性情報（ＨＩ）を基にして、ＬＣＳ要素（リレーション属性）は、ＬＩＳ要素（レコード型、セット型、データ項目名）へ対応づけられる。ＬＩＳ要素のなかでＬＩＳ上には存在するが、対応するＬＣＳ要素かＬＱと／又はＬＣＳ上には現われない要素を明らかにし、これを隠れ構造（ＨＳ）と名づけた。構造変換ではＬＣＳ問合せ（ＬＱ）内に現われるＬＣＳ要素を対応するＬＩＳ要素へ変換するとともに、今述べた隠れ構造も同時に明らかにされねばならない。これは異種性情報（ＨＩ）を用いて行なわれる。この構造変換は、グラフ表現の変形を通して行なわれる。

構造変換（ＳＴ）によって生成されたグラフ（ＤＢＴＧ問合せグラフ又はＤＱＧ）は、ＬＣＳ問合せ（ＬＱ）の意味をＤＢＴＧモデル要素であるレコード型、セット型、データ項目を用いて非手続的に表わしたものである。この非手続的表現からアクセスパスをみつけるのは最適化（ＯＰＴ）の役目である。ＤＱＧでは、レコード型をノードとして、セット型をアークとして表わしている。アクセスパスはこのＤＱＧから、ノードをＤＱＧノード、枝をＤＱＧアークとするアクセス木(ＡＴ)と呼ばれる木を生成することによって得られる。グラフは一般にループをふくんでいるので、アクセス木（ＡＴ）内には、１つのＤＱＧノードに対応して１つ以上のＡＴノードが存在し、１つのＤＱＧアークにはユニークなＡＴ枝（ｂｒａｎｃｈ）が存在する。１つのＤＱＧノードに対して、冗長なＡＴノードが存在する時、これらのＡＴノードをこのＤＱＧノードに対する合流ノード（ｃｏｎ－ｆｌｕｅｎｔ　ｎｏｄｅ　ｏｒ　ＣＮ）と呼ぶ。合流ノードの存在は次のことを意味している。

１）１つのアクセスパス内で合流ノードに対応するＤＱＧノード、（即ちレコード型）が各合流ノードの上方の枝に対応するＤＱＧアーク、（即ちセット型）を介して冗長にアクセスされる。

２）この１つのＤＱＧノードに対する冗長アクセスにおいて、各アクセスにおいて生成される

結果オカーランスの各集合の共通部分（intersection）が、このアクセスパスにおいてもとめられるべき結果となる。

このような共通部分をもとめるためには、一般に各合流ノードごとに中間結果を生成するとともに集合演算（i. e. intersection）の実行とが必要となる。各ノードごとに中間結果を生成するためには、各々に対応してファイルをつくらねばならない〔5.4.4のBFAを参照〕。このことは、中間結果管理上及び実行効率上大きな問題となる。このため、中間結果の数を最少とすることは重要な目標となる。我々のアクセス木生成アルゴリズム（DFA）は、この目標達成を目指している。この方式によるアクセス木では、あるDQGノードに対応する合流ノード（ATノード）は、必ず最初に現われた合流ノードを根ノードとする部分木内に全てふくまれる。あるアクセスパスについてこの部分木内の合流ノードのオカーランスは、この部分木の根ノードとしての合流ノードのオカーランスと等しくなければならない。よってDFAによるアクセス木では、このことは部分木内の合流ノードをアクセスする時、根ノードのオカーランスと比較すればよいことになる。これは根ノードの合流ノードをアクセスする時に1つの結果リレーション（ファイル）に、これのキー属性値を出力しておけばよい。よって我々のDFAではただ1つの中間結果リレーションを必要とするだけであり、このため共通部分を求めるための何等の集合演算も不要である。これは主要な長所である。

DFAによるアクセス木（AT）生成において、あるDQGノードがまだAT内に組みこまれていない複数のDQGアークを持つ時、アクセスされるオカーランス数の期待値がより小さいものを次のAT枝として選んでいる。これはヒューリスティックス（heuristics）である。しかし、あるオカーランスからこの選ばれた枝を介してリンクされているオカーランスからなる部分オカーランス木は必要な条件を全て満足するとは限らない。よって部分オカーランス木自体がある選択度をもつことになる。我々のヒューリスティクスは、より沢山のものをアクセスすることによって制限されるよりも、より小さいものをアクセスすることによって制限される方が有効であるという考えに立っている。よって、次の枝の選択としてこの枝にリンクされたDQGノード内のアクセスされるオカーランス数のより小さいものをまずアクセスすることは有効と考えられる。しかし、これは必ずしも最適解を導くとは限らない。

最適解をもとめるためには、5.4.2のCで論じた部分木の選択度と結合度を用いて生成されたアクセス木内のアクセスされるオカーランス数の期待値を求め、これを極少とするようにアクセス木を再構成することも出来る。ただし、アクセス木の再構成は先に述べた合流ノードに関する条件、即ち、あるDQGに対する合流ノードは、最初の合流ノードを根とする部分木内にあるという条件を満足していなければならない。この木の再構成アルゴリズムは今後検討していくことにしている。

我々のQTは実際の物理的なページアクセスを考慮していない。しかしページアクセスはDBMSの内部スキーマレベルの問題であり、そのインプリメンテーションに依存している。我々はむしろ、中間結果の数をなるべく少なくし、この中間結果の操作のための演算を不要とさせることが、スキーマレベルでは最重要であると考えている。

ＤＢＴＧ以外のモデル、例えばＩＭＳモデルについても我々のＱＴの拡張は容易であると考えている。既存データモデルは事象集合間の関係性集合、例えばＤＢＴＧのセット型、ＩＭＳの階層パスによって特徴づけられる。さらにこの関係性集合は、アクセスの単位でもある。この様な関係性集合情報をＨＩに持たせることによって、他のモデルへの拡張は容易であると考える。

6. 統合化と分散情報

第3章で述べたように分散型データベースシステム（ＤＤＢＳ）設計において、解決されるべき問題点としては、

1) その構成要素たるデータベースシステム（ＤＢＳ）の異種性の解決と

2) 地理的に分散し、かつ論理的に異なった意味を持つデータベースシステム（ＤＢＳ）を1つの論理的（仮想的）データベースシステムに統合することの2点である。

前者は異種性問題、後者は分散問題と呼ばれる。第4章と5章において、既に異種性問題については議論した。本章では、各データベースシステム（ＤＢＳ）がそのデータモデルと言語という構文的側面が同種化されているという前提にたって、第2番目の分散問題について議論することにする。

同種化された各ＤＢＳは、即ち局所概念スキーマ（ＬＣＳ）レベルでは共通のデータモデル（i.e. Ｅ－Ｒモデル）と共通のアクセス言語（i.e. ＱＵＥＬ）とを持っている。問題は、各データベースシステムが共通の構文構造を持ちながら、異なった意味構造を持っていることである。このようなデータベースシステムから、分散型データベースシステム（DDBS）全体にある1つの意味構造を持った論理的ＤＢＳを設定することを統合化（integration）と呼ぶ。統合化とは、ネットワーク共同体（network community or NC）の要求を満たすようなデータベースの意味の記述を各データベースシステムの局所内部スキーマ（ＬＩＳ）を基にしてつくることである。このようなデータの意味の記述を全体概念スキーマ（ＧＣＳ）と呼ぶ。局所内部スキーマ（ＬＩＳ）から全体概念スキーマ（ＧＣＳ）への統合は、全体管理者（ＧＡ）が行う。ＧＡは、ＧＣＳ定義用言語ＧＳＤＬ（global conceptual schema description language）によってＬＩＳとＧＣＳとの対応を記述する。この記述されたものを分散記述（distribution description or DD）と呼ぶ。ＬＩＳとＧＣＳとの対応情報は、分散情報（disribution information）となり、ネットワークデータディレクトリ（ＮＤＤ）として各サイトで管理される。

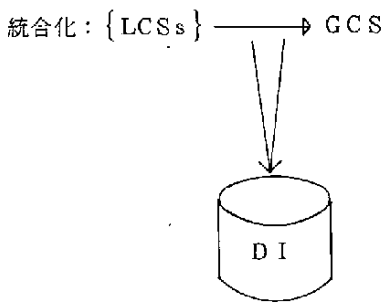


図6.1　統合化の概要

複数の局所概念スキーマ（ＬＣＳ）が1つへ統合されるためには、これらのＬＣＳ間に互いに意味的関連がなければならない。6.1では、このＬＣＳ間の意味的関係性について論じる。次に、この関係性をもとにして全体管理者（ＧＡ）が、ＧＳＤＬを用いて分散記述（ＤＤ）を記述する。6.2ではＤＤについて論じる。6.3では分散記述（ＤＤ）から生成される分散情報（ＤＩ）について論じ、そのスキーマを示す。

6.1　セマンティックリンク（ＳＬ）

分散した局所概念スキーマ（ＬＣＳ）が１つの全体概念スキーマ（ＧＣＳ）へ統合されるためには、これらのＬＣＳは互いにある意味論的関係性をもたねばならない。我々は、この関係性を表わすために、セマンティックリンク（ＳＬ）の概念を導入する。セマンティックリンク（ＳＬ）は、２つのＬＣＳリレーション間に存在し、２つのリレーション内の各々の部分集合間の関係性を表わしている。即ち、これらの部分集合内の対応する２つの属性の領域が同一であるとともに、更に２つの部分集合の集合論的関係性も示している。例えば、２つのＬＣＳリレーションＡ（a_1、a_2、a_3）とＢ（b_1、b_2、b_3、b_4）とを考えてみよう。ＬＣＳ属性a_1とb_2、a_2とb_1とは互いに同じ領域に属し、かつＬＣＳリレーションＡとＢの各々の射影Ａ〔a_1、a_2〕とＢ〔b_2、b_1〕とが同一であるとする。すると図6.2のようなセマンティックリンク（ＳＬ）が、ＬＣＳリレーションＡとＢとの間に存在する。

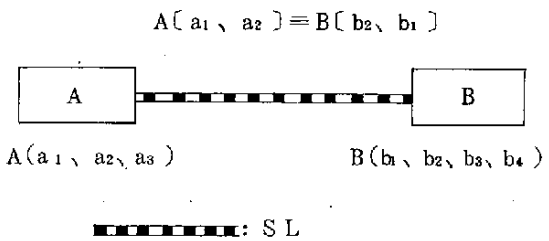


図6.2　ＬＣＳリレーションＡとＢとのＳＬ

集合論的関係性は、図6.3にまとめてある。図中で、Ｒ１とＳ１とは、各々ＬＣＳリレーションＲとＳとの部分集合とする。さらに、Ｒ１とＳ１とは同一のスキームをもっている。Ｒ１≡Ｓ１とは、２つの集合が同一であることを示している。Ｒ１⊃Ｓ１は、Ｓ１がＲ１の部分集合であることを示している。即ち、Ｓ１の全ての組に対応する組をＲ１は持っていることを表わしている。Ｒ１∩Ｓ１は、Ｒ１とＳ１とは共通部分を持つが、互いに非共通でない部分も持っていることを示している。Ｒ１∪Ｓ１は、Ｒ１とＳ１とは全く共通部分を持たないことを示している。

関係性	説明
Ｒ１≡Ｓ１	Ｒ１＝Ｓ１
Ｒ１⊃Ｓ１	Ｒ１∩Ｓ１＝Ｓ１、Ｒ１－Ｓ１≠ϕ　Ｓ１－Ｒ１＝ϕ
Ｒ１∩Ｓ１	Ｒ１∩Ｓ１≠ϕ　Ｒ１－Ｓ１≠ϕ　Ｓ１－Ｒ１≠ϕ
Ｒ１∪Ｓ１	Ｒ１∩Ｓ１＝ϕ

図6.3　ＳＬにおける集合論的関係性

例として、サイト４にあるプロジェクトデータベースシステムＰＲＤＢＳ〔4.6.2を参照〕のＬＣＳリレーションＰＲＯＪＥＣＴと、サイト１及び２にあるプロジェクト管理データベースシステム

PMDBS〔4.6.1を参照〕のLCSリレーションPROJECTについて考えてみよう。これらのスキームは、次のようである。

PRDBS at site 4　PROJECT(proj－no, projname, projeyar, projsyar, projstat, loc)

PMDBS at site 1 and 2　PROJECT (pno, pname, leader, syar, eyar, budget)

この3つのLCSリレーション間のセマンティックリンク（SL）を図6.4に示す。サイト1と2とにある2つのPROJECTリレーションは同一のスキームを持つが、共有部分を持たない。PMDBSのPROJECTリレーションの射影PROJECT(pno, pname, syar, eyar)は、PRDBSのPROJECTリレーションの射影PROJECT(proj－no, projname, projsyar, projeyar)の部分集合となっている。3つのリレーションのオカーランスレベルの関係性を図6.5に示す。

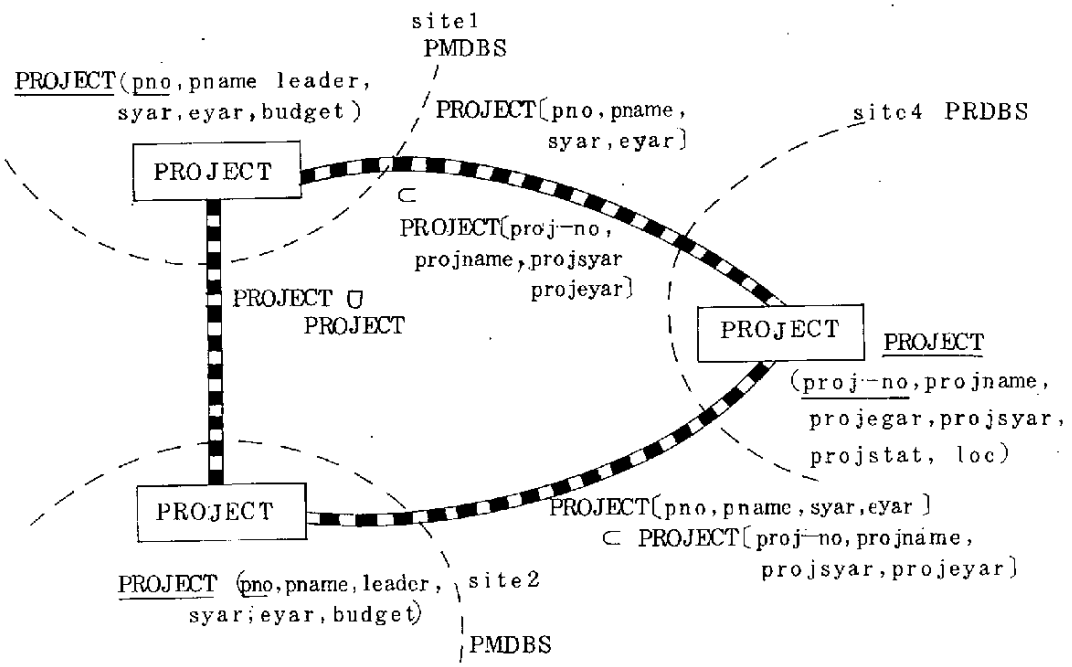


図6.4　3つのLCSリレーション間のSL

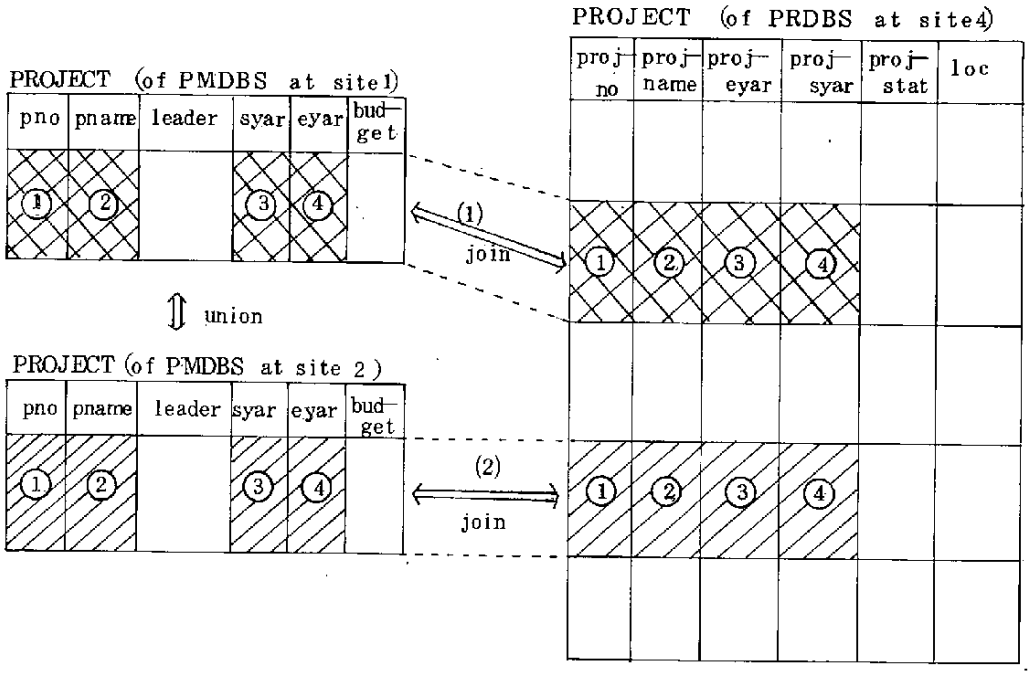


図 6.5　3つのリレーションの関係

6.2　分散記述（DD）

次に、6.1で述べたセマンティックリンク（SL）を用いて全体概念スキーマ（GCS）リレーションを局所概念スキーマ（LCS）リレーションから定義することについて考えよう。GCSリレーションは、全体概念スキーマ（GCS）定義用言語（GSDLと呼ぶ）を用いて定義され、この定義の集合を分散記述（distribution description or DD）と呼ぶ。

全体管理者（GA）は、全体概念スキーマ（GCS）の意味論的側面をセマンティックリンクの助けをかりて明らかにし、その構文的側面をGSDLによって記述する。GCS及びLCSの構文的側面は両スキーマのリレーショナル記述（RD）内のリレーションによって表わされるので、GCSのLCSからの定義は、リレーショナルモデルにおける視野（view）の定義と類似している。リレーショナルモデルの設計において、1つのユニバーサルリレーションの存在が仮定されており、これを属性間の従属（dependency）関係を基にして、射影演算（projection）を用いて、正規形へ（垂直的に）分割されていく。従がって、この様に垂直的に分割された複数のリレーション（基本リレーション（base relation））上の視野は、これらの結合演算によって定義される。しかし、統合型DDBSにおいて、各サイトのLCSリレーションは既に存在していたものであり、1つのユニバーサルリレーションを射影演算によって垂直的に分割されたものではない。このため、LCSリレーションから

ＧＣＳリレーションを定義するためには、視野定義における結合演算に加えて、和演算（union）機能が必要となる。

ＱＵＥＬのようなリレーショナル計算言語は、垂直分割された正規形リレーション上に必要な視野を完全にかつ非手続的に定義できる。しかし、リレーションの和（union）を取ることは出来ない。このため我々は、リレーションの和もとれるようにＱＵＥＬの拡張を行った。このＱＵＥＬを拡張した全体概念スキーマ（ＧＣＳ）定義言語をＧＳＤＬと呼ぶ。

ＧＳＤＬが必要とする機能としては、次のものをあげることができる。

1）ＧＣＳリレーション名とそのタイプ（ＥＳＲかＲＳＲか）の定義

2）ＧＣＳ属性、その領域、及びそれがとる値の長さと型の定義

3）ＧＣＳ属性と、ＬＣＳ属性との対応関係の定義

4）ＧＣＳリレーションと、ＬＣＳリレーションとの対応関係の定義

（結合、射影、制限、及び和演算がリレーショナル演算として必要）

次にＧＳＤＬについて述べる。ＧＳＤＬの形式的定義は付記１に示してある。ＧＳＤＬは、drange文、define文、drop文との３つの文から成っている。drange文は、ＧＣＳリレーションを定義するために参照されるＬＣＳリレーションの組変数を定義するために用いられる。これは次の様な形式をもっている。

drange （l_1、l_2、…、lm)(L_1：s_1、L_2：s_2、…、Lm：sm）；

これは各li（i＝1、2、…、m）は、サイトsiにあるＬＣＳリレーションLiに対する組変数であることを定義している。ここで各ＬＣＳリレーションLiは互いに全て異なる必要はない。各サイト番号Siについても同様である。例えば、サイト１にＬＣＳリレーションＥＭＰ、ＤＥＰＴがあり、サイト２にＬＣＳリレーションＳＡＬがあるとしよう。ここで、ＥＭＰに対して、組変数ｅ１とｅ２を、ＤＥＰＴにｄを、ＳＡＬにｓを定義しようとすると次のようになる。

drange （e1、e2、d、s）（EMP：1、EMP：1、DEPT：1、SAL：2）；

define文は、ＬＣＳリレーションから、新たにＧＣＳリレーションを定義する。これは次のような形式を持っている。

define <type> <gcs−rel−name> (<gcs−att−list>)

<sub−def>{:<sub−def>}；

<gcs − rel − name>は、定義されるＧＣＳリレーション名である。<type>は、これがＥＳＲかＲＳＲかを示している。<gcs − att − list> は、このＧＣＳリレーションのスキームを示している。

<gcs − att − list> :: =<gcs−att−def >{ , <gcs − att − def>}

<gcs − att − def > :: =<gcs−att−name>〔/<domain−name>〕〔:<format>〕

各ＧＣＳ属性に対して、その名前とともに、その領域名（<domain−name>）、データタイプと長さ（<format>）も定義される。ここで定義される領域名は仮想的なものであり、他のＧＣＳリレーションとの関係性を示すために用いられる。

<sub−def>の並びは、ＬＣＳリレーションとＧＣＳリレーションとの対応を示している。<sub−def>は、あるＧＣＳリレーションの副定義と呼ばれ、次のような形式をもっている。

<sub−def> :: =（<target−list>）where <qual>

<target−list>と<qual>はＱＵＥＬにおけるものと同じである。<qual>はＬＣＳリレーションについての条件を結合論理式と制限論理式との述語論理式で表わしている。目標リスト（<target−list>）は、<qual>で得られるリレーションの射影を行うとともに、結果属性の名前づけを行う。この様に１つの<sub−def>は、ＬＣＳリレーションの結合を表わしている。

<target−list>は、次の様な形式を持つ。

<target−list> :: =<t−clause> {, <t−clause>}
<t−clause> :: =<gcs−att−name>=<exp> | <lv−att>
<exp> :: =<a−exp> | ⓐ
<lv−att> :: =<lvar> . <att−name>

<a−exp>は、ＬＣＳ属性上に定義される算術式である。<exp>の値がⓐである時は、対応するＧＣＳ属性値が未定義値（null value）をとることを表わしている。統合型ＤＤＢＳにおいては、定義されたＧＣＳリレーションの一部に対応するデータがどのサイトにも存在しないかもしれない。図6.6は、３つのリレーションA、B、Cの間のセマンティックリンクとこれからつくられるＧＣＳリレーションG（a_1、a_2、b_2）の例を示している。A、a_1値が450から500まで（450<A.a_1<500）とるAの組に対応する組は、ＬＣＳリレーションBとCの中にはないので、ＧＣＳリレーションGのb_2は、a_1のこの値に対して未定義値ⓐ値をとることになる。ＧＣＳリレーションは、ＧＳＤＬを用いて次の様に定義される。

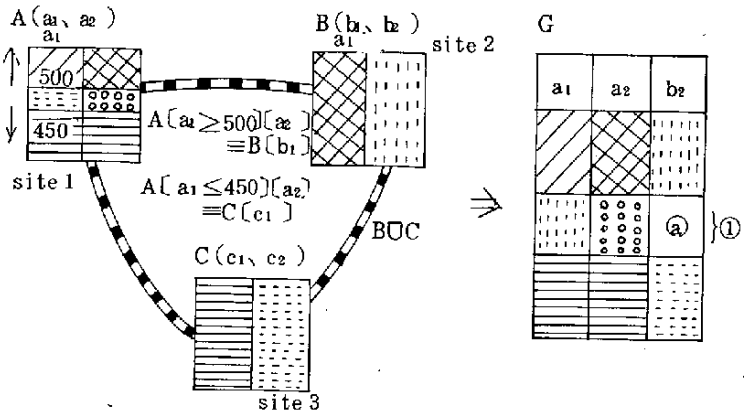


図6.6　ⓐの例

```
drange ( a、 b、 c )  ( A : 1、 B : 2、 C : 3 ) ;

define ESR  G ( a1、 a2、 b2 )
    ( a . a1 、  a . a2 、  b . b2 )
    where  a . a1 ≧ 500  and  a . a2 = b . b1 :

    ( a . a1 、  a . a2 、  b2 = c . c2 )
    where  a . a1 ≦ 450  and  a . a2 = c . c1 :

①{  ( a . a1 、  a . a2 、 b2 = ⓐ )
    where  a . a1 、< 500  and  a . a1 > 450 ;
```

①の部分は、図6.6内のＧＣＳリレーションG内の①の部分を定義している。

前述したように、ＧＣＳリレーション定義では、ＬＣＳリレーションの和演算機能も必要となる。このため<sub--def>の並びは、この中の各々の<sub−def> が生成する結果リレーションの和をとることを表わしている。このために、各<sub−def>内の<target−list>は、この結果リレーションが<gcs−att−list>において定義されたものと同一のスキームを持つように定義している。

最後のdrop文は、以前にdefine文で定義されたＧＣＳリレーションをＧＣＳから除くために用いられる。これは次の様な形式を持つ。

```
drop    < gcs − rel − name > ;
```

この他に、ＧＣＳリレーションのスキームの変更、例えば、ＧＣＳ属性の追加、削除のためのコマンドも必要である。

全体概念スキーマ（ＧＣＳ）は、次の２つのステップによって定義される。

1) ＧＣＳ事象集合リレーション（ＥＳＲ）の定義

2) ＧＣＳ、ＥＳＲからＧＣＳ関係性集合リレーション（ＲＳＲ）の定義

全体概念スキーマ（ＧＣＳ）レベルの事象集合リレーション（ＥＳＲ）は、各サイトに存在するＬＣＳリレーション（ＥＳＲとＲＳＲ）から、ＧＣＳレベルで意味のある事象（entity）に対応してdefine文によって定義される。ＧＣＳ関係性集合リレーションは、上述した様に定義された事象集合リレーション間の関係性を定義するものである。２つの事象集合間の関係性とは、一方の事象、即ち、組と他の事象（組）の集合との間の関連を示している。このため、関係性集合リレーションを定義する方法としては、次の２つがある。

1) ある事象集合リレーション内の各組と、これと関連がある他の事象集合リレーション内の組との対をＧＣＳ関係性集合リレーションの組として、外延として定義する。

2) ＧＣＳ関係性集合リレーションを各サイトのＬＣＳ関係性集合リレーションの視野（view）として定義する。

1)は、全体概念スキーマ（ＧＣＳ）上に、各サイトの局所概念スキーマ上には存在しない関係性を新

たに定義することに対応している。これは、新たに外延を定義することであり、視野の定義、即ち、内包の定義ではない。

既存のＬＣＳ関係性集合リレーション（ＲＳＲ）は、２つのＬＣＳ事象集合リレーション（ESR）内の関連のある組の識別子の値の対をその組として持っている。よって、あるＧＣＳ事象集合リレーションを構成するＬＣＳリレーションと他のＧＣＳ事象集合リレーションのＬＣＳリレーションとが、ＬＣＳ関係性集合リレーションによって関連づけられているならば、ある関係性をこの２つのＧＣＳリレーション間にも定義できることになる。しかし、この様にして定義されたＧＣＳリレーションを事象集合とみるか関係性集合とみるかは、全体概念スキーマ（ＧＣＳ）設計者、即ち全体管理者の視方に依存している。我々の全体概念スキーマ定義用言語ＧＳＤＬは、構文的なリレーションスキームを定義するものである。ＧＣＳリレーションを事象集合と視るならば、define文内の<type>をＥＳＲとし、関係性集合と視ればＲＳＲとすればよい。関係性集合リレーションが関連づける事象集合リレーションは、属性とこれの属する領域によって定められる。これは、局所概念スキーマにおける事象集合リレーションと関係性集合リレーションとの関係と同じである。

ＧＣＳリレーションの定義を例について考えてみよう。例として、図6.4と6.5の３つのＬＣＳリレーションからＧＣＳ事象集合リレーション（ＥＳＲ）ＰＲＯＪＥＣＴ（pno、pname、manager、budget、loc）の定義について考える。サイト１と２にある２つのＰＲＯＪＥＣＴリレーションは、互いに同一のスキーム（union－compatible）を持っており、共通部分（組集合）を持っていない。これはセマンティックリンクＳＬ：ＰＲＯＪＥＣＴ ∪ ＰＲＯＪＥＣＴによって表わされている。この２つのＰＲＯＪＥＣＴリレーションは、ＰＲＤＢＳのＰＲＯＪＥＣＴリレーションの部分集合になっている。これはセマンティックリンクＳＬ：$\mathrm{PROJECT}_{\mathrm{PMDBS}}$〔pno、pname、syar、eyar〕⊂$\mathrm{PROJECT}_{\mathrm{PRDBS}}$〔proj－no、projname、projsyar、projeyar〕によって表わされる。よってもとめるＧＣＳ ＰＲＯＪＥＣＴリレーションは、

1）サイト１のＰＲＯＪＥＣＴとサイト４のＰＲＯＪＥＣＴの結合

2）サイト２のＰＲＯＪＥＣＴとサイト４のＰＲＯＪＥＣＴの結合

3）1）と2）との結果の和　　によって求められる〔図6.5を参照〕。

ＧＳＤＬによって、これを表わすと、図6.7のようになる。図中の副定義につけられた番号は、図6.5内の結合（join）に付けられた番号と一致している。

```
drange (p1、p2、p) (PROJECT:1、PROJECT:2、PROJECT:4);
define ESR PROJECT(pno、pname、manager/PMEMBER、budget、loc)
(1) { (p1.pno、p1.pname、manager,=p1.leader p1.budget、p.loc)
      where p1.pno = p.proj-no and p1.pname = p.projname and
            p1.syar = p.projsyar and p1.eyar = p.projeyar :
(2) { (p2.pno、p2.pname、manager = p2.leader、p2.budget、p.loc)
      where p2.pno = p.proj-no and p2.pname = p.projname and
            p2.syar = p.projsyar and p2.eyar = p.projeyar ;
```

図6.7　ＧＣＳリレーションＰＲＯＪＥＣＴの定義

6.3 分散情報（ＤＩ）

分散記述（ＤＤ）は、主に次の２つのことを行っている。

1) ＧＣＳリレーションのスキームを定義すること、及び

2) ＧＣＳリレーションとＬＣＳリレーションとの対応関係を述語形式で定義すること。

ＤＤが参照するＬＣＳリレーションとＬＣＳ属性が、drange文で定義したサイトに存在するかどうかは、対応するサイトの異種性情報（ＨＩ）を用いてチェックされる。全体管理者（ＧＡ）が定義したＤＤは、分散情報(distribution information or ＤＩ）としてネットワークデータディレクトリ（ＮＤＤ）内に格納される。分散記述（ＤＤ）及びＨＩから分散情報（ＤＩ）を生成するシステムは、分散情報生成システムＤＩＰＳ(distribution information processing system)と呼ばれる。ＤＩＰＳの詳細については8.3を参照されたい。

ＤＩは、情報をリレーショナル形式で管理している。大別してスキーマ情報（ＳＩ）、対応情報（ＣＩ）、所在情報（ＬＩ）の３種から成っている。ＳＩは、定義されたＧＣＳリレーションのスキームについての情報を保持している。ＣＩは、ＧＣＳリレーションとＬＣＳリレーションとの対応情報を保持している。この対応関係はＤＤで定義された述語論理式として格納される。図6.8内のＣＩ属性sub−defは、この定義式を値として取る。所在情報（ＬＩ）は、ＬＣＳリレーションがどのサイトに所在するかを示している。

図6.8から解かる様に、分散情報（ＤＩ）と異種性情報（ＨＩ）とは共通情報を持っている。例えば、ＬＣＳのリレーション名、属性名はそれである。又、ＬＣＳリレーションについてのパフォーマンス情報、e.g. カーディナリティ、選択度も、ＤＩが持つならばその例となる。これらの冗長データをどのように分散し管理するかは重要な問題である。この分散形態は、ＤＩを主に用いる問合せ分割（ＱＤ）のアルゴリズムに依存している。問合せ分割がより正確に詳細に手順を決めるものであれば、ＤＩは十分なかつ最新のパフォーマンス情報を必要としてＨＩとの間の冗長度は高くなる。データが時間変化がなく利用度の高いものならば冗長度を高めることはデータの獲得性（availability）の点から望ましい。しかし、我々は、これらのパフォーマンス情報は、スキーマ情報と比して動的な性格を持つと考える。又、問合せ分割を行うサイトは、その処理が必要とする情報、即ち分散情報（DI）を自分のサイトに持つべきである。もしそうでないならば、問合せ分割の処理効率は、他のサイトにある分散情報を獲得するためのサイト間通信のために、著しく低下してしまうだろう。よって分散情報がパフォーマンス情報（カーディナリティ、選択度等）を含むならば、動的な情報を多くのサイトで冗長に保持することになってしまう。このことは、冗長コピーへの同時アクセスと一致性の制御のための複雑な処理〔ＢＥＲＮＰ　７８〕のためのオーバヘッドを生み出してしまう。従って我々は、これらのパフォーマンス情報は、対応する異種性情報（ＨＩ）内にのみ非冗長に保持されるとする。更に、静的な特性を持つＬＣＳリレーションのスキームについての情報とそれの所在についての情報を、分散情報と異種性情報とが冗長に保持することとする。これらの情報は、対応するＨＩ内、及び

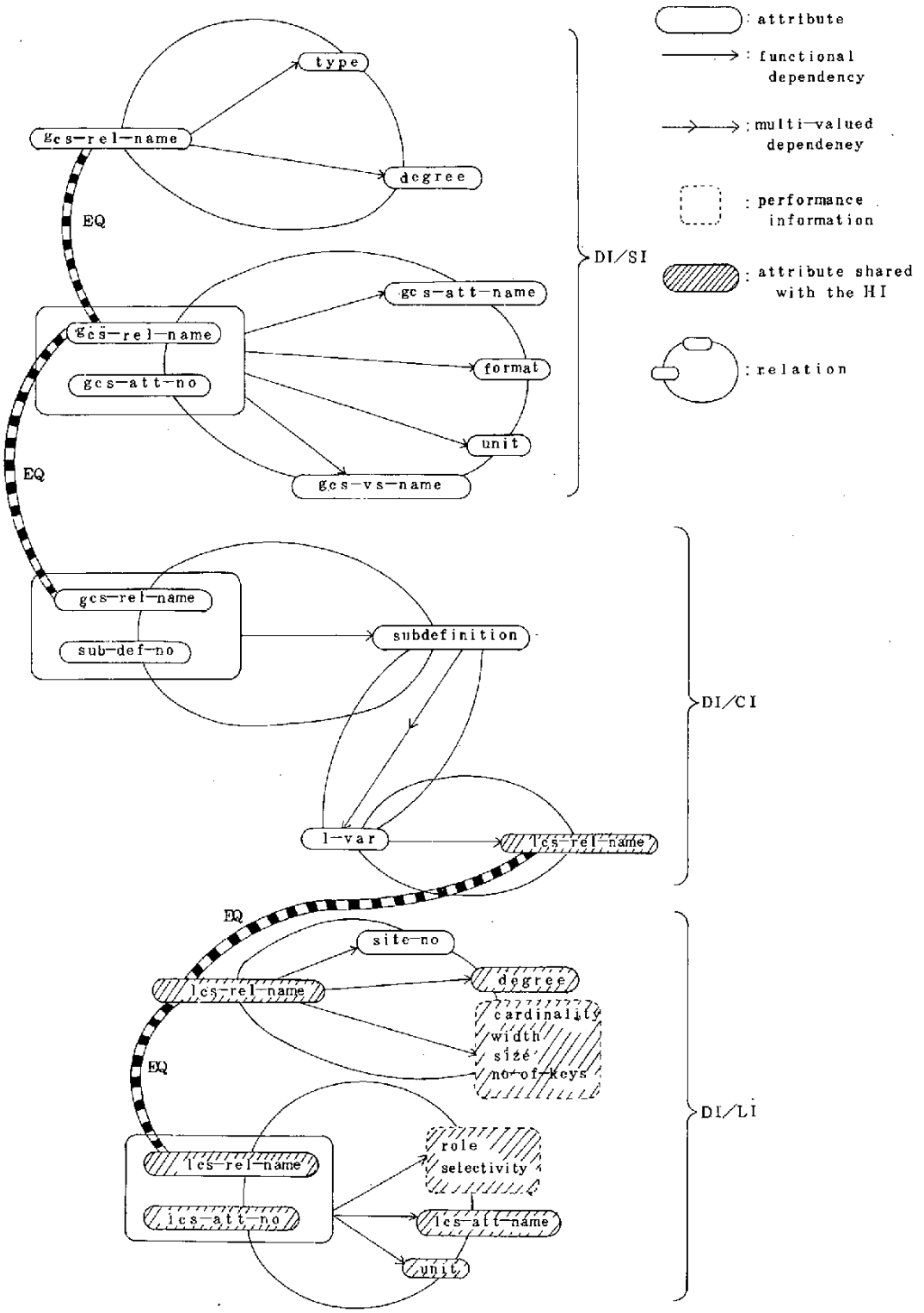

: attribute
: functional dependency
: multi-valued dependeney
: performance information
: attribute shared with the HI
: relation
type
gcs-rel-name
degree
EQ
DI/SI
gcs-att-name
gcs-rel-name
format
gcs-att-no
unit
gcs-vs-name
EQ
gcs-rel-name
subdefinition
sub-def-no
DI/CI
l-var
lcs-rel-name
EQ
site-no
lcs-rel-name
degree
cardinality
width
size
no-of-keys
EQ
DI/LI
role
selectivity
lcs-rel-name
lcs-att-name
lcs-att-no
unit

図 6.8　分散情報（ＤＩ）

分散情報を持つ各サイトとで冗長に保持されることになる。分散情報（ＤＩ）の管理方式は我々の主要テーマであり、第７章の問合せ分割においてさらに論じられる。

6.4 ＤＩＰＳ

分散情報生成システム（ＤＩＰＳ）は、全体管理者（ＧＡ）が全体概念スキーマ（ＧＣＳ）記述用言語ＧＳＤＬによって記述した分散記述（ＤＤ）と、各ＬＣＳリレーションがどのサイトにあるかを示す所在情報（ＬＴ）とを入力として分散情報（ＤＩ）を生成するものである〔図6.9〕。所在情報location information or ＬＩ）は、各サイトの異種性情報（ＨＩ）から生成され、分散情報(DI)の一部となる。DIPSの詳細は8.3節で論じられるので、本節ではその概要を述べることにする。

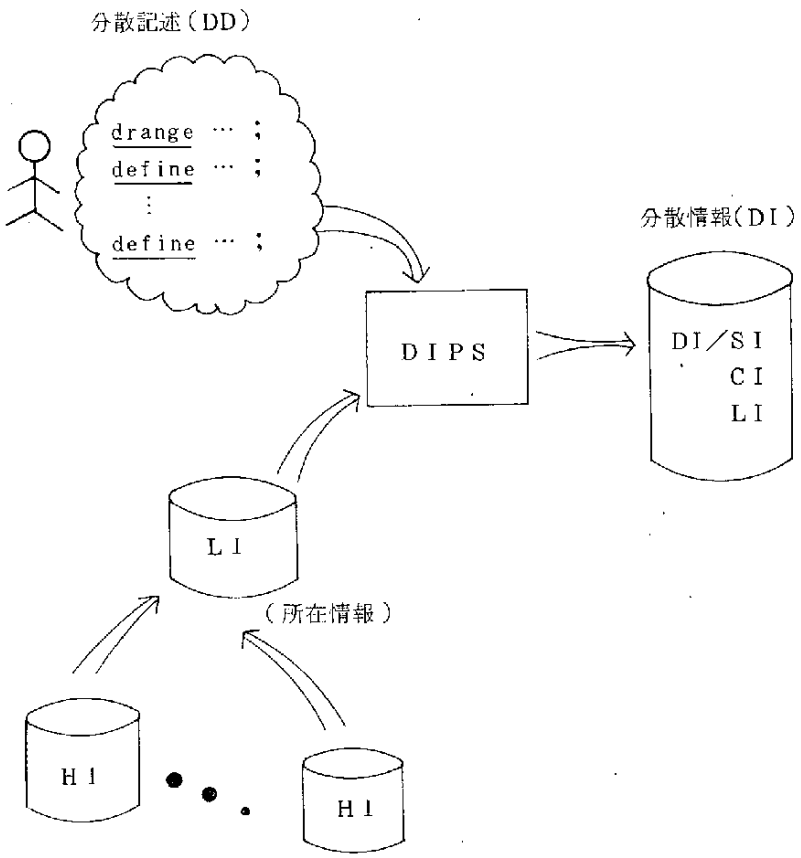


図6.9 ＤＩＰＳの概要

ＧＳＤＬによって記述された分散記述（ＤＤ）は、内部的に２分木表現される。これをＤＤ一木と呼ぶ〔図6.10〕。図6.10内のＤＤ木内のＧＢＲノードは、１つのdefine文内の副定義（<sub－def>）の定義木を乞手に持つ。各副定義を表わす木は、問合せのＱ一木〔8.2〕と同じである。１つの分散記述（ＤＤ）に対して、このなかで参照される組変数は一意である。即ち、どのdefine文も、それが参照する組変数はユニークなＬＣＳリレーションに対応している。さらに１つの分散記述内で参照される属性に対しては、ただ１つの属性ノードが生成されるだけである。即ち、２つのdefine文があるＬＣＳ属性ｘ．ａを参照していたとしよう。この時２つのＤＤ一木は、ｘ．ａに対応する同一のノードを、ｘ．ａとして参照することになる。このためにＡＴＴＢＬテーブルは、属性名とノードへのポインタとの対応表になっている。

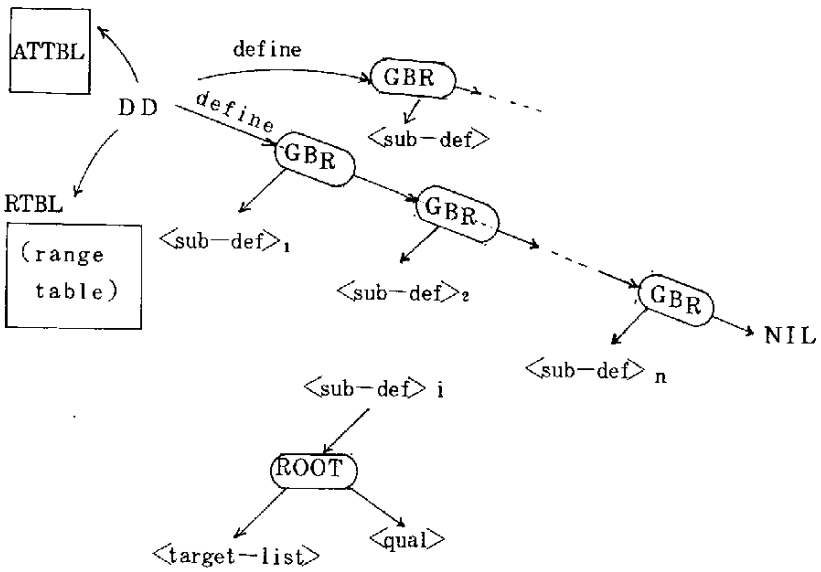


図6.10　DD 一 木

2分木表現された各副定義<sub−def>は、分散情報内の対応情報（correspon−dence information or CI）内に格納される。問合せ分割の問合せ変形〔7.5を参照〕時に、CIから必要な副定義が読み出され、全体概念スキーマ（GCS）要素を参照する問合せを、対応する局所概念スキーマ(LCS)要素を参照する問合せに変形される。

define文は、GCSとLCSとの対応関係を述語論理式で定義するとともに、GCSリレーションのスキームも同時に定義している。このスキーム定義から、分散情報内のスキーマ情報（schema information or SI）が生成される。SIは、GCSリレーション名、タイプ及びGCS属性構成、各GCS属性の領域等についての情報を保持している。

6.5　まとめと問題点

本章では、各サイトに存在するLCSのRDからGCSのRDの定義と、この定義されたGCSとLCSとの対応情報の保持について論じた。前者は統合化であり、後者は分散情報（DI）である。統合化においては、従来の視野（view）の定義演算子（結合、射影、制限）に加えて、和演算(union)が必要となる。このため、和演算を定義できる様にQUELを拡張した。これをGSDLと呼ぶ。更に、GAがGCSリレーションを定義するためのツールとしてセマンティックリンク（SL）の概念を導入した。SLは、2つのLCSリレーション間に存在し、互いのリレーション内のどの部分集合同志が意味論的に等価であり、それらの集合論的関係性はどうかを示している。2つのリレーションの部分集合が意味論的に等価であるとは、2つの部分集合のスキームが等価であり、かつ各々の対応する属性の属する領域が同一であることを意味している。

この様に本章では、GCSの構文的定義について検討した。次の問題は、いかにGCSを設計するかである。GCSの設計は、既に存在しているLCSと、GAの属する共同体の各アプリケーションの要求とを考慮してなされねばならない。この過程は、要求工学における要求仕様を明らかにしてい

く過程と類似している。これらの多くの試みは、分割的（top－down）な設計を目指している。我々のＤＤＢＳの統合的（bottom—up）設計においては、既にデータは存在してしまっているという主要な制約を考慮せねばならない。ＧＣＳ設計過程の検討は今後の重要なテーマである。

加えて、ネットワーク共同体とは何かということである。ネットワーク共同体を構成するアプリケーションとは何かも問題になる。

ＧＣＳ生成後、共同体、アプリケーションの変化に、どのように適応していくかも問題である。設計から運用、環境の変化への適応、新しいアプリケーションの要求……etc.というＤＤＢＳのライフサイクルをどのように考えていくかは重要である。

分散情報（ＤＩ）には、我々は、カーディナリティ、選択度といったパフォーマンス情報はもたせないとしている。パフォーマンス情報は次の様な特徴をもっている。

1）異種性情報（ＨＩ）と冗長している。

2）動的な性質を持つ。

さらに、ＤＩへの要求としては次の点がある。

3）問合せ分割（ＱＤ）〔７章を参照〕を行うサイトには分散情報（ＤＩ）の完全コピーが必要。

我々は、以上の点より、動的な性質のデータを冗長に保持するための同時実行及び一致性制御問題を避けるために、ＤＩにはパフォーマンス情報をもたせないことにした。しかし、冗長コピーの存在は、完全な一致性と同時実行制御（以降簡単のためにＣＣ制御と呼ぶ）を必要するとは限らない。このＣＣ制御の度合は、データの性質と利用要求によって決まる。例えば、数１０万組をもつリレーションの中の数組が除去されたとしても、このリレーションのカーディナリティの他のリレーションに対する重みはほとんど変わらないであろう。よって、除去された後、ある程度の時間後にカーディナリティの変化をコピーＤＢに伝えても、ＱＤにはほとんど影響はないであろう。よって我々は、ＤＩ内にカーディリティ程度のパフォーマンス情報はもたせることを検討している。

7. 問合せ分割（QD） 〔TAKIM80b.c〕

全体概念スキーマ（GCS）層では、各データベースシステムの異種性問題と、分散問題とは解決されている。即ち、分散型データベースシステムは、この層において、1つのスキーマとしての全体概念スキーマと、1つのアクセス言語としてのQUEL〔HELDG75〕とを提供する1つの巨大システムへと仮想化されている。従って、分散型データベースシステムは、この全体概念スキーマ（GCS）層で発せられた問合せ（GCS問合せ）から、各データベースシステムごと又はデータベース間で実行可能な局所概念スキーマ（LCS）層の問合せ(LCS問合せ)のシーケンスを生成せねばならない。我々は、これを問合せ分割（query decomposition)と呼ぶ。全体概念スキーマ（GCS）層と局所概念スキーマ（LCS）層との対応関係は全体管理者（GA）によって分散記述（DD）として定義されている。この対応関係は、分散情報（DI）として、各サイト（実際には全体データベースプロセッサ（GDP）〔3.3〕）ごとに保持されている。問合せ分割は、この分散情報を用いて、全体概念スキーマ問合せ（GCS問合せ）を各サイト又はサイト間で実行可能な局所概念スキーマ問合せ（LCS問合せ）に分割する。問合せ分割は、統合化の逆過程とも考えられる。

問合せ分割（QD）は、次の2つのことを行なう必要がある。

1) GCS問合せの参照する全体概念スキーマ要素（i.e.リレーション、属性）を、対応する局所概念スキーマ要素に分割する。

2) 1)で分割された問合せは、一般に複数サイトを参照しているので、サイト間で問合せ処理をするための、サイト間でのデータの転送方法を決定する。

1)に対しては、全体概念スキーマ層のリレーション（GCSリレーション）の定義が、リレーショナルモデルの視野（view）の定義と等価であるので〔第6章〕、問合せ変形（query modification)〔STONM76〕技術を用いることができる。2)は、分散型データベースシステム固有の問題である。即ち、分散型データベースシステムでは、異なったサイトに存在するリレーションの結合（join）を行なうためには、どちらか一方を他方へ通信回線を用いて転送しなければならない状態が存在する。各データベー

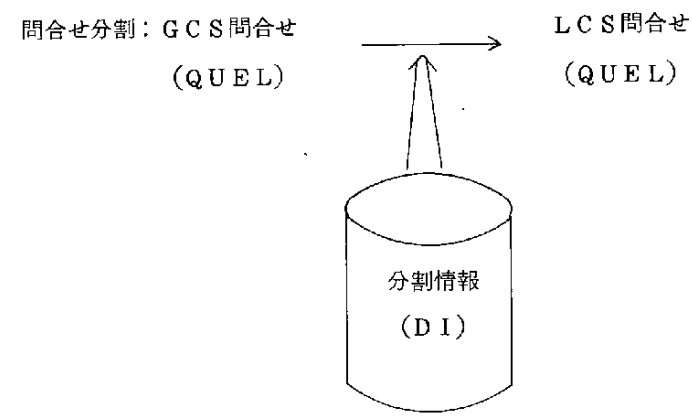


図7.1 問合せ分割の概要

スシステムを結合する通信回線は、その容量の制約から分散型データベースシステムのボトルネックを形成するので、サイト間での処理パフォーマンスの決定要因となる。よって2)の問題は、問合せ分割の主要テーマとなり、現在までに〔HEVNA78〕〔WONGE77〕〔EPSTR78〕〔CHUW79〕等によって多くの試みがなされている。これらが応答時間と通信コストとを最少化するための戦術決定を実行前に決めてしまうのに対して、処理のための必要情報をなるべく少なくかつ更新頻度の低いものだけに押えるアルゴリズムを本章で提案する。

また問合せ分割は、分散型データベース・システムの設計方法と密接に関連している。分割型設計（top-down designing）の場合には、分散型データベースシステムのアプリケーションの要求に合うようにデータを最適配置できる。これに加えて、データを各サイトが冗長に持つことを許せば、多くの問合せの処理をより少ないサイトに局所化できる〔MAHMS76〕。更に、中間結果の見積もりも容易である。例えば、あるリレーションRがR1とR2とへ水平分割され、各々異なったサイト1と2とに分散されたとしよう。又、R1とR2とは共有部分を持たないとしよう。このとき、R1とR2との和（union）演算の結果リレーションのサイズは、R1とR2の各々のサイズの和となる。次にあるリレーションRから垂直分割された2つのリレーションR1とR2とを考えてみよう。R1とR2とを結合（join）した結果リレーションのカーディナリティとして、R1とR2のカカーディナリティのうち小さいものをとることで十分であろう。

しかし、統合型設計（bottom-np designing）の場合には、設計以前にデータは各サイトに存在してしまっている。これらのデータの分散は、アプリケーションの要求に合うものでないかもしれない。又、2つのサイトにある2つのリレーションR1とR2とがある時、この2つは互いに共通部分を持ち得る。よって、R1とR2との和の場合も結合の場合も、その結果リレーションのサイズを見つもることはかなり困難であろう。

7.1 基本仮定

問合せ分割の最適化において、サイト間でのデータの転送は、通信ネットワークを介して行なわれる。このため通信ネットワーク（以降単にネットワークと書く）は重要な役割を持っている。ネットワークが光ファイバ、衛星通信、無線通信を用いることによって、その通信コストが大幅に減少し、大量データの高速転送や放送（broadcast）通信が可能となると問合せ分割の手法も大きく変わってくる。このために、我々は、ネットワークに次の様な仮定を設けた。

1) 通信コストに比して、各サイトの処理コストは無視できる。
2) ネットワークは、site-to-site であり放送ネットワークではない。即ち、通信コストは、あるデータが転送されるサイト数に依存する。
3) ネットワークは軽負荷である。このためネットワーク内の各ノードでの待ち行列生成による遅延はないとする。

4) 通信コストは距離に依存する。〔ＨＥＶＮＡ７８〕は、上記３つの仮定に対して通信コストの距離独立性を仮定しているが、現在のネットワーク技術からみて、これは現実的ではないと考える。ここで論理転送コストLCｉｊを次のように定義する。LCijは、サイトｉからサイトｊへ１つのパケットを転送するのに要する遅延である。LCijは、ｉとｊ間のノードをホップするＩＭＰ数に比例するとする。

更に、問合せ分割の処理上、次の様な仮定を設ける。

1) 各サイトのデータベースシステムは、リレーショナルデータベースシステム（ＲＤＢＳ）として管理できる作業領域（ｗｏｒｋｉｎｇ ｓｐａｃｅ ｏｒ ＷＳ）を備えている。問合せ分割の処理中に生成される中間結果は、この作業領域ＷＳ内に格納される。又、他サイトから転送されてくる結果リレーションもＷＳ内に格納され、必要な演算（ｅ.ｇ.ｊｏｉｎ, ｐｒｏｊｅｃｔｉｏｎ）が行なわれる。このＷＳの管理システムを作業領域（ＷＳ）管理システム（ＷＳ ｍａｎａｇｅｒ ｏｒ ＷＳＭ）と呼ぶ。ＷＳＭを各サイトで容易にインプリメントできるかどうかは、分散型データベースシステム設計において重要な要因となろう。しかし、後述するように、我々はＷＳＭのインプリメンテーションは容易であると考えている。

2) 各サイトは、２つの論理プロセッサを持っているとする〔4.3節を参照〕。これは１つ以上の局所データベースプロセッサ（ＬＤＰ）と１つの全体データベースプロセッサ（ＧＤＰ）とである〔ＴＡＫＩＭ７９ａ〕。全体データベースプロセッサ（ＧＤＰ）は問合せ分割、統合化、分散情報（ＨＩ）の管理とを行なう、局所データベースプロセッサ（ＬＤＰ）は、１つのデータベースシステムに対して1つ存在し、問合せ変換、同種化、異種性情報（ＨＩ）の管理、更に作業領域（WS）の管理とを行なう，又、ユーザが問合せを発する全体データベースプロセッサを、この問合せ処理のための統合全体データベースプロセッサ（ｃｏｏｒｄｉｎａｔｅ ＧＤＰ ｏｒ ＣＧＤＰ）と呼ぶ。ＣＧＤＰは、問合せ処理の集中コントローラの役目を持つ。即ち、問合せが必要とするデータを保持するＬＤＰは、このＣＧＤＰの制御のもとで必要な処理を行なう。

3) 次に、２つのリレーションr′とｒが、各々サイトｉとｊとにあるとしよう。

このr′とｒとの結合（ｊｏｉｎ）を行なうために、次のような仮定を設ける。r′がサイトｉからｊへ転送され、サイトｊでr′とｒとが結合されるとしよう。この時r′とサイトｉとを各々ソースリレーション（ｓｏｕｒｃｅ ｒｅｌａｔｉｏｎ）とソースサイトと呼ぶ。さらにｒとサイトｊとを各々目的リレーション（ｄｅｓｔｉｎａｔｉｏｎ ｒｅｌａｔｉｏｎ）と目的サイトと呼ぶ。r′をサイトｊへ転送し、サイトｊでｒとr′との結合を行なうことを、ステージ（ｓｔａｇｅ）と呼ぶ。ステージは、サイト間問合せ処理の基本単位である。後述する〔７.７〕転送スケジューリング（ＴＳ）とは、このようなステージの最適シーケンスである。

又、LDPｋは、サイトｋにあるＬＤＰを表わすとする。

7.2 目　　標

7.1節で述べた様に、問合せ分割（ＱＤ）は、次の２つの主要な処理から成っている。

1) 全体概念スキーマ層の問合せ（ＧＣＳ問合せ）を、対応する局所概念スキーマ層のリレーション（ＬＣＳリレーション）を参照する問合せに変換する。この変換された問合せを全体ＬＣＳ問合せ（global LCS query or GLQ）と呼ぶ。

2) 全体ＬＣＳ問合せ（GLQ）のサイト間結合（inter-site join）を処理するために、対応するサイト間でのデータの転送方法を決める。

前者は表現の変換であるのに対して、後者は実際のサイト間でのデータの転送を行なわねばならずそれはＤＤＢＳのパフォーマンスを決定する要因となる。このため、どのようにサイト間でデータを転送するかは重要であり、最適な転送シーケンスを見つけねばならない。この最適化のための目標としてはこれまで次の２点がとりあげられてきた〔ＨＥＶＮＡ７８，ＷＯＮＧＥ７７〕。

1) 通信コストの最少化，i．e．全処理時間の最少化。

2) 応答時間の最少化。

分散型データベースの通信ネットワークは、容量の制約から、システムの主要なボトルネックとなっている。このため、なるべくネットワークを流れるトラフィック量を少なくすることが望ましい。このことは 1) の目標を達成することである。一方で、分散型データベースシステムは各サイトが各々データベースシステム（ＤＢＳ）としての処理機能を保有している。このことは、各サイトが処理を並列に行なえることを示している。一般に、並列処理は、応答時間を短縮できる(i. e. 2) の目標の達成)。しかし、並列度を高めることは、各サイト間の処理同期をとるためのサイト間トラフィックを増大させ、全通信コストを高めてしまうことにもなる。このように、通信コストの最少化と応答時間の最少化との間には、トレードオフが存在している〔ＨＥＶＮＡ７８〕。よってこれらの目標をうまく達成することが必要である。

これまでなされてきた多くの試み〔ＣＨＵＷ７９，ＥＰＳＴＲ７８，ＷＯＮＧＥ７７）は、上記した２つの目標を達成しようとするものである。これらの特徴は、次の２点である。

1) 実行前に、全ての転送方法を決めてしまう。

2) 転送方法の決定は、中間結果のサイズの見積もりに基づいている。

中間結果サイズは、問合せの参照するリレーションについてのカーディナリティ（cardinality）その属性の選択度（selectivity）等のパフォーマンス情報に基づいて見積もられる。さらにオフライン的に転送方法を決定するためには、ディレクトリ（我々はこれをネットワークデータディレクトリ又はＮＤＤと呼んでいる）内に全サイトのリレーションについてのパフォーマンス情報(e.g.選択度、カーディナリティ）を持つ必要がある。

選択度について考えてみよう。ここでＲをリレーション、ａをＲの属性と、s_a を属性ａの選択度としよう。選択度とは、ある属性ａとある値ｖに対して問合せの条件 Ｒ．ａ＝ｖ を満足する組(tuple)がリレーション内にどの位あるかを示したものである。〔ＨＥＶＮＡ７８，ＳＥＬＩＰ７９〕

らは、選択度 s_a を次のように定義している。

s_a　R〔a〕の組数／Rの組数

この定義は、リレーションR内で属性aのとる値が均等に分散していることを前提にしたものである。我々は、この仮定が実際のリレーションにおいてよくあてはまるかどうかは疑問と考えている。パフォーマンス情報をより正確とするためには、属性と値の対に対して各々選択度を定義せねばなるまい。このため〔CHUW79〕は、よく使われる値ごとにその属性の選択度をディレクトリに蓄積していく方式を考えている。例えば　R.a＝v　という条件式が、値vに対して良く用いられるとしよう。そして、この条件式を満足する組が、あるアクセスの結果として、α個あったとしよう。Rのカーディナリティを C_R とする。するとディレクトリ内に、属性aは値vに対して、α/C_R の選択度をもつという情報が格納されることになる。しかし、この方法は、より正確な決定をするためには大きな格納領域を必要としてしまう。

上述したような選択度の問題点に加えて、このようなパフォーマンス情報についての我々の主張は、これらがスキーム情報に比してより動的な性質をもっているという点である。更に、問合せ処理（即ちQD）が必要とする情報（データ）は、それと同じサイトになければならない。この点は処理のパフォーマンス上必要不可欠である。我々のアーキテクチャでは、サイトは1つの全体データベースプロセッサ（GDP）を備えている。この様に、少なくとも複数の問合せ分割機能を分散型データベースシステムが持つ理由は、処理のパフォーマンスからの要請に基づいている。何故なら分散型データベースシステム内の1つのサイトのみが問合せ分割機能を持つならば、そのノードは通信トラフィックが集中するとともに信頼性の問題をもたらしてしまうからである。以上のことより、選択度、カーディナリティといった情報を分散情報（DI）が持つならば、次のような問題が生じる。即ち、動的な情報が冗長に各サイトに分散されることによって生ずるこれらの情報の一致性とインテグリティの保持との制御、及びこれらへの同時アクセスの制御問題及び格納オーバヘッド問題である。冗長かつ動的データの一致性と同時実行との制御（consistency and concurrency control)(我々は、これをCC制御と呼ぶ）は、分散型データベースシステム全体のオーバヘッドを著しく増大させてしまう。

さて、問合せ分割の処理に必要な情報、即ち分散情報（DI）は、可能な限り小さくかつ静的（時間とともに変化しない）であるべきである。我々は前述した全通信量の最少化と応答時間の最少化との2つの目標に加えて、最も重要な目標として"DIの最少化と静的保持"とを加えたい。

これまでの議論をまとめると次のようになる。

1) 最適化の目標としては　i）分散情報（DI）の最少化と静的保持
ii）最少通信量，及び
iii）最少応答時間　　の3点がある。

∴ 我々は、このうち特に i) の目標に注目する。

2) 問合せ分割（QD）機能は分散型データベースシステム内の全サイトが持つ。よって全サイトは問合せ分割処理の必要情報、即ち分散情報（DI）を処理とあわせて持つ。

3) このため、分散情報内には選択度、カーディナリティといったパフォーマンス情報を保持しな

い。何故なら、これらは動的性質を持つからである。

4) オフライン的に、実行前に全ての転送方法を決めるのに必要な十分な情報を選択度について持つことは出来ない。このことは、転送方法を実行前に前もって決定するのではなく、実行中にその時々の状況をみながら決めていく方法が、サイト間問合せ処理のための良い候補となり得ることを示している。

7.3 戦　略

これまでの議論を基にして、我々は次の様なサイト間問合せ処理の戦略を設けた。

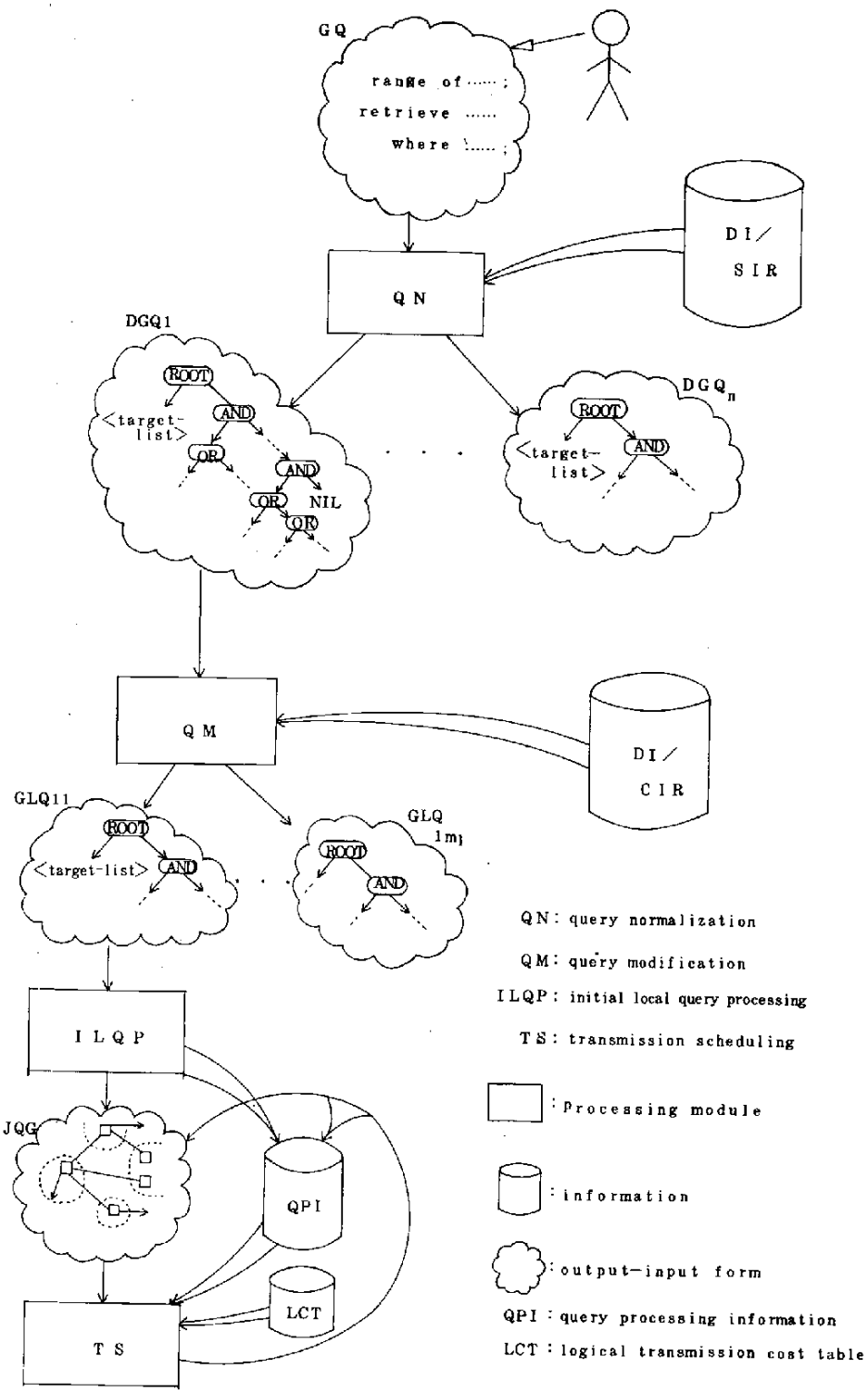
1) 統合全体データベースプロセッサ（ GDP）は、これまでのステージの実行によって生成されたリレーションについての統計情報を基にして、次のステージを動的に決定する。この動的決定によって分散情報内の情報をより少なくかつ静的なものとすることができる。

2) 問合せ内で、各サイトで独立に局所的に処理できる部分は出来るだけ早く処理してしまう。しかる後にサイト間にまたがった部分の処理を行なう。この様な局所処理によってリレーション内の必要な部分のみが転送されることになり、ネットワーク内のトラフィックを減少できる。各サイトでまず局所処理することの前提には、このコストが通信コストに比して無視し得るという仮定〔7.1を参照〕がある。

3) 各々異なったサイトにある2つのリレーションのサイト間結合（inter-site join）を行なう時、より小さいものをより大きいものへ転送し、そこで結合を行なう。我々は、第3のサイトへこれらのリレーションを転送し、そのサイトで結合をとる様な方式はとらないことにする。理由は、2つのリレーションをともに転送するよりは、より小さいものだけを転送した方が通信コストを少なくできるからである。

4) CGDPによって発せられたステージが全て終了していなくても、転送コストがある閾値（threshold value）より小さく、かつ処理中でないリレーションがあれば、これも転送してしまう。このことは、並列して処理可能なステージを、並列して処理させ得ることを示している。1つのステージ内でのソースリレーションの転送コストは、ステージコストの大半を占めている。よってステージの並列処理によって、リレーションの転送を並行して行なえるので応答時間を短縮できる。

我々の問合せ分割は、図7.2に示すように、問合せ正規化（ query normalization or QN）、問合せ変形（query modification or QM）、初期局所問合せ処理（initial local guery processing or ILQP）、及び転送スケジューリング（transmission scheduling or TS）の4つの主要モヂュールから構成されている。問合せ正規化（QN）と問合せ変形（QM）とは、GCS問合せをLCSリレーションのみを参照する問合せ；i.e. 全体LCS問合せ（GLQ）への変換を行なう。即ちこれらは、分散情報（DI）内の対応情報を用いて、問合せの表現の変換を行なう。初期局所問合せ処理（ILQP）と転送スケジューラ(TS)とは全体LCS問合せ（GLQ）を用いてサイト間の問合せの処理を行なう。ILQPは 2) で述


GQ
range of ..... ;
retrieve .....
where ..... ;
DI/
SIR
QN
DGQ1
ROOT
AND
<target-list>
OR
AND
OR
NIL
OR
DGQn
ROOT
AND
<target-list>
QM
DI/
CIR
GLQ11
ROOT
<target-list>
AND
GLQ
1m1
ROOT
AND
ILQP
JQG
QPI
LCT
TS
QN: query normalization
QM: query modification
ILQP: initial local query processing
TS: transmission scheduling
: Processing module
: information
: output-input form
QPI : query processing information
LCT : logical transmission cost table

べた各サイトでの局所処理を可能な限り行なわせるものである。ＴＳは、局所処理の終った後、各サイト間の結合（ｊｏｉｎ）を行なうために、各サイト間でのリレーションの転送を行なわせるものである。以降これらの４つのモヂュールの各々について論じる。

7.4　問合せ正規化（ＱＮ）

ユーザによって出された全体概念スキーマ層の問合せ（ＧＣＳ問合せ）の条件部は、ＧＣＳリレーションを参照する比較述語（comparison predicate）を任意の論理演算子で結合したものとなっている。このため、まずこのような条件部を正規化（normalize）せねばならない。我々は、条件部を和正規形（disjunctive normal form or DNF）〔図7.3を参照〕に正規化する。図7.3 a）の中の＜a－exp＞は、ＧＣＳリレーション属性（これをＧＣＳ属性と呼ぶ）と定数とからなる算術式を表わしている。和正規形（ＤＮＦへの変換は図7.3 b）の様な変換規則を用いてなされる。

$$\langle qual\rangle ::= \langle disjunct\rangle_1 \vee \cdots \vee \langle disjunct\rangle_m$$
$$\langle disjunct\rangle_i ::= \langle c\text{-}pred\rangle_{i1} \wedge \cdots \wedge \langle c\text{-}pred\rangle_{imi}$$
$$\langle c\text{-}pred\rangle_{ij} ::= \langle a\text{-}exp\rangle \langle cop\rangle \langle a\text{-}exp\rangle$$
$$\langle cop\rangle ::= < \mid \leq \mid = \mid \geq \mid > \mid \neq$$

図7.3 a)　DNF

$$a \wedge (b \vee c) \rightarrow (a \wedge b) \vee (a \wedge c)$$
$$\sim (a \wedge b) \rightarrow \sim a \vee \sim b$$
$$\sim (a \vee b) \rightarrow \sim a \wedge \sim b$$
$$\sim (\sim a) \rightarrow a$$
$$\sim \alpha \ cop \ \beta \rightarrow \alpha \sim cop \ \beta$$
$$\sim \begin{pmatrix} < \\ \leq \\ = \\ \geq \\ > \\ \neq \end{pmatrix} \rightarrow \begin{pmatrix} \geq \\ > \\ \neq \\ < \\ \leq \\ = \end{pmatrix}$$

a，bは、任意の論理式　　α，βは　＜a－exp＞

図7.3　b）ＤＮＦへの変換規則

ＧＣＳ問合せはＤＮＦに変形された後、各＜disjunct＞i（i＝1,2,…,m）を条件部とする問合せへ分割させる〔図7.4を参照〕。この分割をＧＣＳ問合せの水平分割（horizontal query decomposition or ＨＱＤ）と呼ぶ。さらに水平分割（ＨＱＤ）された問合せを分割されたＧＣＳ問合せ（decomposed GCS query or DGQ）と呼ぶ。図7.4において、TLはGCS問合せの目標リスト（target-list）を表わしている。Riはi番目の

$$GQ: \begin{array}{l} \underline{\text{range}}(g1,\cdots,gk)(G_1,\cdots,Gk); \\ \underline{\text{retrieve into}}\ R\ (TL) \\ \underline{\text{where}}\ \langle disjunct\rangle_1 \vee \cdots\cdots \vee \langle disjunct\rangle_m; \end{array}$$

$$\Downarrow \text{HQD}$$

$$\left\{ DGQ_i: \begin{array}{l} \underline{\text{range}}\ (g_1,\cdots gk)(G_1,\cdots Gk); \\ \underline{\text{retrieve into}}\ (TL) \\ \underline{\text{where}}\ \langle disjunct\rangle_i; \end{array} \middle| \ i = 1,2,\cdots,m \right\}$$

図7.4　ＧＱの水平分割

<disjunct> i に対応する分割されたＧＣＳ問合せ（ＤＧＱ）の結果リレーション名である。図7.4から解かる様に、全てのＤＧＱi の目標リストは同じなので、これの結果リレーションRn のスキームは同一である。（ｉ＝1,2,…，ｍ）。よって最初のＧＣＳ問合せの結果リレーションＲはこれらのR_i の和（union） となる。即ちR＝R_1 U R_2 U…URｍである。水平分割されたＧＣＳ問合せ（ＤＧＱ）は、各々互いに独立に処理される。

問合せを和正規形（ＤＮＦ）に正規化する理由は次のようである。

1）P_1, P_2 を2つの比較述語としよう。又、P_1 P_2は、各々1つまたは2つのリレーションを参照するものとする。この時、P_1 or P_2 について考えてみよう。P_1とP_2の参照するリレーションについて次の2つの場合がある。即ち、1つはP_1 とP_2 がともに同じリレーションを参照する場合であり、他は互いに異なったリレーションを参照する場合である。前者の例は x. a ＝5 or x. b＝8（ここでｘは組変数 a，ｂはｘによって表わされるリレーションの属する属性である）のような論理式である。このような論理式は、1つ 又は2つのリレーション間で閉じて処理できる。しかし、後者の場合は、2つ以上のリレーションをまとめて処理することが必要になる。例えば、論理式 x.a=g.b or g. c＝z. d を考えてみよう。これを処理するためには、 x. a＝g. b とg. c＝z. d とを独立に処理し、各々の結果の和をとらねばならない。このため問合せを和正規形（ＤＮＦ）に変換し、水平分割する必要がある。

2）更に、x. a＝5 or x. b＝8 のように同一の変数を参照する2つの比較述語の論理和の場合も1）と同様のことが言える。7.5節で述べるように、我々は問合せ変形（query modification）方法によって、ＧＣＳリレーションを参照する問合せを、LCSリレーションを参照する問合せに変換する。よって、ＧＣＳ属性x. aはあるＬＣＳ属性 l_1 . f へ，x. b は l_2 . h へ変換されるかもわからない。これは1）と同様の状態、即ち、P_1 とP_2 とが各々異なったリレーションを参照する状態を生じさせる。よって、問合せを和正規形（ＤＮＦ）に変換しこれも分割する必要がある。

上述した問合せ正規化（ＱＮ）は、リレーション問合せの論理的取扱いを容易にする。しかし、このことは問合せの最適な処理を行なえることを意味していない。我々は、後述する問合せ変形手法を適用するためには、ＧＣＳ問合を和正規形に変形し、水平分割し、各分割されたＧＣＳ問合せ（ＤＧＱ）の条件部は比較述語の論理積である方が良いと考えている。上述した様な論理和をいかに効率よく行なうかは今後の問題である。我々はユーザの問合せのほとんどのパターンは notと or をふくまないと考えている。即ち、多くの場合問合せ正規化（ＱＮ）を用いる必要はなく、ユーザの入力した問合せは、即、分割された全体問合せ（ＤＧＱ）として、次の処理を行なうことができる。

ＱＮでは、問合せの正規化とともに次のことを行う。

1）分散情報（ＤＩ）のスキーマ情報（ＳＩＲ）〔6.3節を参照のこと〕をサーチして、条件部が目標リスト内の全てのリレーションを結合式（join formula）で連結されているかどうかのチェックを行なう。我々は、問合せ変換〔5章〕と問合せ分割の実験から 必要な結合を入力するのを忘れることによるエラーが、入力エラーの多くを占めることがわかっている。この意味から、この連結性

のチェックは有効である。

2）問合せを2進木表現に変換する。ＧＣＳ問合せは、問合せ変換と同様に、システムの内部表現としてして2進木表現を用いる〔図7.5〕。問合せの2進木表現の詳細については、8.2節を参照されたい。

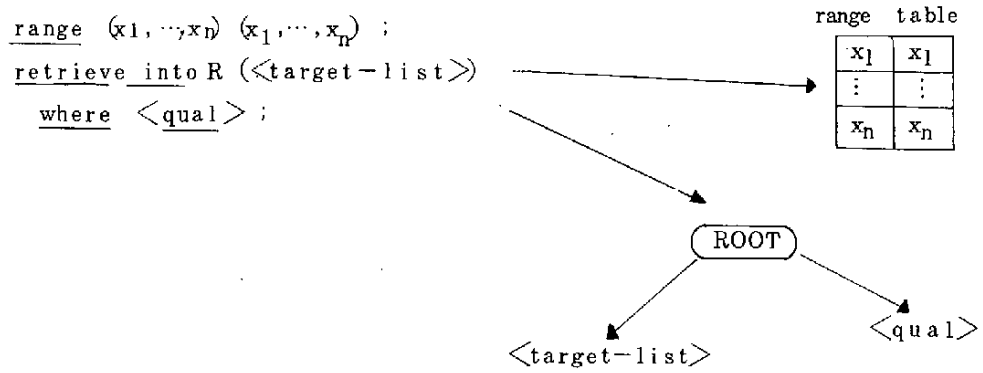


図7.5　ＧＣＳ問合せと2進木表現

7.5　問合せ変形（ＱＭ）

分割されたＧＣＳ問合せ（ＤＧＱ）は、問合せ変形（query modification）手法〔ＳＴＯＮＭ７６〕を用いて対応する局所概念スキーマ層のリレーション（ＬＣＳリレーション）を参照する問合せに変換される。問合せ変形の概要は次の様である。

1）分割されたＧＣＳ問合せ（ＤＧＱ）内のＧＣＳ属性が、ＬＣＳ属性から成る算術式に置き換えられる。この算術式とＧＣＳ属性との対応関係は分散情報（ＤＤ）の目標リストとして定義されており、分散情報（ＤＩ）の対応情報（ＣＩ）〔6.3〕に格納されている。

2）次に、ＤＧＱの条件部に、これが参照するＧＣＳリレーションの定義式の条件部を接合する。このＧＣＳリレーションと条件部との対応関係は、Ｄと同じく分散記述（ＤＤ）の条件部（<qual>）として定義され、分散情報（ＤＩ）の対応情報（ＤＩ）内に格納されている。

第6章で述べたように、全体概念スキーマ層のリレーション（ＧＣＳリレーション）の定義式は、一般に複数の副定義（sub definition）から成り立っている。よって、1つの分割されたＧＣＳ問合せ（DGQ）から、これの参照するＧＣＳリレーションの副定義の各組合せごとに問合せから生成されることになる〔図7.6を参照〕。このようにして生成された問合せを全体ＬＣＳ問合せ（global LCS query or GLQ）と呼ぶ。全体ＬＣＳ問合せ（ＧＬＱ）は、ＧＣＳ問合せ(実際には、分割されたＧＣＳ問合せ（ＤＧＱ））の意味をＬＣＳリレーションによって表わしたものである。

前節の問合せ正規化（ＱＮ）によって生成されるＧＣＳ問合せの2分木表現（これをＧＣＳ問合せ木と呼ぶ）内では、ＧＣＳ問合せの参照するＧＣＳ属性に対してただ1つの属性ノードが生成されるだけである。よって問合せ変形（ＱＭ）では、この属性ノードを分散情報の木表現内の対応するＬＣ

S属性からなる算術式木の根ノードの内容で置き換えることによって問合せを変形できる。この様に問合せ変形は容易にできる。

DD: drange $(\ell_1, \ell_2, \cdots, \ell_m)$ $(L_1 : S_1, L_2 : S_2, \cdots\cdots, L_m : S_m)$;
define <type> G_i (AL_i)
<sub−def>$_{i1}$:<sub−def>$_{i2}$: …… : <sub−def>$_{imi}$;
where <sub−def>$_{ij}$: : = (TL_{ij}) where <qual>$_{ij}$
$i = 1, 2, \cdots\cdots, h$

DGQ: range $(g1, g2, \cdots\cdots gk)$ $(G1, G2, \cdots\cdots, Gk)$;
retrieve into R (TL) where <qual> ;

GLQ:
range $(\ell_1, \ell_2, \cdots\cdots \ell_m)$ $(L_1 : S_1, L_2 : S_2, \cdots\cdots, L_m : S_m)$;
retrieve R_i (TL')
where <qual> ∧ <qual>$_{1i1}$ ∧ <qual>$_{2i2}$ ∧ ……
…… ∧ <qual>$_{kik}$;
$(i_1, i_2, \cdots, ih) \in \{(1.1.\cdots, 1), (1, 1, \cdots, 2), \cdots\cdots, (m1, m2; \cdots mh)\}$

where $R = R_1 \cup R_2 \cup \cdots\cdots \cup R_\alpha$ $(\alpha = m1 \cdot m2 \cdots\cdots \cdot rnh)$
TL′と<qual>は、各々TLと<qual>内のGCS属性を、対応するLCS属性からなる算術式で置き換えたものである。

図7.6 問合せ変形（QM）の概要

例として、8つのLCSリレーションが、次の様なスキームと所在とを持っているとしよう。

サイト1 AAA (a_1, a_2, a_3).
BBB (b_1, b_2, b_3)

CCC (c_1, c_2, c_3)
サイト2 DDD (d_1, d_2, d_3)

サイト3 EEE (e_1, c_2, e_3)

サイト4 FFF (f_1, f_2, f_3)
GGG (g_1, g_2, g_3)
HHH (h_1, h_2, h_3)

これらのLCSリレーションに対して、図7.7のようなGCSリレーションD_1, D_2, D_3が定義されたとしよう。

```
* JIPNET-DDB STARTED TIME 17:40:45
#01
D1;
RANGE ( A, B     )  ( AAA:1, BBB:1          );
RANGE ( C, D     )  ( CCC:2, DDD:2          );
RANGE ( E        )  ( EEE:3                 );
RANGE ( F, G, H  )  ( FFF:4, GGG:4, HHH:4   );
DEFINE ESR  D1 ( D11, D12, D13      )
     ( D11 = B.B1 * 8, D12 = C.C1, D13 = F.F1 +2      )
     WHERE
          A.A1 = B.B1    AND  B.B1 = C.C2    AND  B.B2 = F.F2    AND
          A.A2 = "A"
     ;
DEFINE ESR  D2 ( D21, D22  )
     ( D21 = E.E2, D22 = H.H2 )
     WHERE    E.E1 = H.H2    AND  H.H3 = G.G2
     ;
DEFINE ESR  D3 ( D31 )
     ( D31 =  D.D1   )
     WHERE    D.D1 GE 500
     ;
```

図7.7　分散記述の例

ここで、ＲＡＮＧＥ文は、第６及び本章で述べている ｄｒａｎｇｅ文と等しい。上記された分散記述は、グラフ表現すると図7.8の様になる。図中のノードに（□）は、ＬＣＳリレーションに対応する組変数（ＬＣＳ変数）を表わしている。ノード間のリンクは、これらのノードが表わすＬＣＳリレーション間の結合（ｊｏｉｎ）述語の論理式を表わしている。矢印を持ったリンクは、結果属性リストを示している。短かい平行した２本の棒を端としてもつリンク（ ⊣⊢ ）は、制限式を示している。点線の円は、サイトを表わし、実線の円は、ＧＣＳリレーションを表わしている。

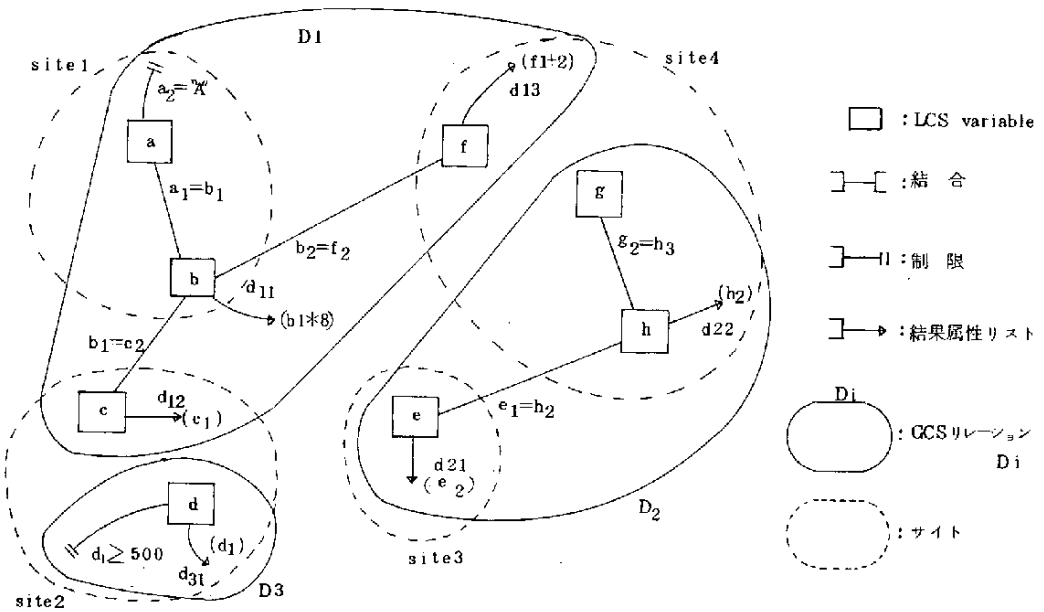


図7.8　ＧＣＳリレーションD1,D2,D3の問合せグラフ表現

次にこの分散記述（ＤＤ）内に定義された３つのＧＣＳリレーションD_1，D_2，D_3に対して次の様なＧＣＳ問合せが発せられたとしよう。

```
@D ;
RANGE ( D1, D2, D3 )    ( D1, D2, D3   );
RETRIEVE INTO RESULT ( R1 = D1.D11,                    R3 = D1.D13 )
WHERE
        D1.D12  =  D3.D31   AND  D1.D13  =  D2.D21   AND
        D1.D11  =  D2.D22   AND
        D2.D22  =  "AAA"
        ;
```

このＧＣＳ問合せのグラフ表現を図7.9に示す。次にＧＣＳ問合せは、分散情報を用いて問合せ変形される。例えばＧＣＳ問合せ内の属性Ｄ１・Ｄ１２は、ＧＣＳリレーションＤ１の定義から、LCS属性の算術式Ｂ１＊８によって置き換えられる。このようなＧＣＳ属性の置き換えの後に、ＧＣＳリレーションの定義式が接合（conjunction）される。最終的に、ＧＣＳ問合せの条件部は次のようになる。

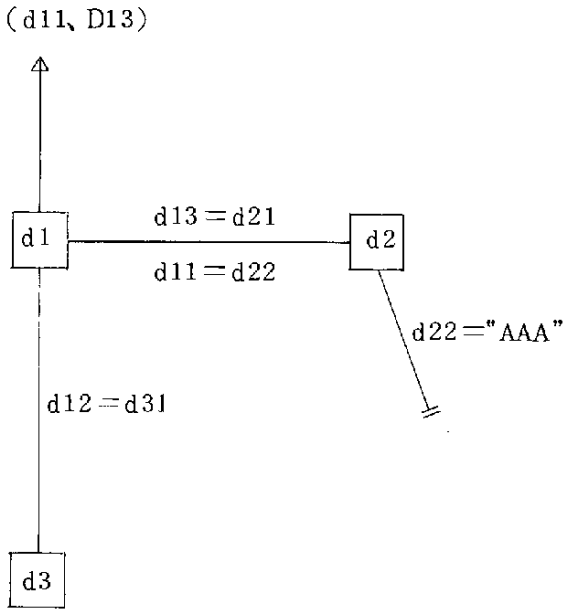


図7.9　ＧＣＳ問合せのグラフ表現

```
C.C1 = D.D1 AND F.F1 + 2 = E.E2 AND B.B1 * 8 = H.H2 AND H.H2 = "AAA" AND
 A.A1 = B.B1 AND B.B1 = C.C2 AND B.B2 = F.F2 AND A.A2 = "A" AND D.D1 GE
500 AND E.E1 = H.H2 AND H.H3 = G.G2
```

この様にて生成された問合せ、即ち全体ＬＣＳ問合せ（ＧＬＱ）の問合せグラフ（ＧＬＱＧ）を図7.10 に示す。

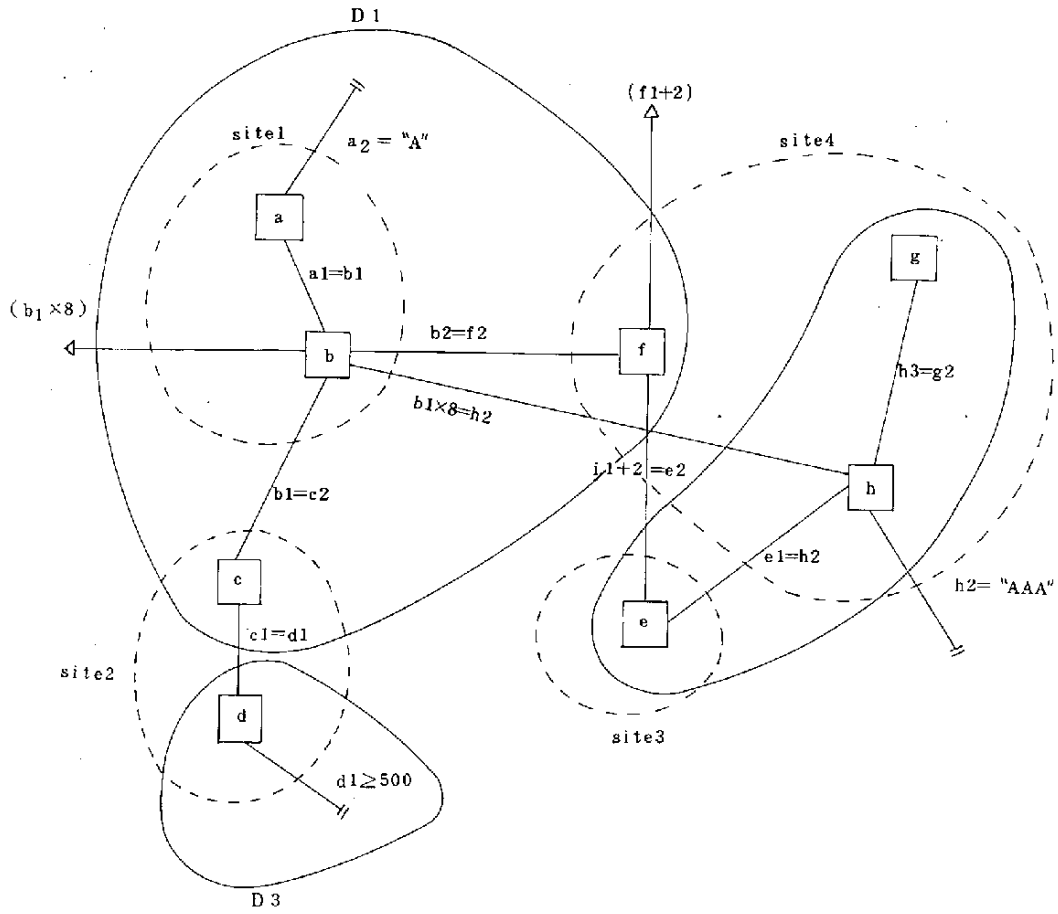


図7.10　全体ＬＣＳ問合せの問合せグラフ

7.6　初期局所問合せ処理（ＩＬＱＰ）

これまで述べてきた問合せ正規化（ＱＮ）と問合せ変形（ＱＭ）とによって全体概念スキーマ層の問合せ（ＧＣＳ問合せ）に対応するＬＣＳリレーションのみを参照する問合せ（即ちＧＬＱ）に変換された。このような問合せの表現の変換を行なった次の問題は、サイト間にまたがった問合せをいかに処理するかである。我々の基本戦略〔7.3を参照〕は先ず各サイトで独立に処理できる部分を処理してしまい、転送されるリレーションサイズを出来るだけ少なくかつ必要なものだけを転送させることである。

全体ＬＣＳ問合せ（ＧＬＱ）は、次の２つの部分に分けられる。

１）単一のサイトのみを参照している比較述語（comparison predicate）の論理式。

２）複数のサイトを参照する比較述語の論理式（この述語は結合述語である）。

前者は各サイトで独立に処理できるものである。これに対して、後者は、結合式がサイト間にまたがっているために、これを処理するためには、リレーションをサイト間で転送する必要がある。さて前者を後者に先だって処理させることによって後者における転送コストを減少させることができる。前者の処理を、我々は、初期局所問合せ処理（initial local query processing or ILQP）と呼ぶ。

初期局所問合せ処理（ＩＬＱＰ）は次の様な処理から成っている。

１）先ず問合せグラフ（query graph）〔ＴＡＫＩＭ７９ｂ、８０ａ〕〔5.3を参照〕をつくる。全体ＬＣＳ問合せ（ＧＬＱ）の問合せグラフ表現を全体ＬＣＳ問合せグラフ（global LCS query graph or ＧＬＱＧ）と呼ぶ。5.3節で指摘した様に、問合せグラフは、グラフ内のリンクによって表わされる論理式の論理積のみを表わせることを注意しておく。

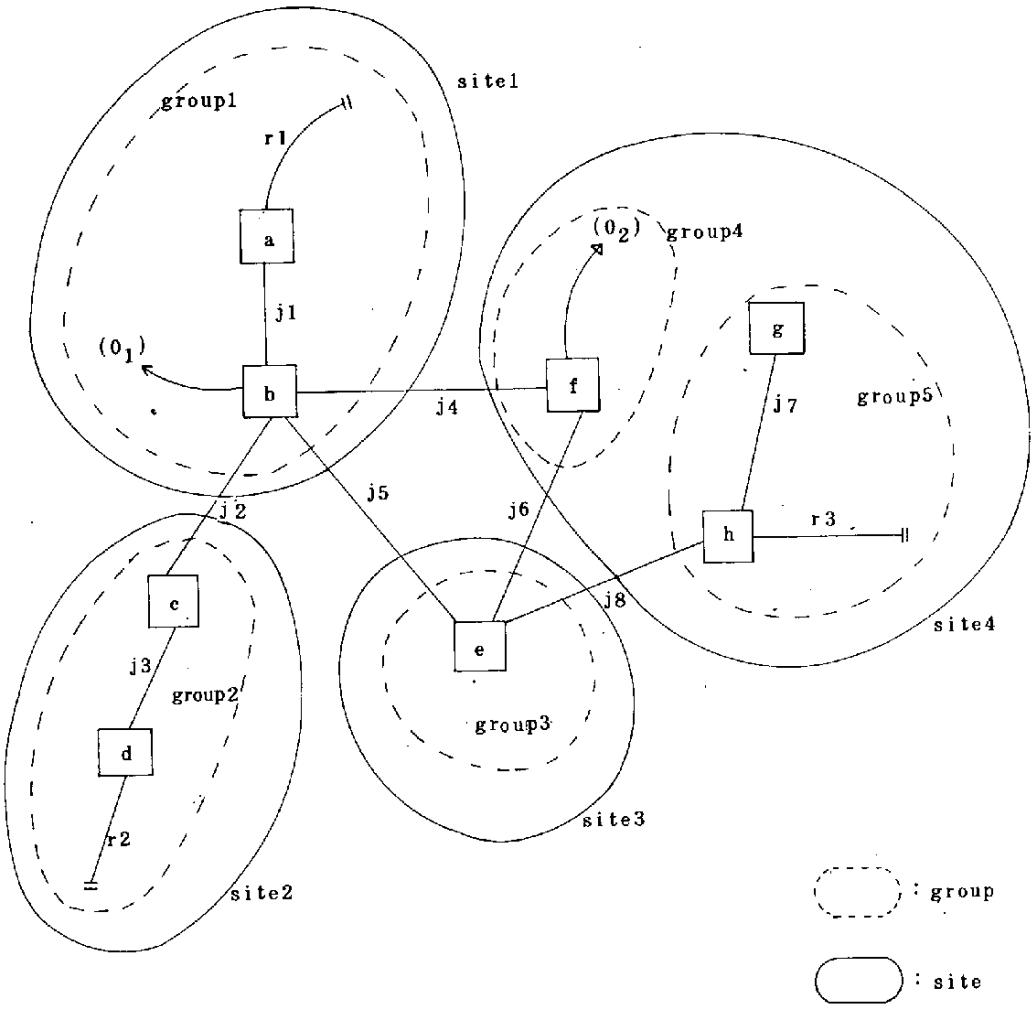
２）全体ＬＣＳ問合せグラフ（ＧＬＧＱ）内のノードを次の様にグループ化する。まず、ノードを各サイトごとにグループ化する。更に各グループ内のノードの中で、そのグループ内の結合リンクで互いに連結されたノードにグループ化する。最終的に、各グループは、同一サイト内にあり、かつそのサイト内の結合リンクによって連結されたノードから成り立つことになる。

３）各グループに対応したＬＣＳ問合せ（ＬＱ）に生成する。各ＬＣＳ問合せの目標リストは、全体ＬＣＳ問合せの目標リスト内に記された結果属性に加えて、対応するグループ内のノードと、他のグループ内のノードとの間の結合リンク内に示された結合属性とを結果属性として定義している。ＬＣＳ問合せ内の条件部は、このグループ内の結合リンクによって表わされる結合論理式と、制限リンクによって表わされる制限論理式との論理積（and）として表わされる。

４）３）で生成されたＬＣＳ問合せを対応するサイトへ発進する。

５）これらのＬＣＳ問合せによって、各グループには結果リレーションが生成されることになる。これらの結果リレーション間には、３）で述べたグループ間の結合リンクが存在する。制限リンクは、全てＬＣＳ問合せで処理されてしまうので、もうここでは存在しない。ＬＣＳ問合せを出したサイトからＡＣＫが来るのを待つ間、結果リレーションをノードとする問合せグラフをつくる。このグラフは、条件を表わすリンクとしては、結合リンクのみを含むので、結合問合せグラフ（join query graph or JQG）と呼ばれる。

例として図7.11（図7.9と同じである）のような全体ＬＣＳ問合せグラフ（ＧＬＱＧ）を考えてみよう。このサイトはｆ，ｇ，ｈの３つのノードをもっている。このうちｇとｈとは連結であるが、ｆはそうではない。よってｆのグループ（group４）と、ｇとｈとのグループ（ｇｒｏｕｐ５）とに分けられ、各々に次のようなＬＣＳ問合せが出される。



: a node representing a tuple variable α

: a k−th join−link

: a k−th restriction−link

: a k−th result−link

j1: a. a_1=b. b_1
j2: b. b_1=c. c_2
j3: c. c_1=d. d_1
j4: b. b_2=f. f_2
j5: b. b_1 *8=e. e_2
j6: f. f_1+ 2=e. e_2
j7: h. h_3=g. g_2
j8: e. e_1=h. h_2

r1: a. a_2= "A"
r2: b. $d_1 \geq 500$
r3: h. h_2= "AAA"

o_1: b. b_1*8
o_2: f. f_1+2

図7.11　ＧＬＱＧの例

```
group4   LQ4 : range (f) (FFF : 4) ;
                retrieve into R4(r41=f. f2, r42=f. f1+2) ;

group5   LQ5 : range (g. h) (GGG : 4, HHH : 4) ;
                retrieve into R5 (r51=h. h2)
                where g. g2 = h. h3 and h. h2 = "AAA" ;
```

LQ4において、結果属性r42は、問合せの結果属性であるとともにノードeとの結合属性である。r41は、ノードbとの結合属性である。LQ5の結果属性r51はノードeとの結合属性である。LQ5はさらに条件部に結合リンクj7と制限リンクr3とに各々対応する結合論理式(g.g2=h.h3)と制限論理式(h.h2="AAA")とを持っている。

同様にグループ1，2，3に対しても次の様なLCS問合せが生成される。

```
group1  LQ1 : range (a. b) (AAA : 1, BBB : 2) ;
               retrieve into R1 (r11 = b. b1 * 8, r12=b: b1, r13 = b.b2)
               where a. a1 = b. b1 and a. a2 = "A" ;
```

r11は全体LCS問合せ(GLQ)の結果属性とともに結合リンクj5を表わしている。

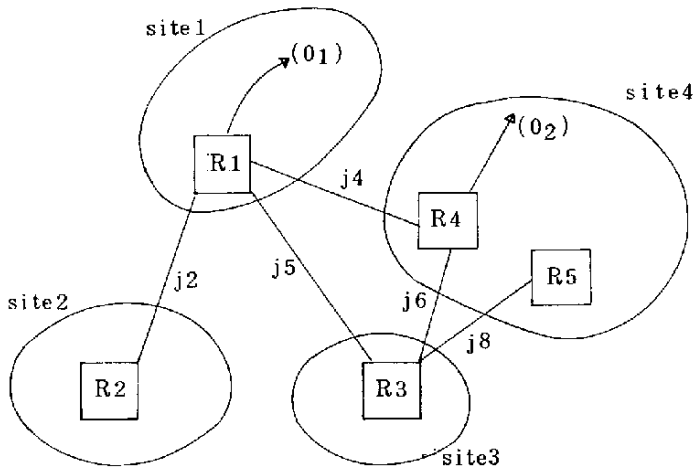
```
group2  LQ2 : range (c. d) (CCC : 2 , DDD : 2) ;
               retrieve into R2 (r21 = c. c2)
               where c. c1 = d. d1 and d.d1 ≥ 500 ;

group3  LQ3 : range (e) (EEE : 3) ;
               retrieve into R3(r31 = e. e1, r32=e. e2) ;
```

グループ3とグループ4とは、各々1つのノードeとfとをもっているだけで、他に連結したノードを持たない。よってこれらに対するLCS問合せは、他のサイトとの結合属性についての射影(projection)を行なうための問合せLQ3とLQ4とが各々出される。例えばLQ3のLCS属性e_1はサイト間結合リンクj8に関する結合属性であり、j5とj6についての結合属性である。この様に、各サイトに出されたLCS問合せは、サイト間処理に必要な属性のみから成るリレーションを結果として生成する。

この様にして各LCS問合せは、結果リレーションR1，R2，R3，R4，R5を生成する。

全体LCS問合せは、これらのリレーションに対する問合せに変形される。変形された問合せは、サイト間の結合のみを条件部に持つ。これらのグラフ表現は結合問合せグラフ(JQG)〔図7.12〕である。図7.7内の結合リンクjh，h=2，4，5，6，8と結果リンク0_1，0_2とは図7.6 内の結合リンクと結果リンクとに対応している。このJQGに対応した問合せはQUELによって次の様に

表わせる。



j 2 ： R 2 . r22 = R 1 . r11　　0 1 ： R 1 . r11

j 4 ： R 1 . r13 = R 4 . r41　　0 2 ： R 4 . r42

j 5 ： R 1 . r11 = R 3 . r32

j 6 ： R 4 . r42 = R 3 . r32

j 8 ： R 5 . r51 = R 3 . r31

図7.12　図8.6のＩＬＱ後のＪＱＧ

```
range (r1,r2,r3,r4)(R1:1,R2 2,R3:3,R4:4,R5:4) ;
retrieve into  R (r1, r11, r4, r42)
where          r2.r22=r1.r11   and
               r1.r11=r3.r32   and  r1.r13=r4.r41  and
               r5.r51=r3.r31   and  r4.r42=r3.r32  ;
```

初期局所問合せ処理（ＩＬＱＰ）は、リレーションの転送に必要な部分のみを結果リレーションとして生成する。このため、ＩＬＱＰによって不必要な部分の転送をさせないことにより、通信コストの低減をはかることができる。

1）ＬＣＳ問合せ（ＬＱ）

```
** LOCAL QUERY TO SITE 02 **
RANGE OF (C    ) IS (CCC);
RANGE OF (D    ) IS (DDD);

RETRIEVE INTO  NRELNAME01 ( C2 = C.C2 )
WHERE C.C1 = D.D1 AND D.D1 GE 500 ;


** LOCAL QUERY TO SITE 04 **
RANGE OF (F    ) IS (FFF);
RANGE OF (G    ) IS (GGG);
RANGE OF (H    ) IS (HHH);

RETRIEVE INTO  NRELNAME02 ( H2 = H.H2 )
WHERE H.H2 = "AAA" AND H.H3 = G.G2 ;

RETRIEVE INTO  NRELNAME03 ( F1 = F.F1 , F2 = F.F2 )
;


** LOCAL QUERY TO SITE 01 **
RANGE OF (A    ) IS (AAA);
RANGE OF (B    ) IS (BBB);

RETRIEVE INTO  NRELNAME04 ( B1 = B.B1 , B2 = B.B2 )
WHERE A.A1 = B.B1 AND A.A2 = "A" ;


** LOCAL QUERY TO SITE 03 **
RANGE OF (E    ) IS (EEE);

RETRIEVE INTO  NRELNAME05 ( E2 = E.E2 , E1 = E.E1 )
;
```

2）サイト間問合せ

```
**  INTER-SITE QUERY  **
RANGE OF (NV01) IS (NRELNAME01);
RANGE OF (NV02) IS (NRELNAME02);
RANGE OF (NV03) IS (NRELNAME03);
RANGE OF (NV04) IS (NRELNAME04);
RANGE OF (NV05) IS (NRELNAME05);

RETRIEVE INTO RESULT ( R1 = NV04.B1 * 8 , R3 = NV03.F1 + 2 )
WHERE NV03.F1 + 2 = NV05.E2 AND NV04.B1 * 8 = NV02.H2 AND NV04.B1 = NV01
.C2 AND NV04.B2 = NV03.F2 AND NV05.E1 = NV02.H2 ;
```

図 7.1 3　初期局所問合せ処理（ＩＬＱＰ）の例

7.7 転送スケジューリング（ＴＳ）

初期局所問合せ処理（ＩＬＱＰ）によって生成された結合問合せグラフ（ＪＱＧ）は、図7.12から解かる様に、各サイト単位で独自に処理できるところは全て処理し終った後の残されたサイト間処理を表わしている。サイト間の結合を処理するためには、どちらかのリレーションを他方へ転送せねばならない。このネットワークを通しての転送は、分散型データベースシステムの最も主要なオーバヘッドとなる。このために7.2で論じた様に、通信量を最少にし、応答時間を最少にしかつ問合せの処理に必要な情報（即ち分散情報）をなるべく小さくかつ静的にするように転送方式を決めねばならない。本節では、この転送方式について論じる。

7.1で述べたように、転送スケジューリング（Transmission Scheduling or TS）とは、ステージ（stage）の最適なシーケンスである。ステージ（stage）とは、次のような機能から成り立っている。

1）ソースリレーション（source relation）をソースサイト（source site）から目的サイト（destination site）へ転送すること。

2）目的サイトで、転送されてきたソースリレーションとそのサイトにある目的リレーションとの結合を行なうこと。

3）その結果を、目的リレーションとして格納すること。

ステージは、サイト間問合せの処理の基本単位である。

ここでr'をサイトiにあるソースリレーションとし、rをサイトjにある目的リレーションとする。$r'：i \rightarrow j$は、ソースリレーションrをソースサイトiからjへの転送を表わすものとする。$C(r'：i \rightarrow j)$は、その転送コストとする。$r'：i \rightarrow j：r$は、ソースリレーションをr'、目的リレーションをrとした時のステージを表わすとする。又、$|s|$はリレーションsのサイズを表わす（ここで$s=r$ or r'）。以降この記法を用いて、我々の転送スケジューリングアルゴリズム（ＴＳＡ）について論じる。

我々のＴＳＡは7.3で論じたように次の特徴をもつ。

1）転送方法は、実行しながら動的に決定される。統合全体データベースプロセッサ（CGDP）は、現在のステージを決定するために、これまでのステージの実行結果についての情報を用いる。

2）問合せ分割が必要とする情報をなるべく小さくかつ静的にすることを第1の目標とする。

3）ある閾値（threshold value）よりも転送コストが小さいリレーションはなるべく並列して転送する。

このような戦術を実行するために統合ＧＤＰ（ＣＧＤＰ）は、分散情報（ＤＩ）に加えて次の2つのディレクトリを管理する。

1）問合せ処理情報（query processing information or QPI）

2）論理転送コスト表（logical transmission cost table or LCT）

問合せ処理情報（ＱＰＩ）は、次のステージを決定するために、これまでのステージの実行結果についての情報を格納するために用いられる。これらの情報は、ステージのＡＣＫにのせて統合ＧＤＰ（ＣＧＤＰ）に、目的サイトから送られてきたものである。問合せ処理情報（ＱＰＩ）は、次の２つのリレーションから成り立っている。

ＱＰＩ／ＲＥＬ　（site-no , rel-no , cardinality , width）

ＱＰＩ／ＡＴＴ　（site-no , rel-no , att-no , width）

ＱＰＩ／ＲＥＬは、ステージで生成された目的リレーションのパフォーマンス情報、即ちリレーションの属性構成についての情報を格納している。

論理転送コストテーブル（ＬＣＴ）は、任意のサイト間の転送コストを示している。ＬＣＴ内の各エントリLC_{ij}は、サイトiからサイトjへ１つのパケットを転送するのに要する遅延を示している。我々は簡単のために、サイトiからjへのパスのうち最小のホップ数を持つパスの転送コストのみをLC_{ij}とする。LC_{ij}は、このホップ数に比例する。論理転送コストテーブル（ＬＣＴ）を用いて、ソーフリレーションr'をサイトiからjへ転送するためのコストは

$$C(r' : i \rightarrow j) = |r'| \cdot LC_{ij}$$

とすることができる。

更に、統合全体データベースプロセッサは、ユーザの入力したＧＣＳ問合せから生成された結合問合せグラフ（ＪＱＧ）を保持している。

我々の転送スケジューリングアルゴリズム（ＴＳＡ）は、次のような基本操作の繰り返しから成っている。

1）次のステージの決定

2）結合問合せグラフ（ＪＱＧ）の縮退

3）問合せ処理情報（ＱＰＩ）の更新

まず、結合問合せグラフ（ＪＱＧ）の縮退について考えよう。又、転送スケジューラ（ＴＳ）が開始される時、結合問合せグラフ（ＪＱＧ）内の全てのノードはＦＲＥＥとマークされているとする。あるステージ$r' : i \rightarrow j : r$が次のステージとして選ばれたとする。この時統合全体データベースプロセッサ（ＣＧＤＰ）はＪＱＧをアクセスして次のように変化させる。

1）ノードr'　を　ＳＯＵＲＣＥとマークする。

2）ノードr　を　ＤＥＳＴとマークする。

3）r'から出ている結合リンクを、rにも付加する。r'と結合リンク　$r'r''$によって結合されたノードをr''　としよう。r''　がrでないとする。この時、rとr''　との間の結合リンク$jr'r''$に対応するリンクjrr''をつくりＪＱＧへ加える。もし既にrとr''　の間に結合リンクがあれば、これにjrr''を接合する。これをrでない全てのr''について試みる。

4）ＱＰＩ／ＡＴＴをアクセスして、ステージ$r' : i \rightarrow j : r$　の結果生成されるリレーションrのスキームに合うように更新する。rは、以前のスキームに加えて、3)でrに付加された結合リンクに関する結合属性が加えられる。

5）もしr'　が結果リンクをもてば、対応する結果リンクをrにも付加する。

例として、図7.12の結合問合せグラフ（ＪＱＧ）を考えてみよう。このＪＱＧにおいて、次のステージとしてＲ４：4→3：Ｒ３が選ばれたとしよう。するとＲ４に関する結合リンク j_4 と j_6 のうち、3)の条件を満足するものは j_4 である。よって j_4 を j_5 に付加し j_9 とする。又、Ｒ４の結果リンク O_2 も、Ｒ３に対応する o_2' として付加される。はじめ、Ｒ３のスキームは R３（r_{31} , r_{32} ）であり、Ｒ４は R４（r_{41} , r_{42} ）である。Ｒ４とＲ３との結合は

range　（r_3 , r_4 ）（Ｒ３：３，Ｒ４：３）；

retrieve into　Ｒ３（$r_{31} = r_3 \cdot r_{31}$, $r_{32} = r_3 \cdot r_{32}$, $r_{33} = r_4 \cdot r_{41}$ ）

where　$r_4 \cdot r_{42} = r_3 \cdot r_{32}$ ；

と書ける。新しいＲ３の属性 r_{32} は、Ｒ１との結合属性とともに、Ｒ４の結果属性 r_{42} に対応する結果属性となる。r_{33} は、j_4 ：R1．r_{18}＝R4．r_{41} に対応するＲ１と新R3 との間のＲ３の結合リンクである。R4：4→3：R3 の結果として縮退された結合問合せグラフ（ JQG）は図7.14のようになる。

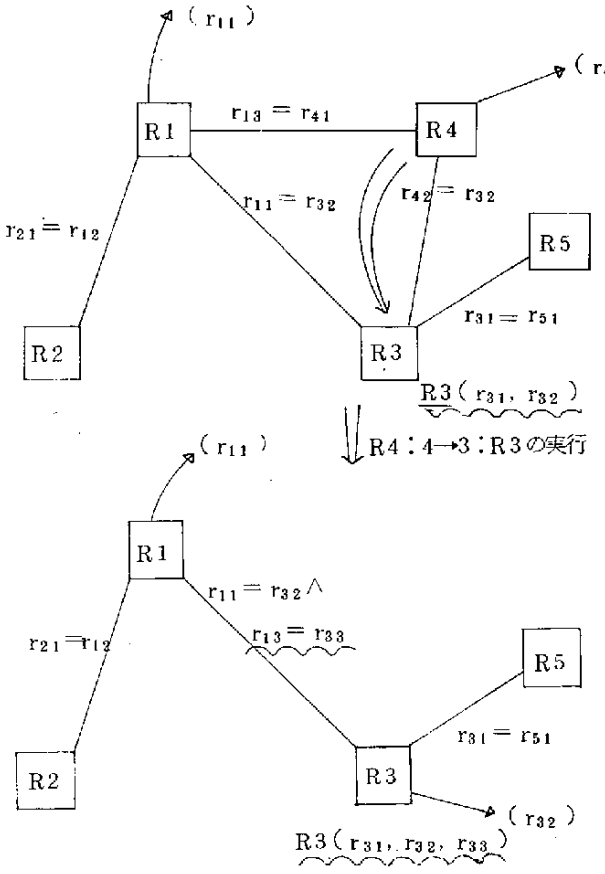


図7.14　Ｒ４：4→3：Ｒ３後のＪＱＧ

統合全体データベースプロセッサ（CGDP）は、ステージ r'：i→j：r が決まるとサイト i の局所データベースプロセッサ（source LDP or SLDP）、即ち、LDPi、へ転送コマンド（T）を送る。

Ｔコマンドは、次の様な形式をもっている

T（stage-no，s-site-no，s-rel-no，
d-site-no，d-rel-no）

stage-no は、ある全体ＬＣＳ問合せ（GLQ）の処理のために生成されたステージの番号である。他は、ステージ s-rel-no(r')：s-site-no(i)→d-site-no(j)：d-rel-no(r) を表わしている。これを受けたソースＬＤＰ（ＳＬＤＰ）は、対応する目的LDP（destination LDP or DLDP）、即ち、LDPj，からＷＳＡ（WS allocated）コマンドが到着するのを待つ。目的ＬＤＰ（ＤＬＤＰ）からＷＳＡが到着すれば、ソースリレーション r' をこの目的ＬＤＰ（ＤＬＤＰ）へ転送する。転送終了後目的ＬＤＰからACKを受けとれば、ソースＬＤＰのＷＳ内のソースリレーション r' を消去する。

ＣＧＤＰは転送（T）コマンドと同時に結合コマンド（J）を目的ＬＤＰＰに送る。Ｊコマンドは次の様な形式をもっている。

J (stage-no,s-site-no,s-rel-no,d-site-no,d-rel-no,target-list,qual,s-rel-size)

このJコマンドは次のことを意味している。

retrieve into d-rel-no (target-list)

where qual ;

即ち、ＳＬＤＰから転送されてきたr'と、ＤＬＤＰ内のrとの結合をqualとtarget-list によって行ないその結果リレーション名をd-rel-no(i.e.r)としてＷＳに格納する。s-rel-size は、ソースリレーションr' のサイズである。これは統合全体データベースプロセッサ内の問合せ処理情報（ＱＰＩ）に格納されている。まず目的ＬＤＰは、転送(T)コマンドを受けとるとs-rel-size によって示された大きさのリレーションを格納するためのエリアを作業領域（ＷＳ）内に確保する。確保できたならば、ソースＬＤＰ（即ちLDPi）へＷＳＡを送り、ソースリレーションr'の転送を待つ。r'を全て受信したならばソースＬＤＰへＡＣＫを返す。次に目的ＬＤＰは、目的リレーションrとソースリレーションr'を結合し、結果をrとして格納してＡＣＫを統合ＧＤＰ（ＣＧＤＰ）へかえす。このＡＣＫには、生成された目的リレーションrのカーディナリティについての情報をのせてＣＧＤＰへ送る。このＡＣＫは次のような形式を持っている。

ACK(stage-no,d-site-no,d-rel-no,cadinality)

統合全体データベースプロセッサ（ＣＧＤＰ）があるステージr'：$i \rightarrow j$：rに対するＡＣＫを目的ＬＤＰ,i.e.ＬＤＰj，から受けとったならばＣＧＤＰは結合問合せグラフ（ＪＱＧ）の縮退と問合せ処理情報（ＱＰＩ）の更新を試みる。これは、次の様になされる。

1) ＪＱＧからソースリレーションに対応するノードr'を除去する。

2) このr'と結合している結合リンクを、ＪＱＧから除去する。

3) QPI/REL及びQPI/ATTリレーションから、r'に関する組を全て消去する。

4) QPI/REL内のリレーションrについてのカーディナリティ等のパフォーマンス情報を、今受信したＡＣＫ内の情報によって置き換える。

次に統合ＧＤＰ（ＣＧＤＰ）は、次に実行されるべきステージの決定を試みる。次の条件を満足するステージr'：$i \rightarrow j$：rが、次に実行されるべきものとして選ばれる。

1) 結合問合せグラフ（ＪＱＧ）内のソースノードr'は、FREEとマークされている。

2) 目的ノードrは、r'と結合リンクによって結合されている。

3) 目的ノードrは、FREE又はDESTとマークされている。

4) r'をサイトiからjへ転送するためのコストC(r'：$i \rightarrow j$)は、JQG内において最少であるとともに、ある閾値（ＴＨＶ）よりも小さい。

第１の条件は、ソースノード（リレーション）r'は、処理中でないこと、即ち、他のステージの目的ノード（リレーション）として、結合演算中でないこと、又は、ソースノードとして転送中でないことを保障している。第２の条件は、rとr'間に結合リンクがあることを確かめている。第３の条件は、目的ノードrは、他のステージの目的ノードとして処理中であってもよいことを示している。即ち、rが他のステージr''：k$\rightarrow j$：rの目的ノードとして、r'' の転送を受けrと結合を行なっていたと

しても、r'をソースリレーションとしてサイトjへ転送できる。このことによって、複数のステージの転送を並行して行なえることになる。

1つのステージにおいて、その転送コストが最も主要なコストとなるので、この転送の並行化は、良い応答時間をもたらすと考える。

第4の条件は、転送可能なリレーションが比較的大きなものばかりで、今実行中のものが比較的小さいものがある時、転送可能な大きなリレーションを転送しないで、小さいものの処理が終了するのを待つことを示している。この転送するかしないかは、統合全体データベースプロセッサ（CGDP）が保持しているある閾値（THV）を用いて行なう。この決定時に転送コストC（r'：$i \rightarrow j$）が全ての可能な転送のなかで最少であるとともに、この値が閾値（THV）より小さいならば、r'：$i \rightarrow j$：rは次のステージとして選ばれる。

もし閾値より小さいものがなくかつ全てのノードがFREEとマークされている時は、閾値（THV）の値を更新する。現在は、可能な全転送のコストの平均をとることとしている。

閾値よりも小さいものがなくかつ全てのノードがＦＲＥＥではない時は、統合ＧＤＰ（ＣＧＤＰ）はこれらの未終了のステージからのＡＣＫを待つ。ＡＣＫを受けとるたびに、これらの４つの条件に合うステージを、結合問合せグラフ（ＪＱＧ）のなかから見つけることを試みる。

閾値（ＴＨＶ）の値は、サイト間問合せ処理の制御バルブの役目を持つ、閾値の値が大きければ、転送のパラレリズムは高まり、小さくすれば逆に低くなる。パラレリズムの高まりは、応答時間を短かくするが逆にかなり大きなリレーションも転送してしまうので、転送コストの増大をもたらす。パラレリズムを小さくすれば応答時間は長くなるが、その都度最少の転送コストのものを選び、それを転送できるので全転送コストを押さえることができる。我々は閾値の値のセットをシミュレーションを通して決定していきたいと考えている。

図7.15は統合ＧＤＰ（ＣＧＤＰ）における転送スケジューリング（ＴＳ）アルゴリズムを示し、図7.17は局所データベースプロセッサ（ＬＤＰ）に対するＴＳアルゴリズムを示している。

統合全体データベースプロセッサ（CGDP）におけるＴＳアルゴリズム

0) r'：$i \rightarrow j$：rは、次のようなステージとする。

r'はサイトi（LDPi）にあるソースリレーションを表わすノード（ソースノード）とし、rをサイトj（LDPj）にある目的リレーションを表わすノード（目的ノード）とする。これは、r'をサイトiからjへ転送し、jでr'とrとを結合し、結果をrとしてjのＷＳに格納するステージである。

：$\rightarrow j$：rは、サイトjに対する初期局所問合せ処理（ＩＬＱＰ）とする。即ち、サイトjでＩＬＱＰの結果としてリレーションrが生成されることを示している。

1)〔ＩＬＱＰ〕

i) 全体ＬＣＳ問合せ（ＧＬＱ）から、対応する問合せグラフＧＬＱＧをつくる。

図7－15 (1) ＣＧＤＰに対するＴＳアルゴリズム

ii) 全体ＬＣＳ問合せグラフ（ＧＬＱＧ）内のノードを次の様にグループ化する。

a) グループ内のノードは全て同一サイト内にある。

b) かつ、グループ内のノードは、そのグループ内の結合リンクによって連結である。

iii) 各グループに対応するＬＣＳ問合せ（ＬＱ）をつくる。

iv) これらのＬＣＳ問合せを対応するサイトへ送出する。

v) 初期局所問合せ処理（ＩＬＱＰ）によって生じる結果リレーションをノードとし、サイト間結合を結合リンクとする結合問合せグラフ（ＪＱＧ）をつくる。

vi) 問合せ処理情報（ＱＰＩ）をＪＱＧに基づいて初期化する。（ＪＱＧにもとづいて、各ノードのスキームをＱＰＩに格納する）

vii) ＩＬＱＰに対するＡＣＫを待つ。

2) 〔ＡＣＫの受信〕

受信したＡＣＫがＩＬＱＰの局所問合せに対するものであれば、4)へ

3) 〔ＪＱＧの縮退〕

i) r'を結合問合せグラフ（ＪＱＧ）から除去する。

ii) r'に連結した結合リンクを除去する。

iii) r'に関する組をＱＰＩ／ＲＥＬ、ＱＰＩ／ＡＴＴから除去する。

4) 〔ＱＰＩの更新〕

i) このＡＣＫが、rを目的ノードとするステージのなかで最も最近のものでなければ、2)へ

ii) ＡＣＫによって運ばれてきた情報によって、QPI/REL内のrに関する組を更新する。

iii) ＪＱＧ内のrノードをFREEとマークする。

5) 〔最終結果チェック〕

縮退された結合問合せグラフ（ＪＱＧ）が１つのノードだけとなれば、これが求める最終結果であるので、これを統合ＧＤＰ（ＣＧＤＰ）へ転送させて、ＴＳを終了する。

6) 〔次のステージの決定〕

次の条件を満足するようなr'：$i \rightarrow j$：ｒを選ぶ

a) ｒ′はＦＲＥＥとマークされている。

b) ｒはｒ′と結合リンクで結ばれている。

c) r'はFREE又はDESTとマークされている。

d) $C(r': i \rightarrow j)$をもとめる。これは$C(r': i \rightarrow j) = |r'| * LC_{ij}$としてＱＰＩ、ＬＣＴを用いてもとめられる。$C(r': i \rightarrow j)$の値がＪＱＧ内で最少でかつＴＨＶ値より小さい。

7) 〔ステージの生成と実行〕

i) このようなステージ$r': i \rightarrow j$：rがみつからなければ、8)へ。

ii) ステージ$r': i \rightarrow j$：rをもとにして、ＴコマンドをLDPiへ、ＪをLDPjへ送る。

iii) r'をSOURCEとマークし、rをDESTとマークする。

図7.15(2) ＣＧＤＰに対するＴＳアルゴリズム

iv) r''をr'のＪＱＧ内で隣接するノードとする。全てのr''（ただし$r'' \neq r$）について、r'とr''間の結合リンクに対応する結合リンクをつくり、ノードrにセットする。

8)〔6)を満足するステージがみつからない〕

i) ＪＱＧ内の全てのノードがＦＲＥＥであれば、ＴＨＶを更新する。新しいＴＨＶは、可能な全ての転送の平均コストとする。　6)へ。

ii) 全てがＦＲＥＥでなければ、　2)へ。

図7.15(3)　ＣＧＤＰに対するＴＳアルゴリズム

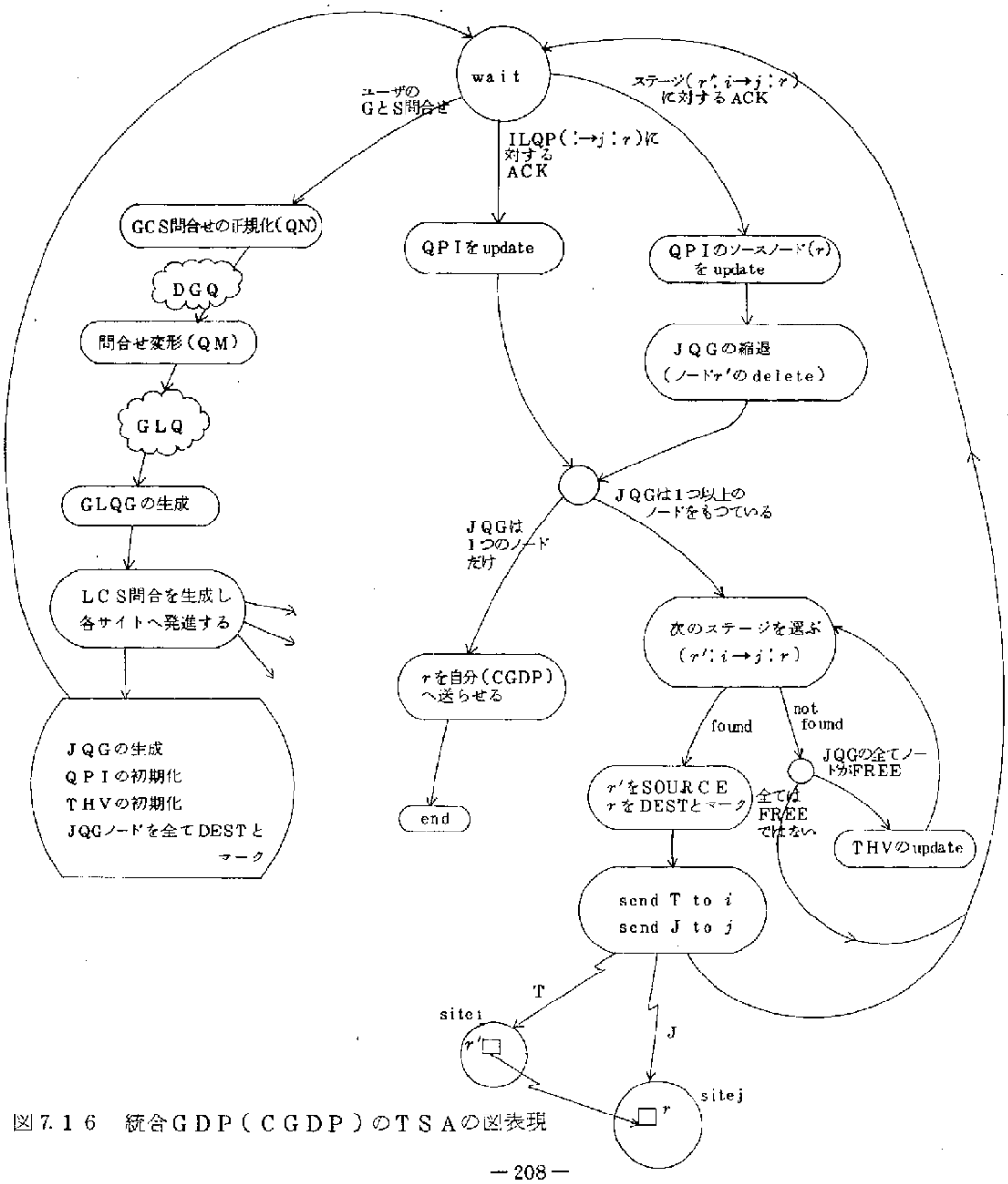


図7.16　統合ＧＤＰ（ＣＧＤＰ）のＴＳＡの図表現

局所データベースプロセッサ（ＬＤＰ）に対するＴＳアルゴリズム

ＴＲＡＮＳアルゴリズム

0) このＬＤＰ、i。e．ＬＤＰi、はソース局所データベースプロセッサ（source ＬＤＰ）である。

1) 転送(T)コマンドi。e．Ｔ（ｋ，i，r'，j，ｒ）、を受けとったならば、目的ＬＤＰ、i. e. ＬＤＰjからのＷＳＡを待つ。

2) ＷＳＡを受けとれば、ソースリレーションr'をそのスキームとともに、LDPjへ送る。

3) r'を全て転送後、LDPjからＡＣＫを受けとれば、LDPjのＷＳからr'を除去する。

ＲＥＣアルゴリズム

0) このLDP，i・e・LDPj，は目的局所データベースプロセッサ（destination ＬＤＰ）である。

1) ＪＯＩＮから指示の指示によって、ＷＳ内にr'を受信するための領域を確保する。

2) もし確保できれば、ＷＳＡをLDPiへ送り、r'の到着を待つ。

3) r'を受信しながら、rとr'の結合属性についてソートする。

4) r'を受信し終えたら、ＡＣＫをLDPiへ返す。
ＪＯＩＮへも受信の完了を知らせる。

ＪＯＩＮアルゴリズム

0) このLDP，i. e。ＬＤＰj，は 目的ＬＤＰである。

1) 結合(J)コマンドをＣＧＤＰから受けとったならば、ＲＥＣへそのことを知らせる。

2) ＲＥＣからのr'の受信完了通知を待つ間、rをr'との結合属性についてソートする。

3) RECからr'受信完了通知が届いたならば、rとr'との結合を行なう。rとr'は既にソートされているので、マージ結合（merge-join）〔ＳＥＬＩＰ７９〕によって結合を行なう。

4) rとr'の結合結果をrとしてＷＳに格納する。

5) 新しく出来たrについての統計情報（i. e. cardinality）を求め、これをＡＣＫにのせる。

6) ＡＣＫを統合ＧＤＰ（CGDP）へ送る。

ＱＴアルゴリズム

0) ＩＬＱＰを行う。

1) 統合GDP（CGDP）からＩＬＱＰによって生成されたＬＣＳ問合せを受信すると、まずＱＵＥＬで書かれたＬＣＳ問合せを目標データベースシステムのＤＭＬのシーケンスへ変換する。

2) このシーケンスを目標データベースシステムで実行させる。

3) 実行結果をリレーションとして作業領域（ＷＳ）に格納する。

4) ＡＣＫに結果リレーションの統計情報をのせて統合ＧＤＰ（ＣＧＤＰ）に返す。

図7.17　ＬＤＰに対するＴＳアルゴリズム

7.8　転送スケジューリング（ＴＳ）の例

例として図7.12について考えよう。まず各サイトでの初期局所問合せ処理（ILQP）は終了し、全てのノードについてのパフォーマンス情報は統合全体データベースプロセッサ（CGDP）の問合せ処理情報（QPI）内に集められているとする。さらに結合問合せグラフ（JQG）内の全てのノード（R_1，R_2，R_3，R_4，R_5）はFREEとマークされているとしよう。この時統合ＧＤＰ（CGDP）の保持するＱＰＩの２つのリレーションを図7.18に示す。さらにこの図では各リレーションのサイズ（＝cardinality ＊ width）も示してある。問合せ処理情報（ＱＰＩ）の保持するもう１つのディレクトリ，論理転送コストテーブル（ＬＣＴ）を図7.19に示す。

論理転送コストテーブル（ＬＣＴ）の各エントリLC_{ij}は、簡単のためにサイトiとjとの間の最短パス上のホップ数としてある。

これらの情報を用いて、ステージを決める。ここでSTi をi番目に決定されたステージとし、ACKiをSTiに対するＡＣＫとする。さらに閾値ＴＨＶの値は５００とする。まずST_1の決定について考えよう。各結合リンクj_k(k=2,4,5,6,8）について可能な転送コストをまとめると次の様になる。

j_2 ： C(R1：1→2)=C(R2：2→1)=500×2=1000

j_4 ： C(R4：4→1)=100×5 =500　　∵ |R4|≦|R_1|

j_5 ： C(R3：3→1)=200×3 =600　　∵ |R3|<|R_1|

＊ j_6 ： C(R4：4→3)=100×1 =100<500　∵ |R4|<|R_3|

j_8 ： C(R3：3→4)=200×1 =200　　∵ |R3|<|R_5|

QPI/REL(site-no rel-no cardinality width) size

1	R1	20	25	500
2	R2	10	50	500
3	R3	20	10	200
4	R4	4	25	100
4	R5	30	10	300

QPI/ATT(site-no rel-no att-no width)

1	R1	r_{11}	5
1	R1	r_{12}	10
1	R1	r_{13}	10
2	R2	r_{21}	50
3	R3	r_{31}	2
3	R3	r_{32}	8
4	R4	r_{41}	5
4	R4	r_{42}	20
4	R5	r_{51}	10

図7.18　ＩＬＱＰ後のＱＰＩリレーション

i \ j	1	2	3	4
1		2	3	5
2	2		2	2
3	3	2		1
4	5	2	1	

i, j ＝site-no

図7.19　ＬＣＴ

これから解かる様にR4：4→3：R3がST_1　として選ばれる。理由は、次の様である。

1）R4とR3はFREEとマークされている。

2）C（R4：4→3）はJQG内で最少である。

3）C（R4：4→3）＜THV（＝500）

よって、統合GDP（CGDP）は、次のような転送（T）コマンドと結合（J）コマンドを各々LDP_4，とLDP_3 に送る。

T（1，4，R4，3，R3）

J（1，4，R4，3，R3，

①（r_{31}＝R3．r_{31}，r_{32}＝R3．r_{32}，r_{33}＝R4．r_{42}）、

②（R4．r_{42}＝R3．r_{32}），③100　　）

①は目標リストを、②を条件式を、③はR4のサイズを表わしている。

次にJQGの変形をする。R4をSOURCEと，R3をDESTとマークする。R4に関する結合リンクをR3に付加する。結果として結合問合せグラフ（JQG）は図7.20 a）のようになる。

次にST_2の決定を試みる。ST_1 でみたように、j_8に沿った転送（R5：4→3）がそのコストは最少でかつ閾値（THV）よりも小さい。このためST_2 は、R5：4→3：R3となる。R3は、まだST_1の目的ノードとして処理中であるが、R5のサイト3への転送を開始する。この転送はST_1の転送とオーバラップされる。この結果結合問合せグラフ（JQG）は図7.20のb）のようになる。

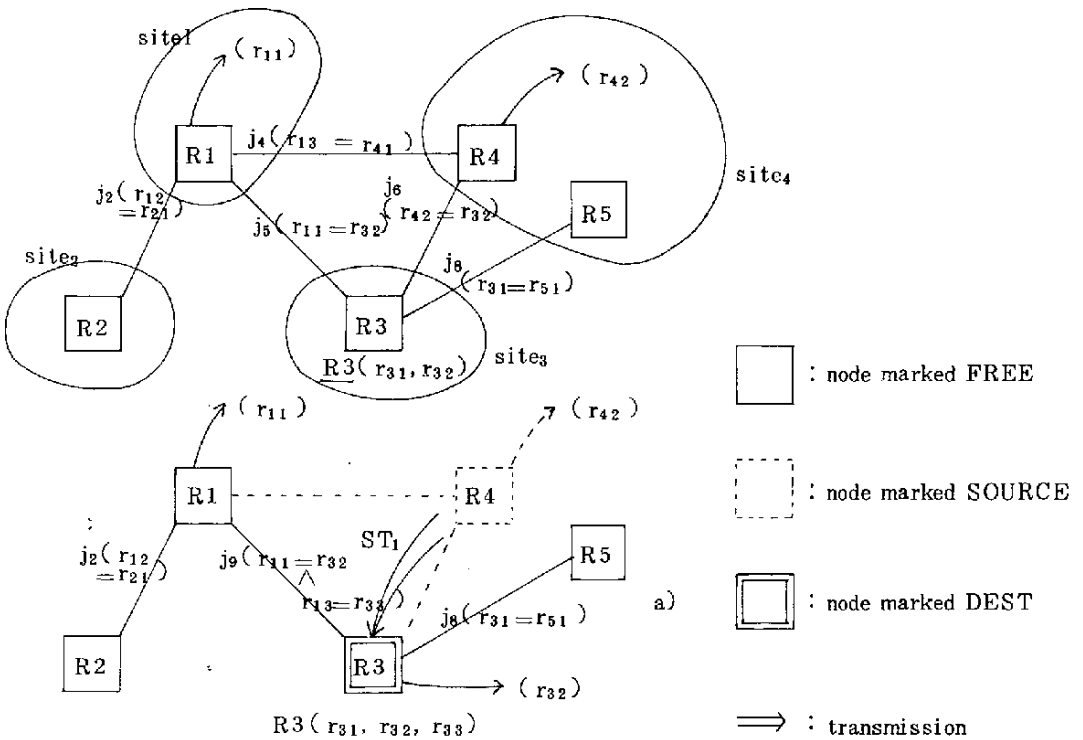
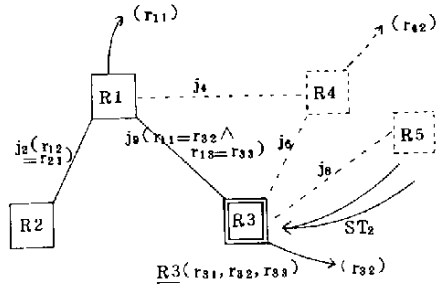
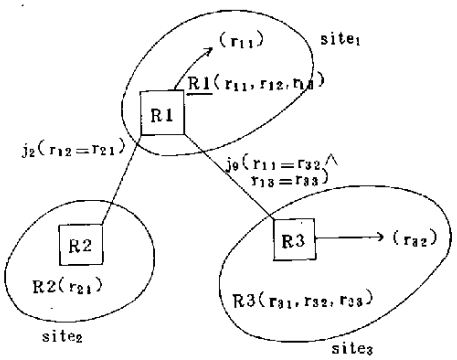


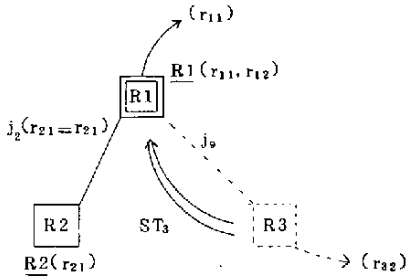
図7.20　TSの例（1）



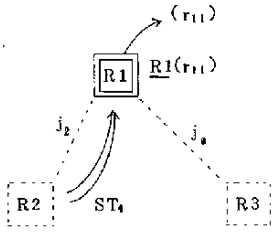
b)



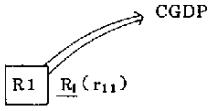
c) ACK_3, ACK_2 が返ってきた後のJQG



d)



e)



f)

図7.20　TSの例 (2)

QPI/REL	(site-no	rel-no	cardinality	width)	size
	1	R1	20	25	500
	2	R2	20	50	500
	3	R3	12	15	180

QPI/ATT	(site-no	rel-no	att-no	width)
	1	R1	r_{11}	5
	1	R1	r_{12}	10
	1	R1	r_{13}	10
	2	R2	r_{21}	50
	3	R3	r_{31}	2
	3	R3	r_{32}	8
	3	R3	$r_{33}(=r_{41})$	5

図7.21　図7.20c)に対応するQPIリレーション

続いてST_3の決定を試みる。FREEノードはR1とR2のみである。よって可能な転送は結合リンクj_2とj_9に沿ったものだけである。

j_2 : C(R1:1→2) = C(R2:2→1) = 500×2 = 1000 > 500

j_9 : C(R1:1→3) = 500 × 3 = 1500 > 500

この様に可能な転送は、全てそのコストが閾値(THV)(=500)よりも大きくなっている。ノードR3はまだDESTとマークされているのでST_1, ST_2 からのACKを待つ。

ST_1 からのACK(即ちACK_1)が統合GDP(CGDP)に届いたとする。ノードR4と結合リンクj_4とj_6とを結合問合せグラフ(JQG)から消去する。続いて問合せ処理情報(QPI)の2つのリレーションからR4についての全ての情報を消去する。R3は、まだDESTとマークされているのでさらにACKを待つ。ACK_2 がCGDPに到着すると、R5とj_8がJQGから除かれ、QPIからR5の情報が除かれる。ACK_2 は、R3に対するステージのなかで最新のステージ、即ちST_2、に対するものであるので、R3をFREEとする。さらにACK_2内の情報を用いてQPI/RELのR3についての情報を更新する。R3のサイズは200であったとしよう。QPIは図7.21のように、JQGは図7.20のc)のようになる。

図7.20c)の結合問合せグラフ(JQG)において可能な転送は次のようである。

j_2 : C(R1:1→2) = C(R2:2→1) = 500×2 = 1000>500

j_9 : C(R3:3→1) = 180 × 3 = 540 > 500

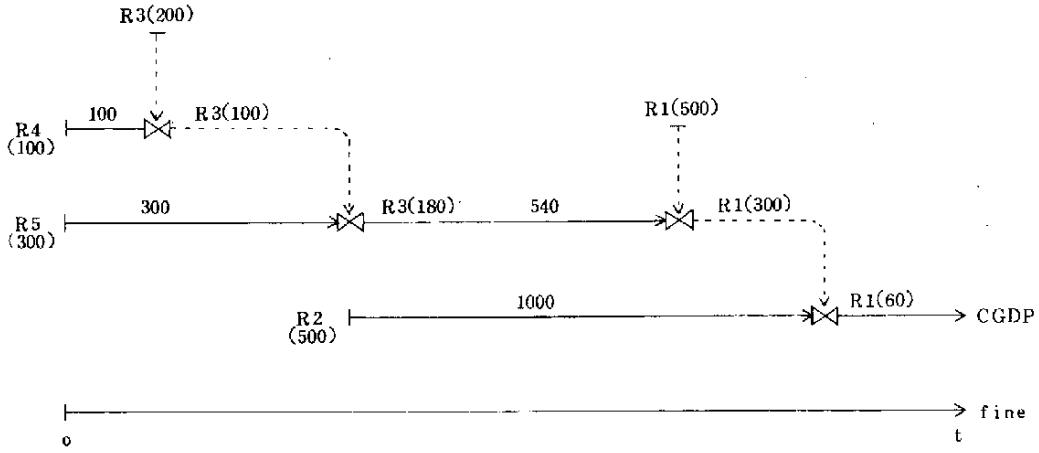
JQG内の全てのノードR1, R2, R3はFREEとマークされているのでTHVは次のようにリセット

される。　　THV = (1500(=C(R1:1→3) + 1000 + 1000 + 540)/4 = 1010

よってR3:3→1:R1がST_3として選ばれる〔図7.20d)〕。

C(R2:2→1)=1000<1010であるので、ST_4としてR2:2→1:R1が決定される〔図7.20e)〕。ST_3とST_4も、ST_1とST_2と同様に、互いの転送はオーバラップされる。

ACK_3とACK_4がCGDPにかえってきて、ST_4後のR1のサイズは50であったとしよう。最終的に図7.20f)のようにJQGは1ノードグラフとなる。このノードR1は最終結果となりCGDPへ転送される。



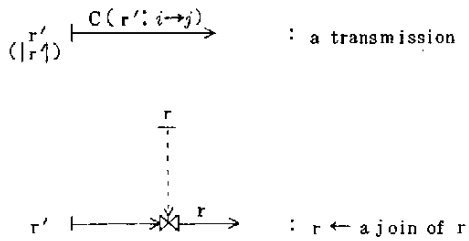


図7.22　図7.20のまとめ

この例を時間軸をいれてまとめると図7.22のようになる。この図からわかる様にR4とR5の転送はパラレルに行なわれる。又 R3とR2 の転送もパラレルに行なわれる。最終的なR1がサイト1で生成されるまでの全処理時間(tp)と応答時間(tr)は次の様になる。

tp = 100 + 300 + 540 + 1000 = 1940

tr = 300 + 1000 = 1300

我々の転送スケジューリングを用いずに、全てのリレーションをサイト１に集めて結合をとる場合について考えてみよう。

$$C(R1:1\rightarrow1) = 0$$
$$C(R2:2\rightarrow1) = |R_2| * LC_{21} = 500 \times 2 = 1000$$
$$C(R3:3\rightarrow1) = |R_3| * LC_{31} = 200 \times 3 = 600$$
$$C(R4:4\rightarrow1) = |R_4| * LC_{41} = 100 \times 5 = 500$$
$$C(R5:4\rightarrow1) = |R_5| * LC_{41} = 300 \times 5 = 1500$$

よって、全処理時間は

$$tp = 1000+600+500+1500=3600$$ となり

応答時間は $tr = 1500$ となる。我々の転送スケジューリングアルゴリズムは、これよりも優れていることがわかる。

7.9 ＬＤＰとＧＤＰアーキテクチャ

図7.23は全体データベースプロセッサ（ＧＤＰ）と局所データベースプロセッサ（ＬＤＰ）のアーキテクチャーを示している。統合ＧＤＰ（ＣＧＤＰ）は、問合せ正規化（ＱＮ）、問合せ変形（ＱＭ）、初期局所問合せ処理（ＩＬＱＰ）、転送スケジューリング（ＴＳ）の４つのモジュールから成っている。ユーザのＧＣＳ問合せを受けとるとＱＮ、ＱＭによって、全体ＬＣＳ問合せ（ＧＬＱ）を生成する。ＧＬＱは、ＧＣＳ問合せを対応するＬＣＳリレーションで表わしたものである。ＩＬＱＰは、サイトで独立に処理できるＬＣＳ問合せを生成し、対応するサイトへこれを送信する。さらに結合問合せグラフ（ＪＱＧ）を生成する。ＴＳはこのＪＱＧ、問合せ処理情報（ＱＰＩ）と論理転送コストテーブル（ＬＣＴ）と用いてステージを動的に決定していく。

ＬＤＰは問合せ変換（ＱＴ）と作業領域管理システム（ＷＳＭ）との２つのモジュールから成っている。問合せ変換はＱＵＥＬによって書かれたＬＣＳ問合せを、そのサイトのデータベースシステムで実行可能なＤＭＬのシーケンスに変換する。これをデータベースシステムで実行させ、結果をリレーションとしてＷＳに格納する。

ＷＳＭは、ＣＧＤＰのＴＳの制御下でサイト間処理を行なうためのモジュールである。ＷＳＭは、ＪＯＩＮ、ＴＲＡＮＳ、ＲＥＣという３つの副モジュールとＷＳと呼ばれる格納媒体とから成っている。目的ＬＤＰのＲＥＣは、図7.17のＲＥＣアルゴリズムによってソースＬＤＰのＴＲＡＮＳからのソースリレーション（r′）を受信する。ソースＬＤＰのＴＲＡＮＳモジュールはＴＲＡＮＳアルゴリズムによって、ソースリレーションを目的ＬＤＰへ転送する。ＪＯＩＮもＪＯＩＮアルゴリズムによってＲＥＣで受給されたソースリレーションr′とそのサイトにあった目的リレーションｒとの結合を行なう。これらはともにソートされているのでこれらのリレーションの結合は簡単である。このため作業領域（ＷＳ）も、ＳＡＭによってつくれる。

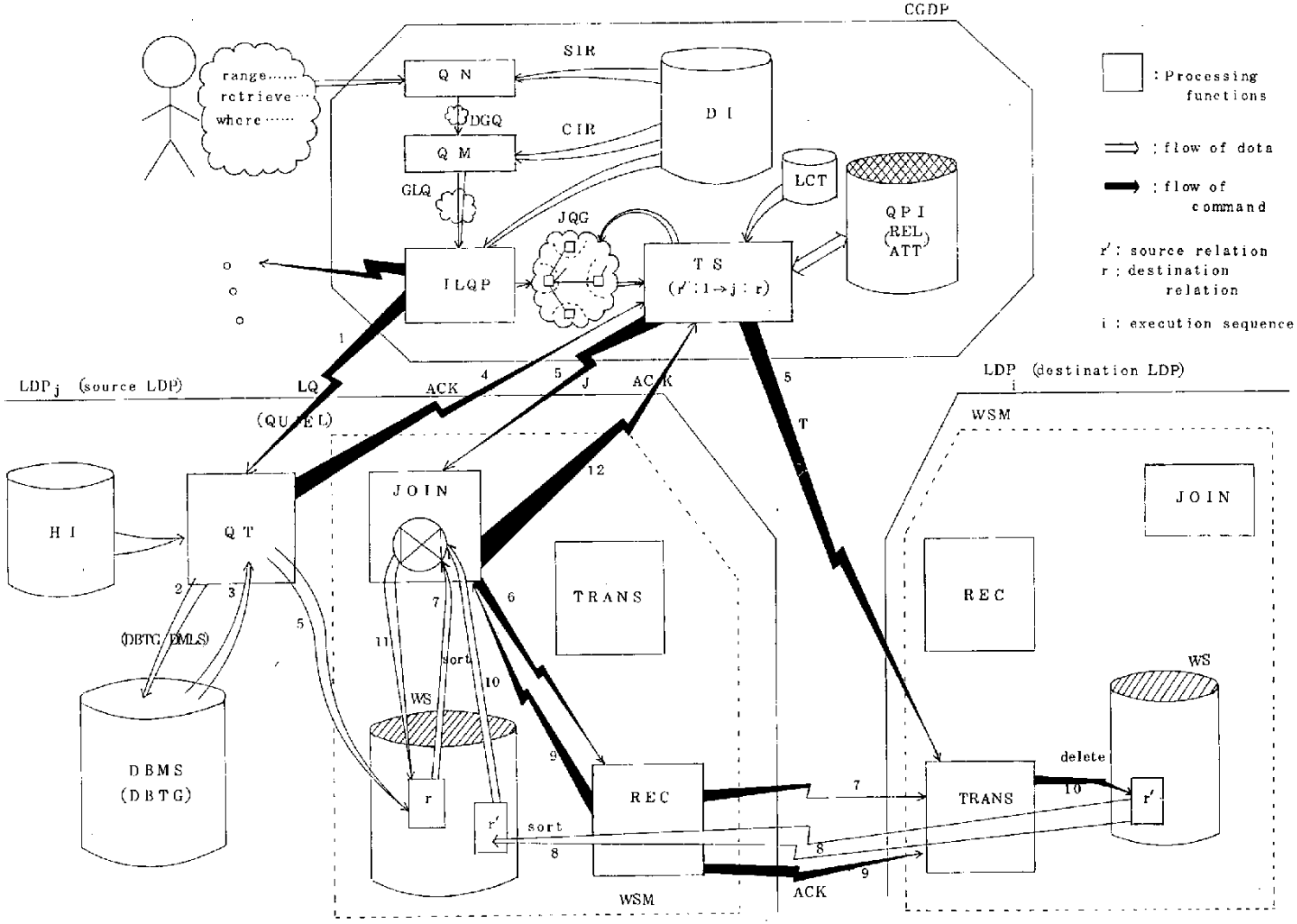


図7.2.3 The Architectures of the CGDP and LBPs

このようにWSMは、容易にインプリメントできる。
図7.2.3内の番号は、処理の順番を表わしている。

7.10 ＱＤＰ

問合せ分割システム（ＱＤＰ）は、これまで論じてきた問合せ分割（ＱＤ）を行なうシステムである。又ＱＤＰは全体データベースプロセッサの主要なサブシステムになる。ＱＤＰは、全体概念スキーマ（ＧＣＳ）層のユーザの発したＧＣＳ問合せを入力として、各サイト単位又は、各サイト間で処理可能なＬＣＳ問合せを生成し、実行させるシステムである〔図7.24〕。

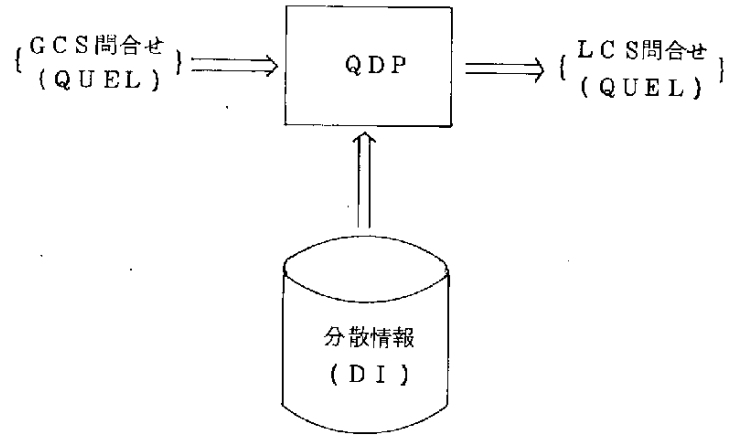


図7.24　ＱＤＰの概要

本システムの詳細については、8.4節に述べられているので、本節ではその概要だけを述べることとする。問合せ分割システム（ＱＤＰ）は、次の5つの主要モデュールから成っている。

1) ＧＣＳ問合せ木生成システム（ＧＱＴＲＧ）
2) 問合せ正規化システム（ＱＮ）
3) 問合せ変形システム（ＱＭ）
4) 局所問合せ処理システム（ＬＱＰ）
5) 転送マネージャ（ＴＳ）

ＧＣＳ問合せ木生成システム（ＧＱＴＲＧ）は、ＱＵＥＬによって記述されたＧＣＳ問合せから、ＱＤＰの内部表現としての2分木表現を生成する。図7.25は、ＧＣＳ問合せの2分木表現を示している。

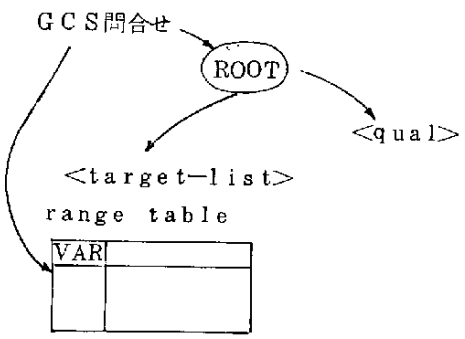


図7.25　ＧＣＳ問合せ木

<qual> は、ＧＣＳ問合せの条件部の論理式を2分木で表わしたものである。range tableは、ＧＣＳ組変数とＧＣＳリレーション名との対応表である。問合せ木の詳細については、8.2.2を参照されたい。問合せ木内で、ある属性に対応するノードはただ1つしかつくられない。

問合せ正規化システム（ＱＮ）は、ＧＱＴＲＧで生成された問合せの2分木表現を入力とし

て、＜qual＞を和正規形へ正規化する。正規化された条件式の各＜disjunct＞ごとに問合せ（分割されたＧＣＳ問合せ（ＤＧＱ））を生成する〔図7.26〕。

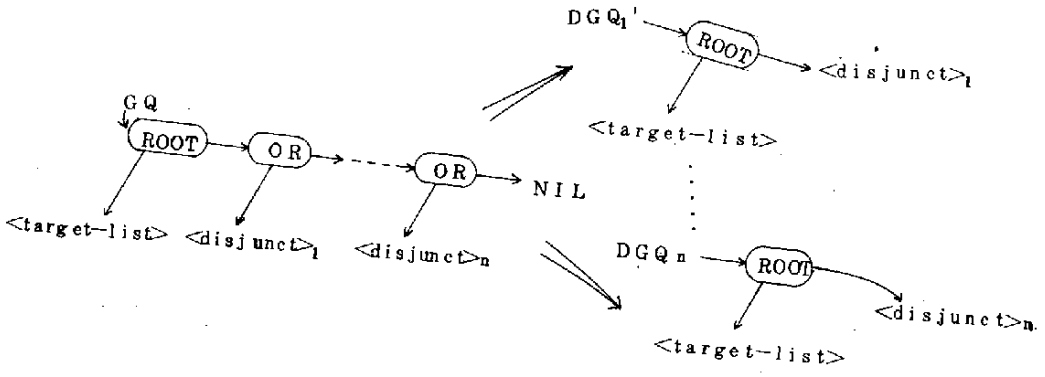


図7.26　問合せの水平分割

各分割されたＧＣＳ問合せ（ＤＧＱ）は、問合せ変形システム（ＱＭ）によって、分散情報（ＤＩ）内の対応情報（ＣＩ）を用いて問合せ変形される。問合せ木（Ｑ－ｔｒｅｅ）内で、ＧＣＳ属性のノードはユニークにつくられているので、このＧＣＳ属性ノードを対応するＬＣＳ属性の算術式の木の根ノードでおきかえればよい。ここで生成される問合せは、各ＤＧＱを対応するＬＣＳリレーションによって表わしたもので、全体ＬＣＳ問合せ（ＧＬＱ）である。

局所問合せ処理システム（ＬＱＰ）は、7.6で論じた初期局所問合せ処理（ＩＬＱＰ）を行なう。ＬＱＰでは、各サイトで独立に処理できるＬＣＳ問合せをまず生成する。ついで、これらのＬＣＳ問合せの終了後に残されたサイト間ＬＣＳ問合せを生成する。このサイト間ＬＣＳ問合せは、サイト間の結合式のみを持つ。これから結合問合せグラフ（ＪＱＧ）が生成される。

転送スケジューラ（ＴＳ）は、結合問合せグラフ（ＪＱＧ）を用いて、サイト間の問合せ処理を行なわせるものである。現在ＴＳのインプリメントは行なっていない。ＴＳのインプリメントには、実際のネットワークを実動させる必要がある。これには、7.9節で論じたようなＧＤＰとＬＤＰとのインプリメントが必要である。今後、我々のネットワークＪＩＰＮＥＴを用いたインプリメントを検討していきたい。

ＱＤＰの例を次に示してる。図7.7と同じ分散記述を考えてみよう。

```
* JIPNET-DDB STARTED TIME 10:44:45
#01
D1;
RANGE ( A, B     )  ( AAA:1, BBB:1              );
RANGE ( C, D     )  ( CCC:2, DDD:2              );
RANGE ( E        )  ( EEE:3                     );
RANGE ( F, G, H  )  ( FFF:4, GGG:4, HHH:4       );
DEFINE ESR  D1 ( D11, D12, D13       )
     ( D11 = B.B1 * 8, D12 = C.C1, D13 = F.F1 +2       )
     WHERE
               A.A1 = B.B1     AND  B.B1 = C.C2     AND  B.B2 = F.F2     AND
               A.A2 = "A"
     ;
DEFINE ESR  D2 ( D21, D22  )
     ( D21 = E.E2, D22 = H.H2 )
     WHERE     E.E1 = H.H2     AND  H.H3 = G.G2
     ;
DEFINE ESR  D3 ( D31 )
     ( D31 =  D.D1    )
     WHERE     D.D1 GE 500
     ;
```

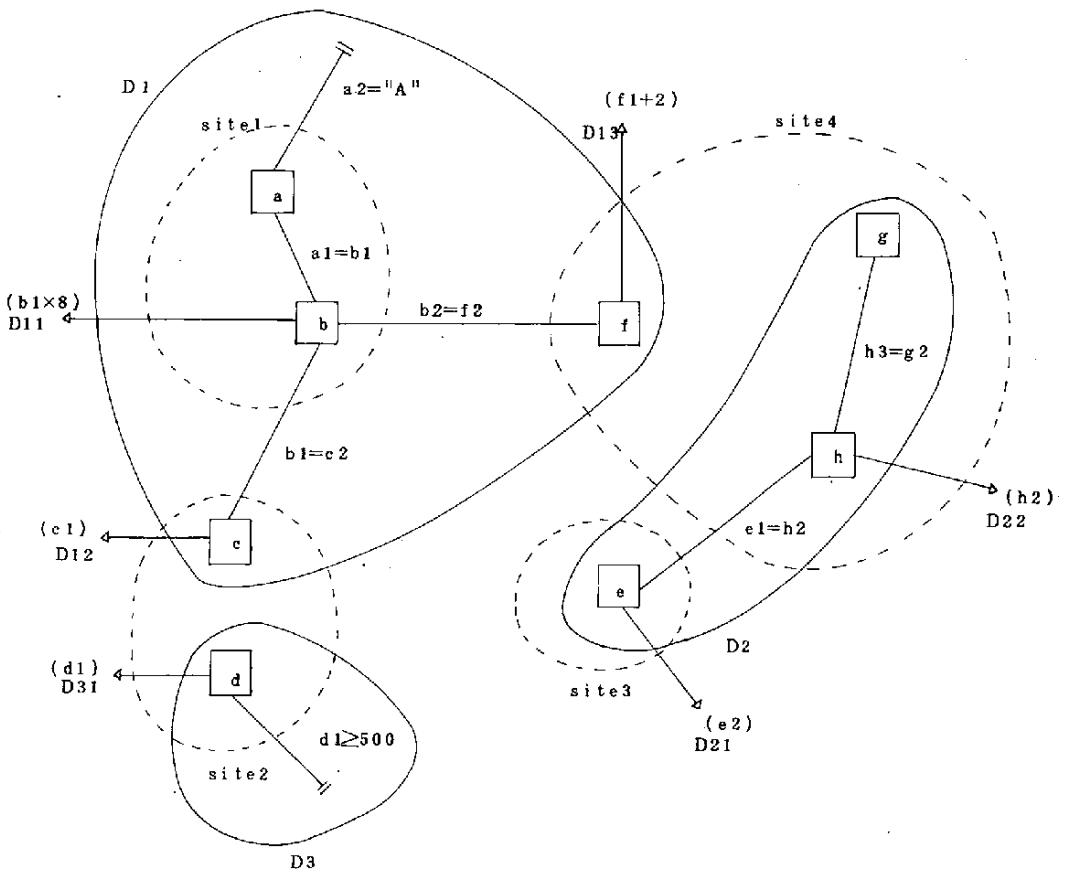


図7.27　分散記述とそのグラフ表現

この３つのＧＣＳリレーションＤ１、Ｄ２、Ｄ３に対して図7.28の様なＧＣＳ問合せが出されたとする。

```
QD;
RANGE ( D1, D2, D3 )     ( D1, D2, D3 );
RANGE ( D4, D5     )     ( D1, D2     );
RETRIEVE INTO TEST21     ( R1 = D1.D11, R2 = D4.D11, R3 = COUNT ( D5.D22 ) )
WHERE
     D1.D11  = D2.D21     AND  D2.D22  = D3.D31     AND  D3.D31  = D5.D21     AND
     D1.D11  = D4.D11     AND  D4.D11  = D3.D31     AND
     D1.D12  = "JIPNET-DDBS"                        AND
    (D4.D12  = "DISTRIBUTED DATABASES"              OR   D4.D12  = "NETWORKS" )
      ;
```

図7.28　ＧＣＳ問合せ（ＴＥＳＴ２１）

このＧＣＳ問合せは、ＧＣＳリレーションＤ１に対して２つのＧＣＳ変数Ｄ１とＤ２を、ＧＣＳリレーションＤ２に対しては、ＧＣＳ変数Ｄ２とＤ５とを、ＧＣＳリレーションＤ３にはＧＣＳ変数Ｄ３を宣言している。このＧＣＳ問合せの問合せグラフは図7.29のようになる。

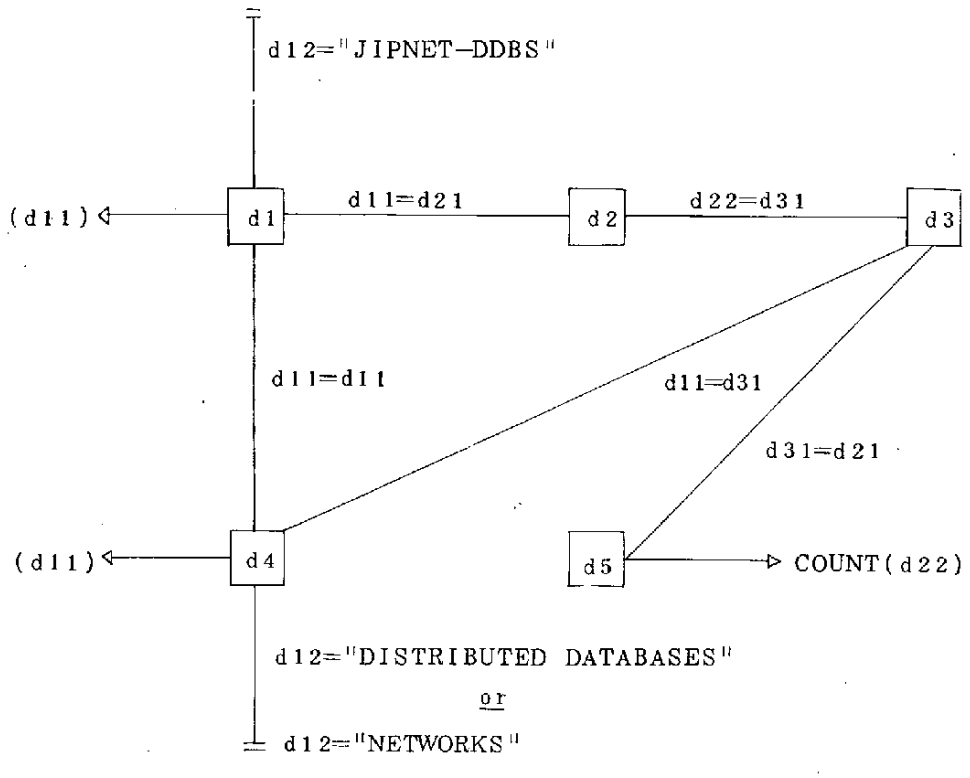


図7.29　ＧＣＳ問合せ（ＴＥＳＴ２１）の問合せグラフ

問合せ変形の結果は次のようになる。

```
GO;
B.B1 * 8 = E.E2 AND H.H2 = D.D1 AND D.D1 = ¥OE.E2 AND B.B1 * 8 = ¥OB.B1
* 8 AND ¥OB.B1 * 8 = D.D1 AND C.C1 = "JIPNET-DDBS" AND (¥OC.C1 = "DIST
RIBUTED DATABASES" OR ¥OC.C1 = "NETWORKS" )AND A.A1 = B.B1 AND B.B1 = C.
C2 AND B.B2 = F.F2 AND A.A2 = "A" AND ¥OA.A1 = ¥OB.B1 AND ¥OB.B1 = ¥OC.
C2 AND ¥OB.B2 = ¥OF.F2 AND ¥OA.A2 = "A" AND ¥OE.E1 = ¥OH.H2 AND ¥OH.H3 =
¥OG.G2 AND E.E1 = H.H2 AND H.H3 = G.G2 AND D.D1 GE 500
```

これに対応する全体ＬＣＳ問合せグラフ（ＧＬＱＧ）は図7.30のようになる。ここでダッシュのついたＬＣＳ変数は、ＧＣＳ組変数Ｄ4、Ｄ5に対応したものである。

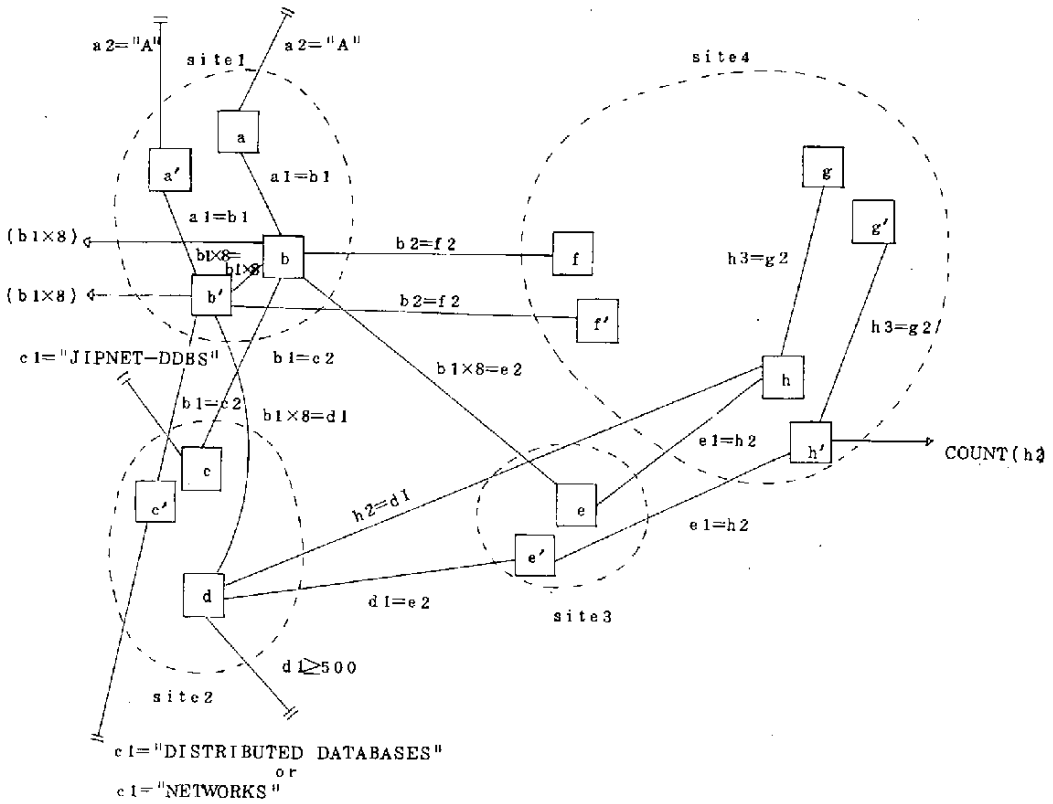


図7.30　全体ＬＣＳ問合せグラフ（ＧＬＱＧ）

ＩＬＱＰによって生成される各サイト単位の問合せを図７.３１に示す。

```
** LOCAL QUERY TO SITE 01 **
RANGE OF (A    ) IS (AAA);
RANGE OF (B    ) IS (BBB);
RANGE OF (¥0A  ) IS (AAA);
RANGE OF (¥0B  ) IS (BBB);

RETRIEVE INTO  NRELNAME01 ( B1 = B.B1 , ¥0BB1 = ¥0B.B1 , B2 = B.B2 ,
¥0BB2 = ¥0B.B2 )
WHERE B.B1 * 8 = ¥0B.B1 * 8 AND A.A1 = B.B1 AND A.A2 = "A" AND ¥0A.A1 =
¥0B.B1 AND ¥0A.A2 = "A" ;

** LOCAL QUERY TO SITE 02 **
RANGE OF (C    ) IS (CCC);
RANGE OF (D    ) IS (DDD);
RANGE OF (¥0C  ) IS (CCC);

RETRIEVE INTO  NRELNAME02 ( C2 = C.C2 )
WHERE C.C1 = "JIPNET-DOBS" ;

RETRIEVE INTO  NRELNAME03 ( D1 = D.D1 )
WHERE D.D1 GE 500 ;

RETRIEVE INTO  NRELNAME04 ( C2 = ¥0C.C2 )
WHERE (¥0C.C1 = "DISTRIBUTED DATABASES" OR ¥0C.C1 = "NETWORKS" );

** LOCAL QUERY TO SITE 04 **
RANGE OF (F    ) IS (FFF);
RANGE OF (G    ) IS (GGG);
RANGE OF (H    ) IS (HHH);
RANGE OF (¥0F  ) IS (FFF);
RANGE OF (¥0G  ) IS (GGG);
RANGE OF (¥0H  ) IS (HHH);

RETRIEVE INTO  NRELNAME05 ( H2 = H.H2 )
WHERE H.H3 = G.G2 ;

RETRIEVE INTO  NRELNAME06 ( H2 = ¥0H.H2 )
WHERE ¥0H.H3 = ¥0G.G2 ;

RETRIEVE INTO  NRELNAME07 ( F2 = F.F2 )
;

RETRIEVE INTO  NRELNAME08 ( F2 = ¥0F.F2 )
;

** LOCAL QUERY TO SITE 03 **
RANGE OF (E    ) IS (EEE);
RANGE OF (¥0E  ) IS (EEE);

RETRIEVE INTO  NRELNAME09 ( E2 = E.E2 , E1 = E.E1 )
;

RETRIEVE INTO  NRELNAME10 ( E2 = ¥0E.E2 , E1 = ¥0E.E1 )
;
```

図７.３１　ＬＣＳ問合せ

ＩＬＱＰ後に生成されるサイト間問合せと結合問合せグラフ（ＪＱＧ）とは図7.３２のようになる。

```
**  INTER-SITE QUERY  **
RANGE OF (NV01) IS (NRELNAME01);
RANGE OF (NV02) IS (NRELNAME02);
RANGE OF (NV03) IS (NRELNAME03);
RANGE OF (NV04) IS (NRELNAME04);
RANGE OF (NV05) IS (NRELNAME05);
RANGE OF (NV06) IS (NRELNAME06);
RANGE OF (NV07) IS (NRELNAME07);
RANGE OF (NV08) IS (NRELNAME08);
RANGE OF (NV09) IS (NRELNAME09);
RANGE OF (NV10) IS (NRELNAME10);

RETRIEVE INTO TEST21 ( R1 = NV01.B1 * 8 , R2 = NV01.*0BB1 * 8 , R3 =
COUN(NV06.H2 ))
WHERE NV01.D1 * 8 = NV09.E2 AND NV05.H2 = NV03.D1 AND NV03.D1 = NV10.E2
AND NV01.*0BB1 * 8 = NV03.D1 AND NV01.B1 = NV02.C2 AND NV01.H2 = NV07.F2
 AND NV01.*0BB1 = NV04.C2 AND NV01.*0BB2 = NV08.F2 AND NV10.F1 = NV06.H2
 AND NV09.E1 = NV05.H2 ;
```

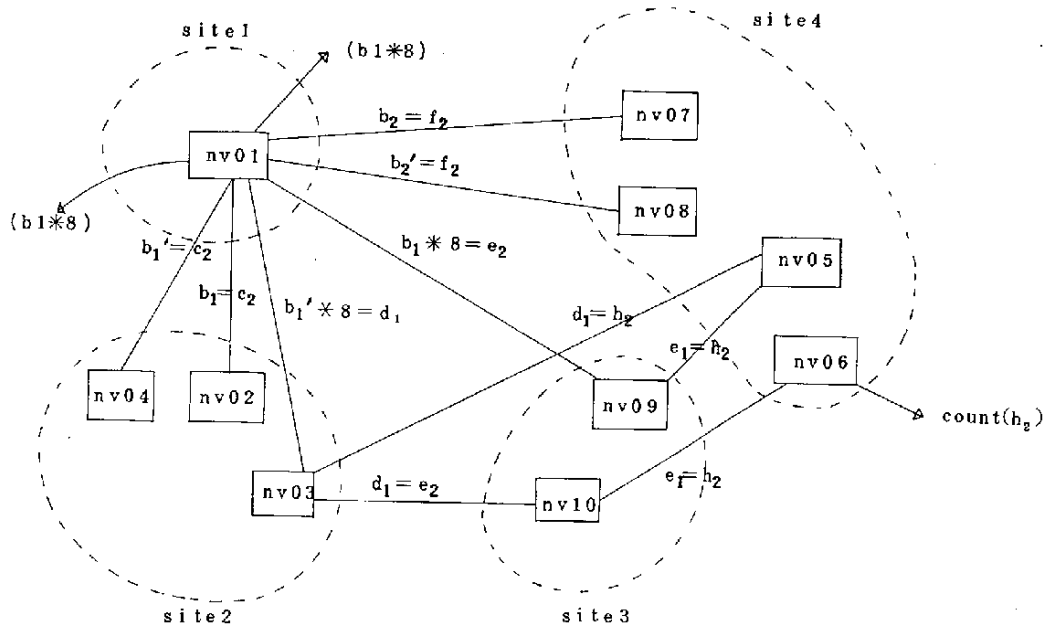


図7.３２　サイト間ＬＣＳ問合せとＪＱＧ

7.11 ＱＤＰの議論

本節では、我々が開発した問合せ分割システム（ＱＤＰ）の成果と問題点について論じる。まず問合せ正規化（ＱＮ）、問合せ変形（ＱＭ）、初期局所問合せ処理（ＩＬＱＰ）、転送スケジューリング（ＴＳ）、全体及び局所データベースプロセッサのアーキテクチャの各々について論じ、ついで全体について論じる。問合せ分割システムは、ＱＮの一部、ＱＭ、ＩＬＱＰのインプリメントを終わっている。これらはＰＬ／Ｉで記述され、ＰＬ／Ｉ文数で２０００、オブジェクトサイズ１１０ｋＢである（ただし、これにはＤＩＰＳもふくまれている）。仕様作製に３人月、インプリメントに２人月を要した。

Ａ．問合せ正規化（ＱＮ）

問合せの正規化は、問合せ変換（第５章）と問合せ分割において必要である。比較述語式からなる論理式は、積又は和正規形に変形される必要がある。問合せ分割では、和正規形にすることが必要であり、問合せ変換では変形和正規形（modified disjunctive normal form）に変換することが必要である。現在、我々の問合せ正規化システムは、任意形式の論理式を正規化できない。むしろユーザの責任で、正規化された論理式を入力する様にしている。

論理式の正規化は、図7.4に示した式のパターンを検出し、必要な式の変形を行なうことで一般に行なえる。このためには、リスト処理機能が必要になる。現在我々はPL/Iでインプリメントを行なっている。しかしＰＬ／Ｉは、リスト処理機能を持たない。このため、問合せを木構造表現するための自由セルの管理、ガーベジコレクター等をＰＬ／Ｉで作製している。又再帰的呼出し機能を実現するためにソフトウェアスタックを作製している。この様なシステムを用いて、パターンの認識と式の変換を行なうことは相当のプログラム労力とデバッグ労力を要してしまう。

ユーザの使い易さの点からは、この問合せ正規化システム（ＱＮ）は必要なものである。リスト処理用言語ＬＩＳＰは、人工知能研究中心に利用されてきているが、実際のシステムのインプリメンテーションにはその効率、既存ソフトウェアとの互換性（移行性）等の問題点からあまり使われていない。しかし、将来、ユーザの使い良さを本格的に実現するためには、上述した様な式の種々の操作が必要になる。従来の性能中心の考え方から、ユーザの使い良さという視点が重要になってくれば、ＬＩＳＰによるインプリメントも可能であり、我々も今後検討していきたい。

Ｂ．問合せ変形（ＱＭ）

ＧＣＳ問合せの２分木表現内で、この問合せの参照するＧＣＳ属性に対するノード（ＡＴＴＮ）｛8.2.2を参照｝はユニークにつくられる。ＧＣＳ問合せ内でＧＣＳ属性が複数回参照されても、この問合せ木内（Ｑ－ｔｒｅｅ）では、対応する属性ノードは１つだけつくられる。この問合せ木の性質は、木生成に必要なノード数を減じるとともに重要な性質を持っている。問合せ変形では、この性質を用いて、次の様に簡単に問合せの変形を行なえる。即ち、ＧＣＳ問合せ木が参照するＧＣＳ属性ノードを、分散情報木（ＤＤ－木）内の対応するＬＣＳ属性から成る算術式の木の根ノードの内容で置きかえればよい。この様に、ＧＣＳ問合せ木内の属性ノードの内容を置き換えるだけで、簡単に対応するＬＣＳ属性からなる問合せ、即ち全体ＬＣＳ問合せ（ＧＬＱ）に変形

できる。この意味で、ＧＣＳ問合せ木内で属性ノードをユニークに保持することは有効である。

問合せ変形では、ＧＣＳ問合せ内のＧＣＳ属性をＬＣＳ属性からなる算術式で置き換えるとともに、分散記述内の対応する条件式が接合される。このため、新たに生成された条件式内の論理式の無矛盾性のチェックが必要となる。例えば、ＧＣＳ問合せの条件式がｘ．ａ＞５００であり、さらにｘ．ａ＜４８０が接合されるならばｘ．ａ＞５００ and ｘ．ａ＜４８０となり、この真理値は常に偽である。条件式が常に偽である問合せの答は存在しないので、この様な問合せを対応するサイトへ発する必要はない。更に、簡約化できる算術式はなるべく簡約化（ｓｉｍｐｌｉｆｙ）した方がよい。例えばｘ．ａ＋ｘ．ａ＋５ は 2＊ｘ．ａ＋５ とした方がよい。この様に、実際の問合せを発する以前に次のような式の処理をしてしまうことは、処理効率上望ましい。

1）算術式の簡約化　　e．g．　x．a＋5＝y．b＋5 ⇒ x．a＝y．b

4－(x．a＋8)＝y．b ⇒ －4－x．a＝y．b

2）論理式の簡約化　　e．g．　x．a＝x．a and x．a＝y．b ⇒ x．a＝y．b

x．a＝y．b and x．a＝y．b ⇒ x．a＝y．b

3）論理式の恒偽性のチェック

（e．g．　x．a＞500 and x．a＜450 ⇒ false）

現在の我々の問合せ分割システムは、上述の様にチェックは行なっていない。問合せ正規化（ＱＮ）で述べた様に、式の操作を行なうには、高度な数式処理（formula manipulation）機能が必要となってしまう。我々のインプリメントに用いているＰＬ／Ｉ言語では、このような高度な記号処理を行なうことは困難である。問合せ正規化と同じくＬＩＳＰによるインプリメンテーションが望ましいと考えられる。

Ｃ．初期局所問合せ処理（ＩＬＱＰ）

問合せ変形（ＱＭ）で生成された全体ＬＣＳ問合せ（ＧＬＱ）から、各サイト単位のＬＣＳ問合せを生成し、かつ、その処理後のサイト間問合せを、この初期局所問合せ処理（ＩＬＱＰ）は行なうことができる。これまで述べた問合せ正規化（ＱＮ）と問合せ変形（ＱＭ）とが問合せの表現の変換（ＧＣＳ要からＬＣＳ要素への変換）であるのに対して、初期問合せ処理（ＩＬＱＰ）と後述する転送スケジューラ（ＴＳ）とは、問合せのサイト間処理のためのものである。

初期局所問合せ処理（ＩＬＱＰ）システムは、7.6節で述べた様に各サイト単位の問合せをまず生成し、ついでサイト間の問合せを生成することができる。サイト単位の問合せは、サイト内の結合リンクによって連結なノード（リレーション）のグループに対して出される。サイト内で単独なノードに対しては、サイト間の結合に必要な属性（結合属性）について射影（ｐｒｏｊｅｃｔｉｏｎ）を取るような問合せを出せる。

Ｄ．転送スケジューリング（ＴＳ）

本報告書では、転送スケジューリングのアルゴリズム（ＴＳＡと呼ぶ）の提案を行なった。このＴＳＡの特徴は次の様である。

1）ＴＳＡの目標は、第1に、問合せ分割（ＱＤ）の必要とする情報、即ち、分散情報（ＤＩ）

を最少化し、かつなるべく静的なものにすることである。これは、ＱＤをＤＤＢＳ内の各サイトが持ち、かつＤＩはＱＤと同一サイトにおくためであるＱＤの必要情報としては、次のものがある。

a）１）の表現変換で必要とするＧＣＳリレーションとＬＣＳリレーションとの対応情報

b）変換されたＬＣＳリレーションがどこにあるかを示す所在情報

c）最適化のために必要なＬＣＳリレーションのカーディナリティ、これらの属性の選択度といったパフォーマンス情報

ａ）とｂ）の情報は、時間とともに変化しない静的な性質を持っている。しかし、ｃ）はａ）とｂ）とに対してより動的な性質を持ち、かつより多くの格納コストを必要としていることが言える。よって、分散情報（ＤＩ）としては、ａ）とｂ）とを保持させることにする。

2）今までの問合せ分割（ＱＤ）アルゴリズムは、実行前に全ての転送スケジュールを決めてしまうものであった。これらは又、問合せの条件式の選択度を用いて中間結果の見積もりを行なうことによって、最適な方法をみつけるものである。しかし、我々はこの選択度が現実のリレーションによく適応するかどうか疑問である。よって、実行前にオフライン的に全ての転送スケジュールを決定してしまうためには、十分な選択度に関する情報を得れない。このため我々のＴＳＡは、１つの問合せ（ＧＱ）に対する中央コントローラとしての統合全体データベースプロセッサ（ＣＧＤＰ）が、必要なデータをもつ局所データベースプロセッサ（ＬＤＰ）の実行をモニターしながら、適宜転送方法を決めていくようにした。

3）次の転送方法（我々はステージと呼ぶ）は、モニターされた情報に基づいて問合せのグラフ表現としての結合問合せグラフ（ＪＱＧ）の縮退を通して決定される。

4）転送コストとしては、転送距離も考慮した。転送コストが距離不依存との仮定は、現在及びここしばらくの間は現実的でないと考える。

5）ある閾値を設け、あるサイトから他への転送コストがこの値より小さい時は、積極的に転送させることによって、転送のパラレリズムを得るようにした。これは良い応答時間をもたらすと考える。

我々の転送スケジューリングアルゴリズム（ＴＳＡ）において閾値（ＴＨＶ）の役割は重要である。ＴＨＶの値が大きければ、転送のパラレリズムは高まるが、逆に小さくなれば、パラレリズムは小さくなる。しかし高パラレリズムは、適したもののみの転送が行なわれることを意味しない。したがって、全転送コストの増大をもたらすかもしれない。統合ＧＤＰにおいてこの閾値（ＴＨＶ）の値の決定をいかにするかは今後の課題である。又、分散情報（ＤＩ）はＬＣＳリレーションのカーディナリティ等のパフォーマンス情報をもたないので、ＴＳＡの初めにいかにＴＨＶを決めるかも今後の課題である。

任意のサイトｉとｊの間の単位転送コストは、論理転送コストテーブル（ＬＣＴ）内に格納されている。ネットワーク内の転送コスト（時間）ｔは、ネットワーク内の回線が均一とすると一般に転送量（Ｖ）と転送パス長とに比例する。転送パス長はパス内のノード（ＩＭＰ）の数ｎで

ある。即ちt＝c・V・n、ここでcは定数、よって我々はＬＣＴ内には、サイトｉとｊとの間の最短パス内のノード数を格納している。しかし実際に、サイト間でこの最適パスを通って転送されるとは限らない。さらに、我々は、待行列遅延（queueing delay）はないものとしている。分割型にＤＤＢＳを設計した場合には、利用度の高いデータを冗長に他のサイトにコピーすることによって1つのサイトへのトラフィックの集中を防ぐことができる。しかし、統合型分散型データベースシステムでは、既にデータは存在しており、新たにコピーをもうけることはないと考えてよいだろう。従って、利用度の高いサイトには多くのトラフィックが集中し得ることになり、このサイトを中心としたネットワークノードに待ち行列は生成し得ることにもなる。待ち行列生成はルーティングによって前述した転送パスも変化させることになる。ネットワークのこの様な動的性質をどのように静的に反映させていくかも今後の課題と考えられる。

転送スケジューリングシステム（ＴＳ）のインプリメンテーションは、局所データベースプロセッサ（ＬＤＰ）と全体データベースプロセッサ（ＧＤＰ）との実現を合わせて考えていく必要がある。今後シミュレーションを通して転送スケジューリングアルゴリズムの検討をすすめていきたい。

転送スケジューリング（ＴＳ）及び初期問合せ処理（ＩＬＱＰ）は、ネットワークを介した処理を行なう必要がある。このため、ネットワーク及び各データベースシステムの障害に対する回復手段が必要になる。ＩＬＱＰに対する障害は、もし冗長コピーが他のサイトになければ、ここで処理を中断して、このサイトの復旧を待つことになる。転送スケジューリング中に、あるサイトの作業領域管理システム（ＷＳＭ）が障害を起こした時は、このサイトの復旧を待つ間、このＷＳ内の中間結果の生成をさかのぼって他の処理を進めていくこともできる。このためには、結合問合せグラフ（ＪＱＧ）の履歴を保存しておく必要がある。今後の重要なテーマである。

E．ＧＤＰとＬＤＰのアーキテクチャ

上述した転送スケジューリングアルゴリズムに基づいて問合せ分割を行なうための分散型データベースシステムアーキテクチャを示した。各局所データベースプロセッサ（ＬＤＰ）は、それが管理するデータベースシステムに対する問合せの変換器（ＱＴ）に加えてサイト間処理のためのＷＳＭ（WS manager）が必要となる。このＷＳＭを各サイトにいかに安価に容易にインプリメントできるかはＤＤＢＳ設計の重要なポイントになる。ＷＳＭは、問合せ分割処理の中間結果をリレーション形式で格納するＷＳ（working space）を中心に、この中のリレーションの結合を行なうＪＯＩＮ、他のサイトへの転送を行うＴＲＡＮＳ、他からの受信を行なうＲＥＣから成っている。1つのステージは、リレーションの転送と結合とから成るが、この中に占める転送コスト（時間）はその大半を占める。よって、転送時間を利用して結合を行なう以前に、結合される2つのリレーションのソートを行なうことができる。他のサイトから受信されるリレーションは、受信しながらソートを行なえる。結合する処理は、簡単にインプリメントできる。我々はこの様に、ＷＳＭのインプリメンテーションは容易であり、処理も簡単であるので各サイトがＷＳＭを保持するコストは比較的安価なものと考える。さらに、ＷＳとしてＳＡＭを用いる

ことによって、各サイトのWSMは共通のものを用いられるだろう。

各局所データベースプロセッサ（ＬＤＰ）は、１つのデータベースシステムに対して存在している。よって１つのサイトは、複数のＬＤＰを持つことになる。サイト内のデータベースシステム間のリレーションの転送は容易なのでWSを各ＬＤＰが１つづつもつのではなく、各サイトが１つ持つ方がよいかもしれない。この時には、ＷＳＭは、ＧＤＰが管理すべきだろう。

Ｆ．問合せ分割システム全体について

分散記述（ＤＤ）〔6.2節〕は、１つのＧＣＳリレーションに対して複数の副定義（sub-definition）を持ちうる。問合せ分割の問合せ変形（ＱＭ）では、これが参照するＧＣＳリレーションの副定義の各組合せに対して１つの全体ＬＣＳ問合せ（ＧＬＱ）を生成する。例えば、ＧＣＳリレーションＤ１、Ｄ２、Ｄ３があり、各々が４、２、２個の副定義をもっているとしよう。するとこれらを参照する１つのＧＣＳ問合せから合計４・２・２＝１６個の全体ＬＣＳ問合せが生成され、これらは互いに独立に処理されることになる。これらの問合せが参照するＬＣＳリレーションは共通のものもあり、これらが全体ＬＣＳ問合せ（ＧＬＱ）の数だけアクセスされることになる。この冗長アクセスを防ぐために、１つのＧＣＳ問合せから生成される全体ＬＣＳ問合せのグラフ（ＧＬＱＧ）を集めて、これらの共通部分を抽出することが考えられる。共通部分はただ１回だけアクセスし、共通でない部分は各ＧＬＱごとにアクセスさせる。しかし、複数の全体ＬＣＳ問合せグラフ（ＧＬＱＧ）の中から共通部分を抜き出すことは容易ではない。

この問題の他の解法としては、分散記述（ＤＤ）を変えることである。現在の分散記述は、各副定義に対応する結合を行なった後に和（union）を行なうことを表わしている。この理由は、リレーショナル計算表現がリレーション間の演算としてリレーションの和を表わせず、結合のみを表わせるからである。即ち、各副定義は、リレーション間の結合を表わし、副定義の並びが各結合の結果の和（union）を表わしている。これらを関係代数（結合、和等）演算のシーケンスによって表わす。また問合せも関係代数演算によって表わす。関係代数演算の変換規則を用いて最適なシーケンスを生成することも考えられる。

しかし、我々の現在の方法も、各ＧＣＳリレーションが副定義を１つづつしか持たないならば有効に問合せ変換できる。和をとる必要のある処理をどのように最適化するかは、今後の課題である。

ＧＣＳ問合せは、aggregate関数を持たないとしている。もし、aggregate関数を持つならば、これらは独立のＧＣＳ問合せとしてまず処理される。この処理結果の値で最初のＧＣＳ問合せのaggregate関数を置き換える。これはaggregate をふくまない問合せとなる。aggregate 関数を有効に処理させる方法は今後の課題である。

7.12 まとめと問題点

本章では、ＧＣＳ問合せ（ＧＱ）を各サイトで分散して処理させる、いわゆる問合せ分割（ＱＤ）

について論じた。問合せ分割のインプリメントについては、ＴＳを残して終了しており8.4節を参照されたい。

問合せ分割は、次の２つの主要部分に分けて考えられる。

1) 全体概念スキーマ（ＧＣＳ）を参照する問合せ（ＧＱ）から、対応する局所概念スキーマ（ＬＣＳ）を参照する問合せ（ＧＬＱ）への表現の変換をする部分。

2) 全体ＬＣＳ問合せ（ＧＬＱ）は、複数サイトのＬＣＳリレーションを参照している。このためサイト間での結合を処理するためのリレーションの転送方法の決定をする部分。

１）は、〔ＳＴＯＮＭ７６〕等による問合せ変形（query modification）の手法を用いて行なうことができる。

２）は、問合せ分割（ＱＤ）問題で現在最も関心を集めているものである。何故なら、この転送方法が最も分散型データベースシステムのパフォーマンスを決定するからである。よって我々も２）の問題について多くの検討を行ない、次のような特徴をもつ転送アルゴリズム（ＴＳＡと呼ぶ）の提案を行なった。

問合せの正規化（ＱＮ）において、我々はＧＱをまず和正規形（ＤＮＦ）に変換すると述べた。理由として、x. a=y. a or y. b=z. b のような異なった変数を参照する比較述語の和（disjunction）があり得るからであった。しかし、このようにＤＮＦに正規化することに対する論点として次の点をあげることができる。

1) ほとんどの問合せは、not, or をふくまないであろう。

2) or が使われるパターンは、x. a=500 or x. a=600 or x. a=800 といった論理式の様に、同一の変数（しかも同一の属性）を参照する比較述語の和である。

２）で述べた様なパターンに関しては、ＤＮＦにする必要はないと考えるが、異なった変数を参照する比較述語については、ＤＮＦに変形し各disjunct ごとに問合せをつくることが処理上必要である。よって、論理式の論理和を今述べた２つのパターンに分けることが必要となる。例えば次の論理式を考えてみよう。

(x. a=y. a or x. a=z. a) and (x. b="A" or x. b="B")

これは、次のように変形されるべきである。

(x. a=y. a and (x. b="A" or x. b="B")) or

(x. a=z. a and (x. b="A" or x. b="B"))

即ち、～～～ の引かれたclauseはその中のリテラルが全て同一の変数を参照しているので、分解されないでそのままにしておかれる。このような正規形を我々は、変形和正規形（modified disjunctive normal form or MDNF）と呼ぶ。ＭＤＮＦは形式的に次のように定義される。

MDNF::=<m−disjunct>$_1$ or <m−disjunct>$_2$ or … or <m−disjunct>$_n$

<m−disjunct>::=<m−conjunct>$_{i_1}$ and <m−conjunct>$_{i_2}$ and …<m−conjunct>$_{i_m}$

$$<\text{m-conjunc}>_{ij} ::= <\text{c-pred}> \mid <\text{c-pred}>_{ij1} \ \underline{\text{or}} \ <\text{c-pred}>_{ij2} \ \underline{\text{or}} \cdots \underline{\text{or}} \ <\text{c-pred}>_{ijk_{ij}}$$

$$<\text{c-pred}> ::= <\text{a-exp}> \ <\text{cop}> \ <\text{a-exp}>$$

ここで$<\text{c-pred}>_{ij1}$（$l = 1,2,\cdots, k_{ij}$）は全て同一の変数を参照している。

問合せ内の条件式をＭＤＮＦに変数するは、記号処理能力が必要となる。このため我々は簡単な正規化（8.4節を参照）のみを行なっているだけである。

問合せ変形（ＱＭ）において、分散記述（ＤＤ）内の副定義（subdefinition）の組合せに対応して複数の全体ＬＣＳ問合せ（ＧＬＱ）が生成され、各々は独立に処理される。実際は各々のＧＬＱは互いに共通して処理できる部分をかなり持っている。現在の手法では各ＧＬＱは全く独立に処理されるために、これらの共通部分は各ＧＬＱに対応して重複して処理されてしまう。この重複処理を防ぐために、各ＤＧＱで生成されたＧＬＱＧ（global LCS query graph）を用いて共通部分を見つけ出すことが考えられる。この検出された共通部分は、共通して１回だけ処理し、他の部分を各ＧＬＱごとに処理させれば重複処理をかなり防げる。１つの問合せからｎコのＧＬＱ（ＧＬＱi，i＝1,2，…，ｎ）が生成されたとする共通部分としては、全てのＧＬＱG_i（i＝1,2，…，ｎ）が保有しているノードとリンクとから成るグラフである。しかし、これはＴＳをかなり複雑化してしまう。

ＧＣＳのＲＤは、ＬＣＳのＲＤから定義された視野とも考えられる。ＧＣＳが１つのサイトのみからつくられるとすれば、ＧＣＳはリレーショナルモデルの視野と等価になる。このためＱＤにおけるＱＮ，ＱＭ，ＩＬＱＰは、視野プロセッサ（view processor）として位置づけることができる。問合せ変換（ＱＴ）をＤＢＴＧデータベースシステムに対するリレーショナルインタフェースとして問合せ分割（ＱＤ）を視野プロセッサとして、１つのサイトのデータベースシステムに設定することも出来る。

8. 分散型データベースシステム（ＪDDBＳ）のインプリメンテーション

これまで、分散型データベースシステム（ＪＩＰＮＥＴ－ＤＤＢＳ）（以降ＪＤＤＢＳと呼ぶ）実現のために、次の点について論じてきた。

1） スキーマ層設計　　ⅰ） 同種化〔４章〕
　　ⅱ） 統合化〔６章〕
2） アクセス機能　　ⅰ） 問合せ変換〔５章〕
　　ⅱ） 問合せ分割〔７章〕

本章では、これらで論じたことを基にして実現したシステム、ＨＩＰＳ、ＤＩＰＳ、ＱＴＰ、ＱＤＰの各々について論じる。ＨＩＰＳは、異種性情報生成システムと呼ばれ、同種化の機能を持つ。ＤＩＰＳは、分散情報生成システムと呼ばれ、統合化の機能を持つ。ＱＴＰは、問合せ変換を行い、ＱＤＰは問合せ分割を行う。

8.1 ＨＩＰＳ(Heterogeneity Information Processing System)

ＨＩＰＳは、ＬＣＳ／ＨＩ（異種性情報）に関する記述を解読し、５種類１０個のＨＩ情報のリレーション（ファイル）を生成するとともに、このＨＩ情報からＬＣＳのリレーショナル記述（ＲＤ）を生成するものである。図８－１に、ＮＤＤ／ＨＩＰＳの概要を示す。入力するＬＣＳ記述は、各ＤＢＭＳ固有のスキーマ記述をＥ－Ｒモデルに基づいて記述したものである。

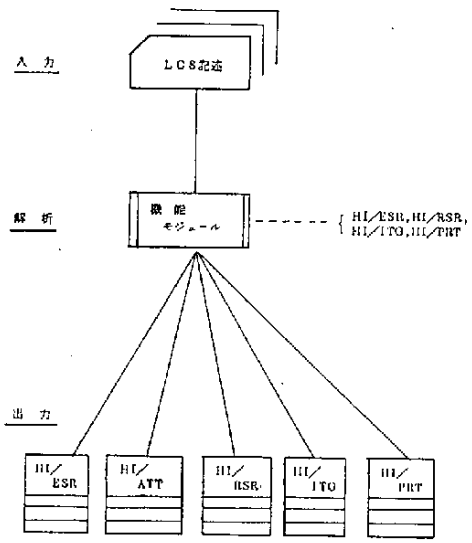


図８－１　ＮＤＤ／ＨＩＰＳの概要

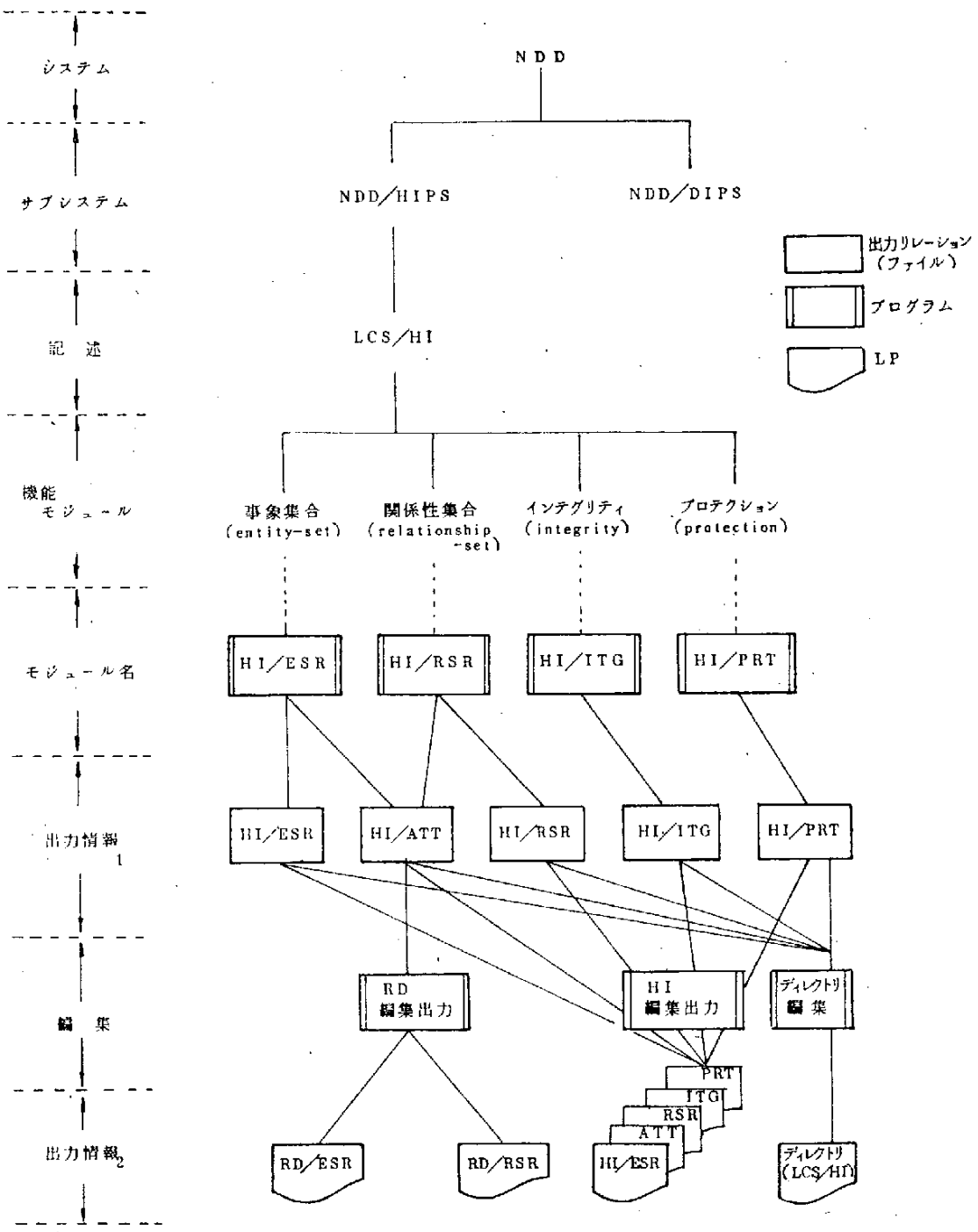


図8－2　ＨＩＰＳのモジュール関連図

8.1.1 ＬＣＳ／ＨＩ記述

ＬＣＳ／ＨＩ記述は次のような体系になっている。またこれを図示すると図８－３のようになる。

※　ＬＣＳ／ＨＩ記述の骨格

- １つ以上のＤＢエントリの記述

※※　ＤＢエントリ記述の骨格

- １つのＤＢステートメント（＃＃ＤＢ）の記述
- １つ以上のモジュールエントリの記述
- １つのＤＢエンドステートメント（＃＃ＥＮＤ）の記述

※※※　モジュールエントリ記述の骨格

- １つ以上の"事象集合／関係性集合に関するステートメント"の記述

表８－１は、各モジュールエントリを構成するステートメントの種類（番号）毎の許される最少、最大数を示したものである。

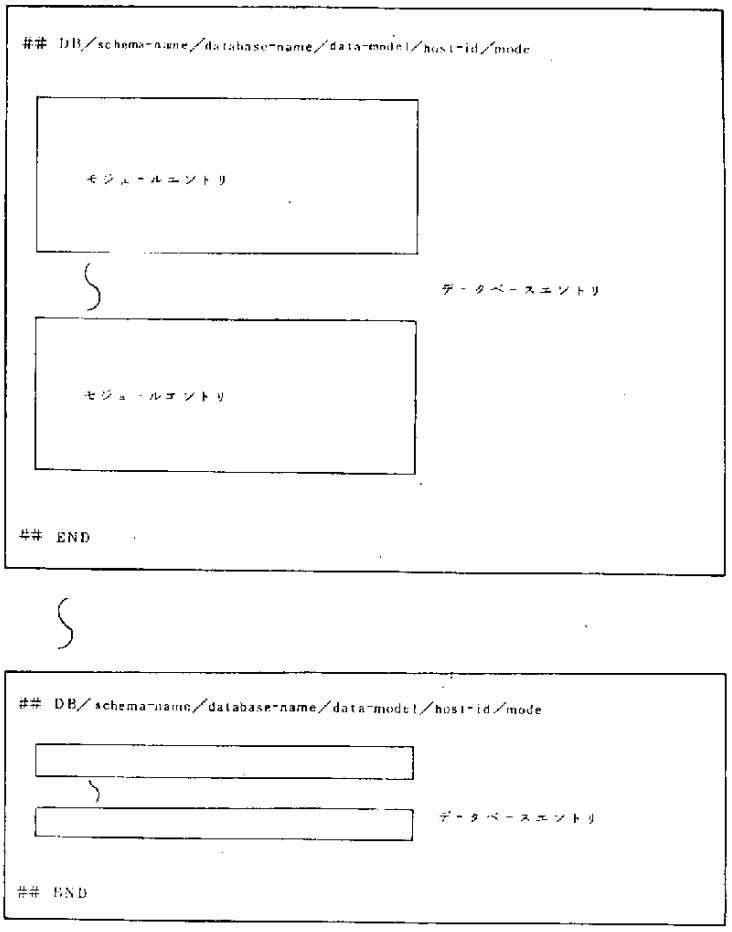


図８－３　ＬＣＳ／ＨＩ記述：構成例

表８－１　エントリと記述ステートメント数の関係

エントリ名	ステートメント	最大	最大	備　　考
データベース	＃＃ＤＢ．	1	N	―
	＃＃ＥＮＤ	0	N	
ＥＳＲ	１０１	1	1	同一エンタティセット、もしくはリレーションシップセット内での条件である。
	２０１～２９９	1	９９	
ＲＳＲ	１０１	1	1	
	２０１	0	1	
	３０１～３９９	1	９９	
	４０１～４９９	1	９９	
	５０１～５９９	0	９９	
ＩＴＧ	１０１	0	1	
	２０１～２９９	0	９９	
	９０１～９９９	0	９９	
ＰＲＴ	１０１～１９９	0	９９	
	２０１	0	1	
	３０１～３９９	0	９９	
	９０１～９９９	0	９９	

さらに、各モジュールエントリのステートメント番号と出力ファイル（リレーション）との対応関係を図８－４に示す（→；１：１に対応、→→；１：ｎに対応）。

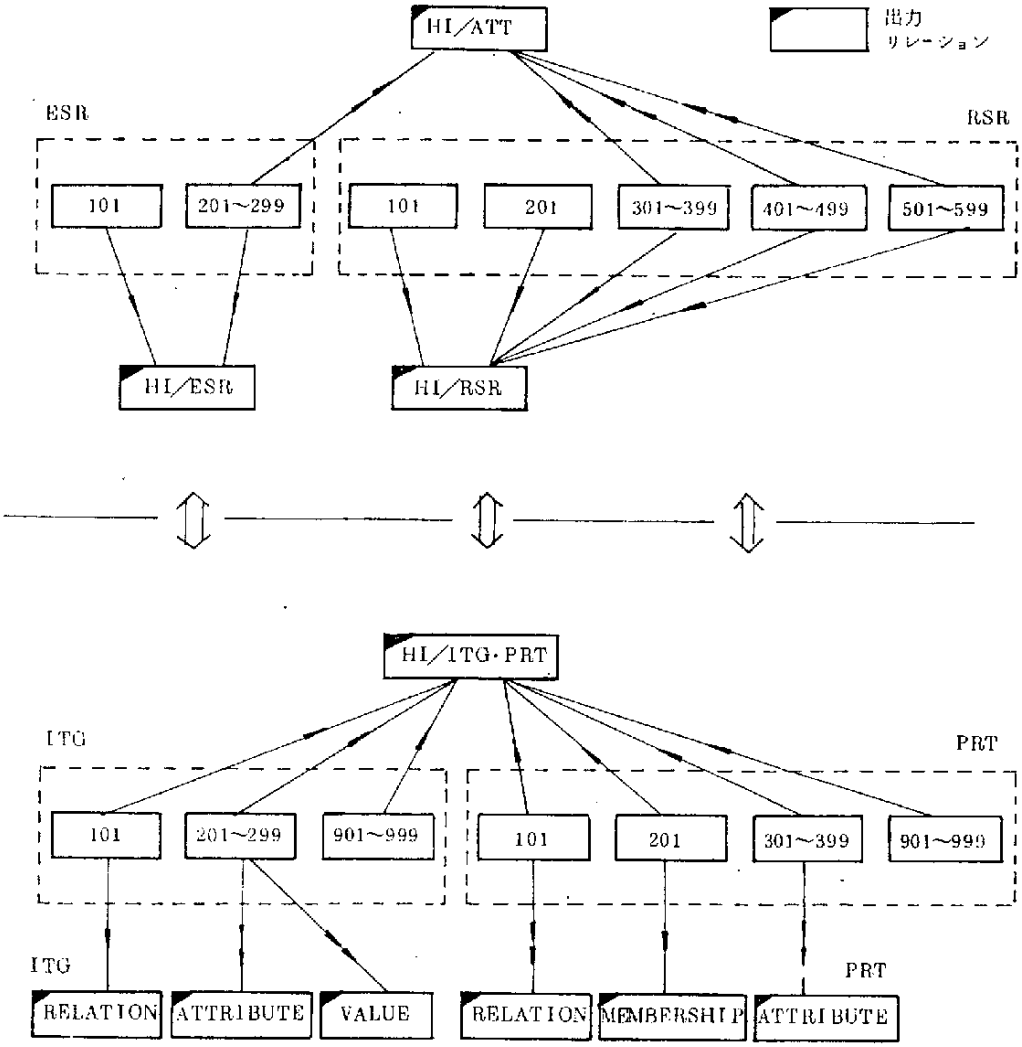


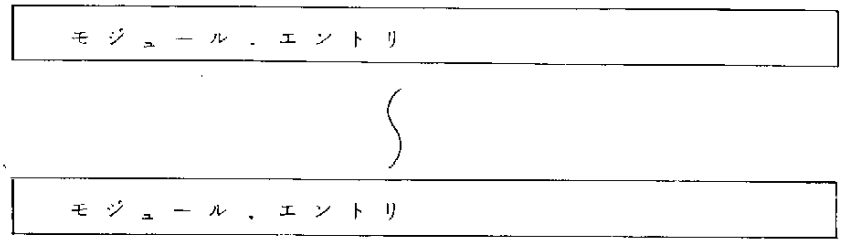
図８－４　ステートメント番号と出力ファイル（リレーション）の関係

A．データベースエントリ（ＤＢ　entry）

データベースエントリは、ＮＤＤ／ＨＩＰＳで処理される単位の１つであり、スキーマ（schema）毎に記述される。

一般形式：

##DB／schema-name／database-name／data-model／host-id／mode



##END

解　説：

##ＤＢ　ステートメントは、ＨＩＰＳで処理するスキーマ名（サブスキーマ名）、データベース名、データモデルの種類（ＤＢＴＧ、ＩＭＳ、ＲＥＬＡＴＩＯＮＡＬ）、ホスト名、処理モード（Ｃ：創成、Ｕ：更新、Ａ：追加、Ｄ：削除）を記述している。

##ＤＢ　ステートメントにエラーがある場合は、そのＤＢ　entry のＨＩ情報は生成しない。

B．モジュールエントリ（module entry）

モジュールエントリは、データベースエントリの中で記述され、ＬＣＳのＨＩ記述を具体的に行うものである。

一般形式：

module-name mode { entity-set / relationship-set } detail-description

解　説：

- module-name

  モジュールエントリには、ＨＩ／ＥＳＲ、ＨＩ／ＲＳＲ、ＨＩ／ＩＴＧ、ＨＩ／ＰＲＴの４種類があり、それぞれ、ＥＳＲ、ＲＳＲ、ＩＴＧ、ＰＲＴのいづれかを指定する。

- mode

  処理モード（Ｃ：創成、Ｕ：更新、Ａ：追加、Ｄ：削除）のいづれかを指定する。但し、今回の implementation では、全てＣとみなしている。

- entity-set 又は、relationship-set

  処理対象の事象集合名、関係性集合名を指定する。英字で始まる１６文字以内の文字列

とする。

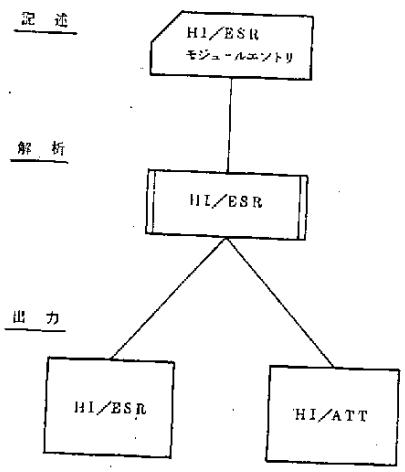
・detail-description

ＬＣＳ／ＨＩに関する詳細記述を行う。モジュールエントリにより、それぞれ複数の記述形式の識別は、名形式が一意に定まるstatement-no によって行う。

a) ＨＩ／ＥＳＲ　モジュールエントリ

ＨＩ／ＥＳＲモジュールエントリでは、ＬＣＳレベルでの事象集合に関する記述を行う。解析の結果は、ＨＩ／ＥＳＲおよびＨＩ／ＡＴＴに出力される。なお、ＨＩ／ＡＴＴには、ＨＩ／ＲＳＲモジュールエントリの結果も合わせて出力される。

図８－６に、ＨＩ／ＥＳＲモジュールエントリの構成を示す。




```
*
*          **********************
*          *  HI/ESR  SAVE AREA  *
*          **********************
*
01  HI-ESR-SAVE-AREA.
    03  Hi-ESR-101.
        05  HES-101-PROCESS-MODE          PIC  X.
        05  HES-101-ENTITY                PIC  X(16).
        05  HES-101-AREA-ROOT             PIC  X(16).
        05  HES-101-MODE                  PIC  X(02).
        05  HES-101-PROTECTION            PIC  X(02).
        05  HES-101-INTEGRITY             PIC  X(02).
        05  HES-101-CARDINALITY           PIC  9(04).
*
    03  HI-ESR-201                  OCCURS  99 TIMES.
        05  HES-201-PROCESS-MODE          PIC  X.
        05  HES-201-ATTRIBUTE             PIC  X(16).
        05  HES-201-TYPE                  PIC  X.
        05  HES-201-LENGTH                PIC  9(03).
        05  HES-201-KEY-TYPE              PIC  X(05).
        05  HES-201-PROTECTION            PIC  X(02).
        05  HES-201-INTEGRITY             PIC  X(02).
        05  HES-201-CARDINALITY           PIC  9(04).
        05  HES-201-VALUE-SET             PIC  X(12).
```

図８－５　ＨＩ／ＥＳＲの記述形式

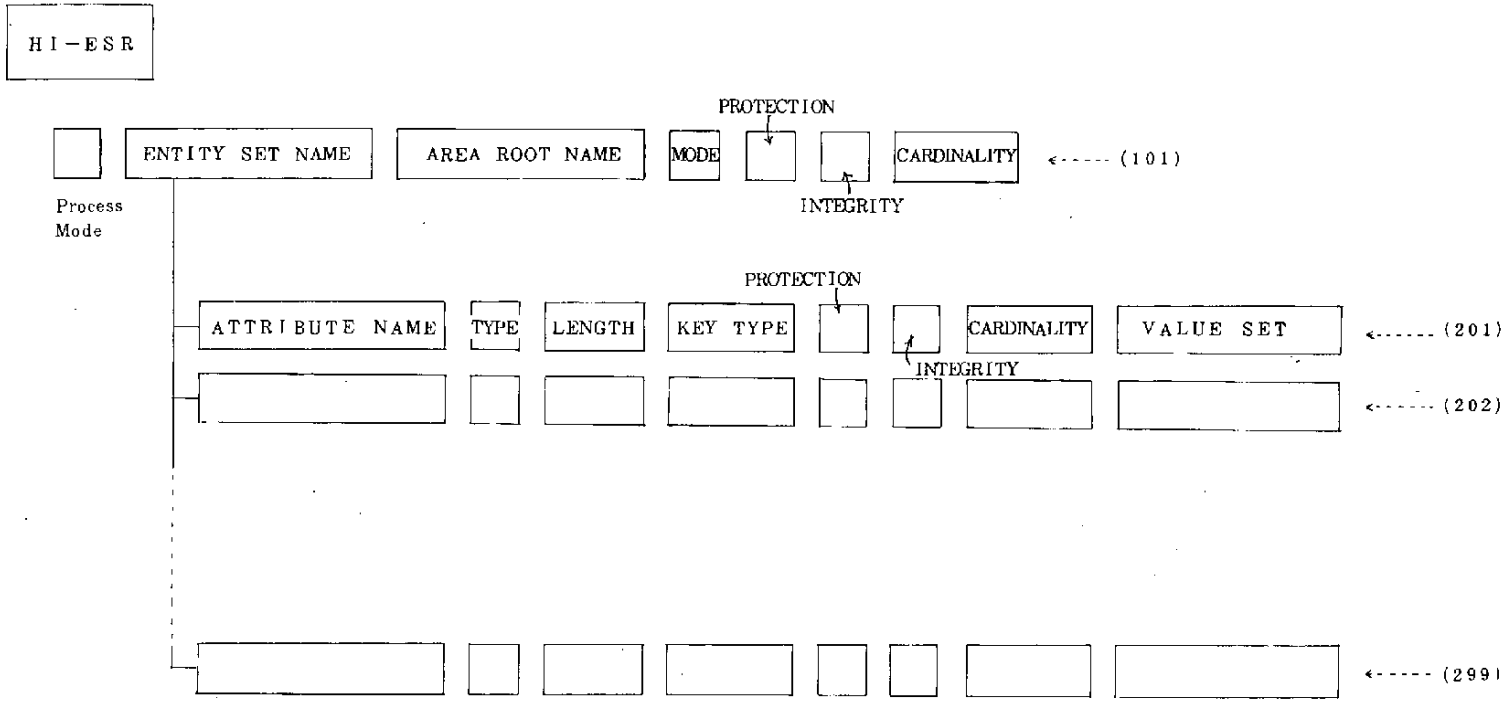
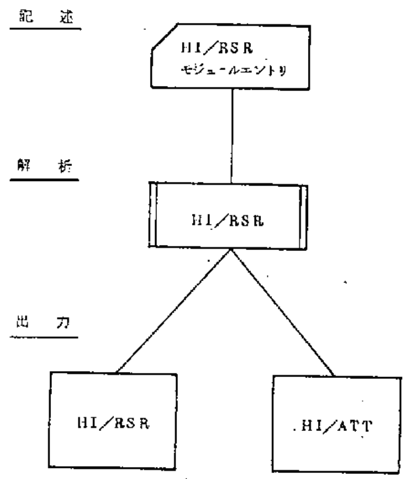


図8－6　HI／ESRモジュールエントリの構成

b）ＨＩ／ＲＳＲ　モジュールエントリ

ＨＩ／ＲＳＲモジュールエントリでは、ＬＣＳレベルでの関係性集合に関する記述を行う。解析の結果は、ＨＩ／ＲＳＲおよびＨＩ／ＡＴＴに出力される。

図８－７に、ＨＩ／ＲＳＲモジュールエントリの記述形式を、図８－８に、その構成図を示す。




```
*
*                ****************************
*                *  HI/RSR   SAVE AREA  *
*                ****************************
*
 01  HI-RSR-SAVE-AREA
     03  HI-RSR-101.                          REDEFINES   HI-RSR-SAVE-AREA.
         05  HRS-101-PROCESS-MODE           PIC  X.
         05  HRS-101-RELATIONSHIP           PIC  X(16).
         05  HRS-101-SOURCE-SS              PIC  X(16).
         05  HRS-101-DEST-SS                PIC  X(16).
         05  HRS-101-MODE                   PIC  X(02).
         05  HRS-101-RS-TYPE                PIC  X(03).
         05  HRS-101-PROTECTION             PIC  X(02).
         05  HRS-101-INTEGRITY              PIC  X(02).
         05  HRS-101-CARDINALITY            PIC  9(04).
*
*
     03  HI-RSR-201.
         05  HRS-201-PROCESS-MODE           PIC  X.
         05  HRS-201-SOURCE-SET             PIC  X(16).
         05  HRS-201-DEST-SET               PIC  X(16).
*
*
     03  HI-RSR-301                         OCCURS  99  TIMES.
         05  HRS-301-ATTRIBUTE              PIC  X(16).
         05  HRS-301-TYPE                   PIC  X.
         05  HRS-301-LENGTH                 PIC  9(03).
         05  HRS-301-KEY-TYPE               PIC  X(05).
         05  HRS-301-PROTECTION             PIC  X(02).
         05  HRS-301-INTEGRITY              PIC  X(02).
         05  HRS-301-CARDINALITY            PIC  9(04).
         05  HRS-301-VALUE-SET              PIC  X(12).
         05  HRS-301-DEFAULT                PIC  9.
*
     03  HI-RSR-401                         OCCURS  99  TIMES.
         05  HRS-401-ATTRIBUTE              PIC  X(16).
         05  HRS-401-TYPE                   PIC  X.
         05  HRS-401-LENGTH                 PIC  9(03).
         05  HRS-401-KEY-TYPE               PIC  X(05).
         05  HRS-401-PROTECTION             PIC  X(02).
         05  HRS-401-INTEGRITY              PIC  X(02).
         05  HRS-401-CARDINALITY            PIC  9(04).
         05  HRS-401-VALUE-SET              PIC  X(12).
         05  HRS-401-DEFAULT                PIC  9.
*
*
     03  HI-RSR-501                         OCCURS  99  TIMES.
         05  HRS-501-ATTRIBUTE              PIC  X(16).
         05  HRS-501-TYPE                   PIC  X.
         05  HRS-501-LENGTH                 PIC  9(03).
         05  HRS-501-KEY-TYPE               PIC  X(05).
         05  HRS-501-PROTECTION             PIC  X(02).
         05  HRS-501-INTEGRITY              PIC  X(02).
         05  HRS-501-CARDINALITY            PIC  9(04).
         05  HRS-501-VALUE-SET              PIC  X(12).
```

図８－７　ＨＩ／ＲＳＲの記述形式

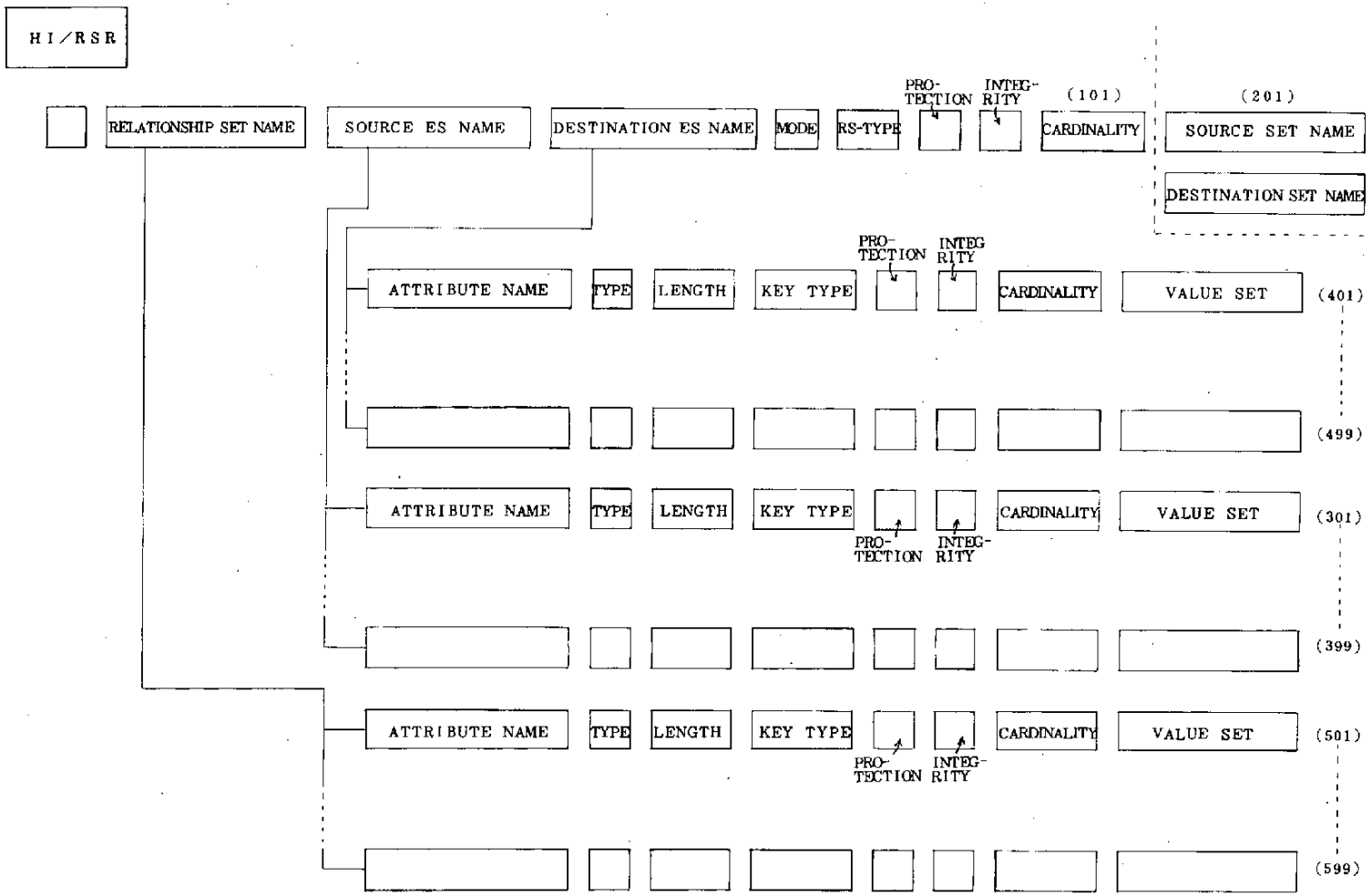
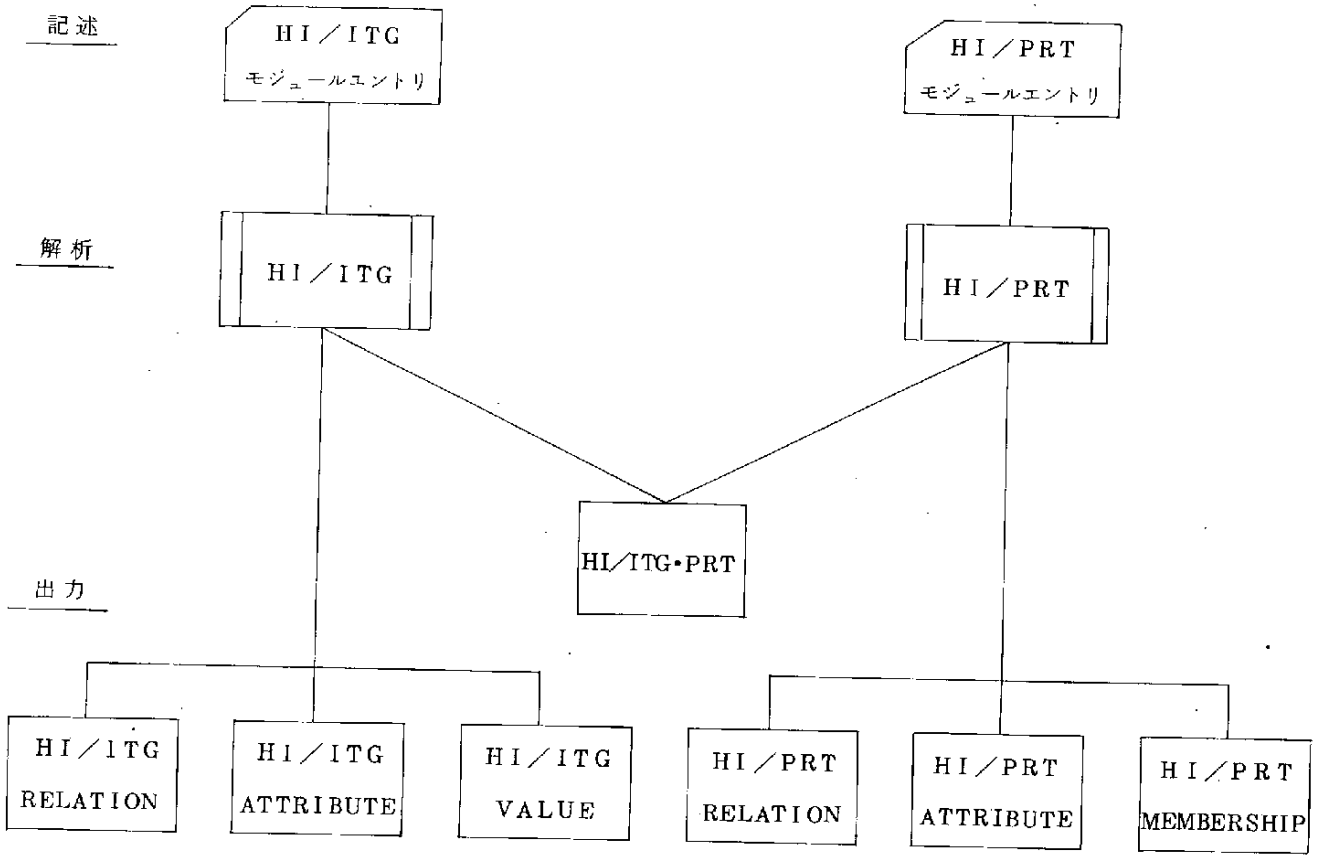


図8－8　HI／RSRモジュールエントリの構成

e) HI／ITGとHI／PRT



① ＨＩ／ＩＴＧモジュールエントリ

ＨＩ／ＩＴＧモジュールエントリでは、ＬＣＳレベルでのインテグリティ（integrity）に関する記述を行う。インテグリティ記述は、ＨＩ／ＥＳＲおよびＨＩ／ＲＳＲモジュールエントリでの記述に対応している。なお、ここでのインテグリティ記述は、データベースのスキーマで明確に定義されているものに限る。

スキーマ上で定義されていないインテグリティの記述は、ステートメント番号９０１以降で具体的に記述するようにする。例えば、項目間に何らかのインテグリティを設定する場合が考えられる。

```
*
*            **********************
*            *  HI/ITG  SAVE AREA  *
*            **********************
*
 01  HI-ITG-SAVE-AREA
                                    REDEFINES   HI-ESR-SAVE-AREA.
     03  HI-ITG-101.
         05  HIT-101-PROCESS-MODE          PIC  X.
         05  HIT-101-ES-RS-NAME            PIC  X(16).
         05  HIT-101-ES-RS-TYPE            PIC  X(03).
         05  HIT-101-KEY-TYPE              PIC  X.
         05  HIT-101-CARD-ES-RS            PIC  9(04).
         05  HIT-101-CARD-TUPLE            PIC  9(04).
*
*
     03  HI-ITG-201                 OCCURS  99 TIMES.
         05  HIT-201-PROCESS-MODE          PIC  X.
         05  HIT-201-ES-RS-NAME            PIC  X(16).
         05  HIT-201-ES-RS-TYPE            PIC  X(03).
         05  HIT-201-ATTRIBUTE             PIC  X(16).
         05  HIT-201-CARD-ATT              PIC  9(04).
         05  HIT-201-VALUE-TYPE            PIC  X.
         05  HIT-201-VALUE-TBL     OCCURS   3 TIMES.
             07  HIT-201-VALUE-KEY         PIC  9(03).
             07  HIT-201-VALUE             PIC  9(07).
*
*
     03  HI-ITG-901                 OCCURS  99 TIMES.
         05  HIT-901-PROCESS-MODE          PIC  X.
         05  HIT-901-ES-RS-NAME            PIC  X(16).
         05  HIT-901-ES-RS-TYPE            PIC  X(03).
         05  HIT-901-DETAIL                PIC  X(50).
```

図８－９　ＨＩ／ＩＴＧの記述形式

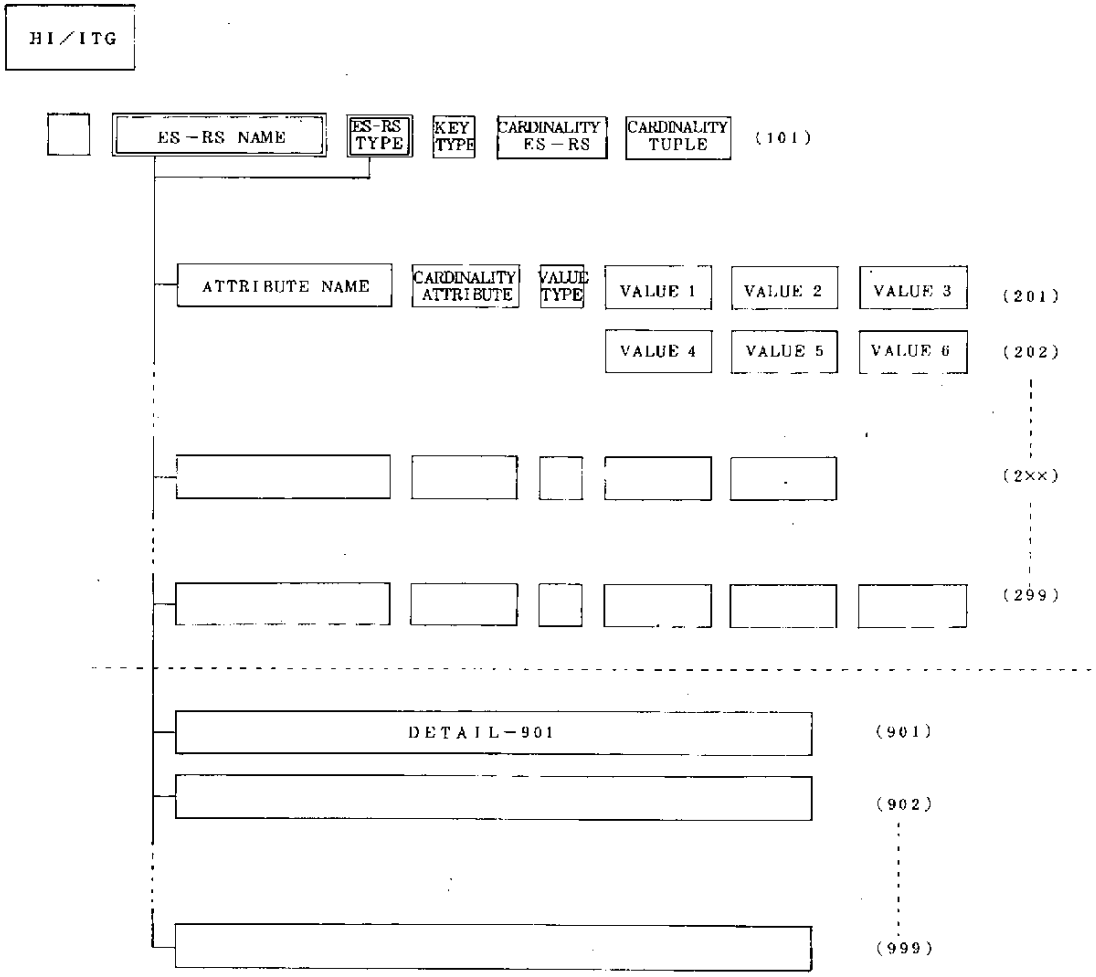


図8－10　HI／ITGモジュールエントリの構成

② ＨＩ／ＰＲＴモジュールエントリ

ＨＩ／ＰＲＴモジュールエントリでは、ＬＣＳレベルでのプロテクション（protection）に関する記述を行う。プロテクション記述は、ＨＩ／ＥＳＲおよびＨＩ／ＲＳＲモジュールエントリでの記述に対応している。なお、ここでのプロテクション記述は、データベースのスキーマで明確に定義されているものに限る。

スキーマ上で定義されていないプロテクションの記述は、ステートメント番号９０１以降で具体的に記述するようにする。

プロテクションの記述に際しては、表８－２に示すプロテクションテーブルに基づいて行う。

```
*
*              **************************
*              *  HI-PRT  SAVE AREA   *
*              **************************
*
 01  HI-PRT-SAVE-AREA
                                        REDEFINES   HI-ESR-SAVE-AREA.
     03  HI-PRT-101                    OCCURS  99 TIMES.
         05  HPR-101-PROCESS-MODE          PIC  X.
         05  HPR-101-ES-RS-NAME            PIC  X(16).
         05  HPR-101-ES-RS-TYPE            PIC  X(03).
         05  HPR-101-MODE-KEY          OCCURS   3 TIMES.
             07  HPR-101-MODE              PIC  X(03).
             07  HPR-101-KEY               PIC  X(10).
*
*
     03  HI-PRT-201.
         05  HPR-201-PROCESS-MODE          PIC  X.
         05  HPR-201-ES-RS-NAME            PIC  X(16).
         05  HPR-201-ES-RS-TYPE            PIC  X(03).
         05  HPR-201-STORAGE-CLASS         PIC  X(09).
         05  HPR-201-REMOVAL-CLASS         PIC  X(09).
*
*
     03  HI-PRT-301                    OCCURS  99 TIMES.
         05  HPR-301-PROCESS-MODE          PIC  X.
         05  HPR-301-ES-RS-NAME            PIC  X(16).
         05  HPR-301-ES-RS-TYPE            PIC  X(03).
         05  HPR-301-ATTRIBUTE             PIC  X(16).
         05  HPR-301-MODE-KEY          OCCURS   2 TIMES.
             07  HPR-301-MODE              PIC  X(02).
             07  HPR-301-KEY               PIC  X(10).
*
*
     03  HI-PRT-901                    OCCURS  99 TIMES.
         05  HPR-901-PROCESS-MODE          PIC  X.
         05  HPR-901-ES-RS-NAME            PIC  X(16).
         05  HPR-901-ES-RS-TYPE            PIC  X(03).
         05  HPR-901-DETAIL                PIC  X(50).
```

図８－１１　ＨＩ／ＰＲＴの記述形式

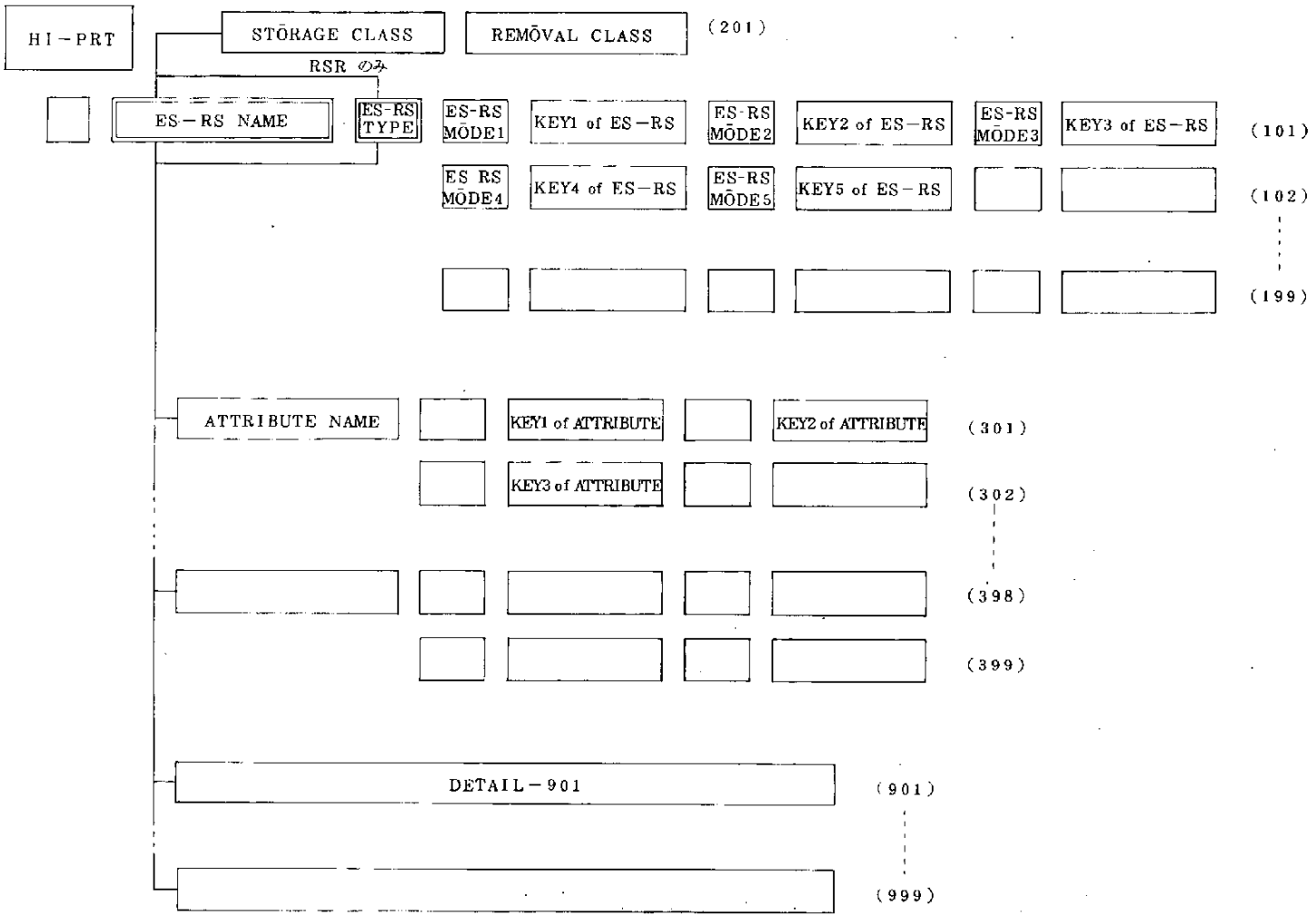


図8－12 HI／PRTモジュールエントリの構成

表８－２　プロテクションテーブル

	DBTG（網型）	IMS（階層型）	Relational（関係型）
事象集合（entity-set）	DE DELete　INSert FINd　MODify GET　REMove STOre	IE GET INSert DELete REPlace	RE RETrieve APPend DELete REPlace
関係性集合（relationship-set）	DR FINd INSert REMove ORDer	IR GET INSert DELete REPlace	RR RETrieve APPend DELete REPlace
子特性（membership）	DM storage-class / romoval-class AUTOMATIC / FIXED MANUAL / MANDATORY OPTIONAL		
属性（attribute）	DA GET MODify STOre	IA GET INSert DELete REPlace	RA RETrieve APPend DELete REPlace

※　プロテクションテーブル内の大文字部分のみが記述される。

C．出力リレーション（output relation）

エントリモジュールと出力リレーションとの対応は、図８－２に示すとおりである。特に、ＨＩ／ＩＴＧとＨＩ／ＰＲＴから出力される７個のリレーションは、ＨＩ／ＩＴＧ・ＰＲＴを中心に管理される〔図８－１３〕。

各出力リレーションの生成について以下に示す〔図８－１４～図８－２３〕。

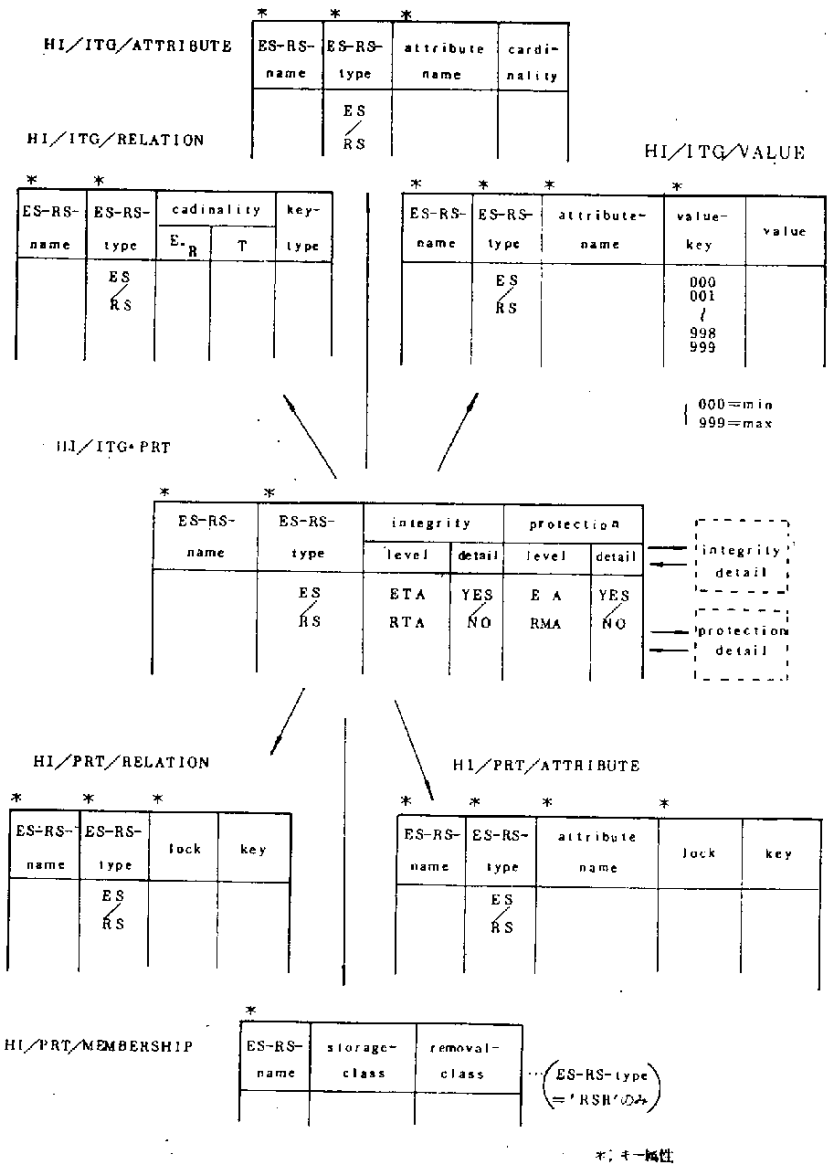


図８－１３　インテグリティとプロテクション
（integrity and protection）

アウトプット・リレーション名 HI／ESR (Entity Set Relation)	属性 長さ	モジュール・エントリ名 HI／ESR　(101)
*1 ENTITY SET NAME	X(16)	ENTITY SET NAME
*2 AREA ROOT NAME	X(16)	AREA ROOT NAME
NUMBER OF KEYS	9(04)	201～299の内でK××××（キーの種類）の数 ゼロのとき、DEGREEと同じ
MODE	X(02)	CU、CD、VS、DI、SY …… DBTG SU、SM …… IMS
DEGREE	9(02)	ATTRIBUTE NAME の総数（201～299）
WIDTH	9(04)	アトリビュート長の総計（201～299の総計）
SIZE	9(08)	WIDTH * CARDINALITY
PROTECTION	X(01)	PROTECTION指定 PY のとき Y PN のとき N
INTEGRITY	X(01)	INTEGRITY指定 IY のとき Y IN のとき N
CARDINALITY	9(04)	CARDINALITY（ゼロ以上の整数）
TID	X(08)	SPACE
FILLER	X(14)	SPACE

図8－14　出力リレーション：HI／ESR

アウトプット・リレーション名 HI／RSR	属性 長さ	モジュール・エントリ名 HI／RSR （101）	HI／RSR （201）
*1 RELATIONSHIP SET NAME	X(16)	RELATIONSHIP SET NAME	
S－ES NAME	X(16)	S－ES NAME	
D－ES NAME	X(16)	D－ES NAME	
MODE	X(02)	CU、CD、VS、DI、SY	
		SU、SM、or ブランク	
RS TYPE	X(03)	S／L／ゐ、H／N／ゐ、C／ゐの組合せ	
DEGREE	9(02)	アトリビュート数（301～、401～、501～	
		の総数；2以上の整数）	
WIDTH	9(04)	アトリビュート長の総計（301～599の総計）	
SIZE	9(08)	WIDTH * CARDINALITY	
SOURCE SET NAME	X(16)		SOURCE SET NAME（DBTG用）
DEST SET NAME	X(16)		DEST SET NAME（DBTG用）
PROTECTION	X(01)	PROTECTION 指定 PYのときY	
		PNのときN	
INTEGRITY	X(01)	INTEGRITY 指定 IY のとき Y	
		IN のとき N	
CARDINALITY	9(04)	CARDINALITY （0以上の整数）	
TID	X(08)	SPACE	
FILLER	X(07)	SPACE	

図8－15　出力リレーション：HI／RSR

アウトプット・リレーション名 HI／ATT (ATTRIBUTE)	属性 長さ	モジュール・エントリ名 HI／ESR　(201～299)	モジュール・エントリ名 HI／RSR
*1 ES－RS NAME	X(16)	ENTITY SET NAME	RELATIONSHIP SET NAME
*2 ES－RS TYPE	X(02)	'ES'	'RS'
*3 ATTRIBUTE NO	9(02)	ESR毎の1からの連番	RSR毎の1からの連番
ATTRIBUTE NAME	X(16)	ATTRIBUTE NAME	S-ES KEY NAME(301～399) D-ES KEY NAME(401～499) NON KEY ATTRIBUTE NAME(501～599)
VALUE SET	X(12)	バリューセット名(RELATIONALモデルのとき) 空白のときATTRIBUTE NAME	同 左
ROLE OF KEY	X(05)	KEY TYPE　K0001～K9999 又は N0000	KEY TYPE　K0001～K9999 又は N0000
TYPE	X(01)	TYPE (C、D、B)	TYPE (C、D、B)
LENGTH	9(03)	LENGTH アトリビュート長(1以上の整数)	LENGTH アトリビュート長(1以上の整数)
PROTECTION	X(01)	PROTECTION 指定 PY のとき Y PN のとき N	同 左
INTEGRITY	X(01)	INTEGRITY 指定 IY のとき Y IN のとき N	同 左
CARDINALITY	9(04)	CARDINALITY(0以上の整数)	CARDINALITY (0以上の整数)
TID	X(08)	SPACE	SPACE
SELECTIVITY	9(03)	(1／CARDINALITY)＊ 100	(1／CARDINALITY)＊ 100
FILLER	X(06)	SPACE	SPACE

301～399
401～499では省略されることがある。

図8－16　出力リレーション：HI／ATT

アウトプット・リレーション名 HI／ITG・PRT	属性 長さ	モジュール・エントリ名 HI／ITG　(101)	モジュール・エントリ名 HI／PRT
*1 ES－RS NAME	X(16)	ES－RS NAME(ENTITY SET又は RELATIONSHIP SET NAME)	
*2 ES－RS TYPE	X(02)	ES－RS TYPE ESR のとき 'ES' RSR のとき 'RS'	
INTEGRITY LEVEL	X(03)	<1col.> KEY TYPEキ6又はCARDINALITY of ES－RSキ6で ES－RS TYPE が ESR のとき 'E' RSR のとき 'R' 以外は空白	{E, R, 空白}
		<2col.> CARDINALITY－TUPLEキ6なら 'T' 以外は空白	{T, 空白}
		<3col.> 201～の指定あり 'A' 以外は空白	{A, 空白}
INTEGRITY DETAIL	X(04)	901～の指定あり 'Y△△△' なし 'N△△△'	
PROTECTION LEVEL	X(03)		<col.1> 101～の指定ありで ES-RS TYPEがESRのとき 'E' {E, R, 空白} RSRのとき 'R' 以外は空白 <col.2> 201の指定あり 'M'(RSRのみ) {M, 空白} 以外は空白 <col.3> 301～の指定あり 'A' {A, 空白} 以外は空白
PROTECTION DETAIL	X(04)		901～の指定あり 'Y△△△' なし 'N△△△'
TID	X(08)	SPACE	
FILLER	X(20)	SPACE	

図8－17　出力リレーション：HI／ITG・PRT

アウトプット・リレーション名 HI／ITG／RELATION	属性 長さ	モジュール・エントリ名 HI／ITG　（101）
*1 ES－RS NAME	X(16)	ES－RS NAME （エンタティ・セット名又はリレーションシップ・セット名）
*2 ES－RS TYPE	X(02)	ES－RS TYPE　ESR のとき　'ES' RSR のとき　'RS'
CARDINALITY ES－RS	9(04)	CARDINALITY ES－RS （0以上の整数） （エンタティ・セットもしくはリレーションシップ・セットレベルでのカーディナリティ）
CARDINALITY TUPLE	9(04)	CARDINALITY TUPLE （0以上の整数） （TUPLE レベルでのカーディナリティ）
KEY TYPE	X(01)	KEY TYPE （U、M、Nのいづれか）
TID	X(08)	
FILLER	X(25)	

図8－18　出力リレーション：HI／ITG／RELATION

アウトプット・リレーション名 HI／ITG／ATTRIBUTE	属性 長さ	モジュール・エントリ名 HI／ITG　（201及び継続行202～）
*1 ES－RS NAME	X(16)	ES－RS NAME
*2 ES－RS TYPE	X(02)	ES－RS TYPE　ESR のとき　'ES' RSR のとき　'RS'
*3 ATTRIBUTE NAME	X(16)	ATTRIBUTE NAME
CARDINALITY ATTRIBUTE	9(04)	CARDINALITY ATTRIBUTE（0以上の整数）
TID	X(08)	SPACE
FILLER	X(14)	SPACE

図8－19　出力リレーション：HI／ITG／ATTRIBUTE

アウトプット・リレーション名 HI／ITG／VALUE	属性 長さ	モジュール・エントリ名 HI／ITG （201～299）
*1 ES－RS NAME	X(16)	ES－RS NAME
*2 ES－RS TYPE	X(02)	ES－RS TYPE が ESR のとき 'ES' RSR のとき 'RS'
*3 ATTRIBUTE NAME	X(16)	ATTRIBUTE NAME
*4 VALUE KEY	9(03)	VALUE TYPE ＝ R のとき 000（最小値）及び999（最大値） 〃 ＝ E のとき 001～998（数え上げの数値毎に1 up.） 〃 ＝ 右 のとき このリレーションに出力しない。
VALUE	9(07)	0以上の整数（VALUE KEY に対応した数値）
TID	X(08)	SPACE
FILLER	X(08)	SPACE

図8－20 出力リレーション：HI／ITG／VALUE

アウトプット・リレーション名 HI／PRT／RELATION	属性 長さ	モジュール・エントリ名 HI／PRT （101～）
*1 ES－RS NAME	X(16)	ES－RS NAME
*2 ES－RS TYPE	X(02)	ES－RS TYPE が ESR のとき 'ES' RSR のとき 'RS'
*3 LOCK	X(03)	ES－RS MODE－(n) ｝（1枚当りMax. 3件出力）
KEY	X(10)	KEY(n) of ES－RS ｝（1枚当りMax. 3件出力）
TID	X(08)	SPACE
FILLER	X(21)	SPACE

図8－21 出力リレーション：HI／PRT／RELATION

アウトプット・リレーション名 HI／PRT／ATTRIBUTE	属性 長さ	モジュール・エントリ名 HI／PRT　（301～）
*1 ES−RS NAME	X(16)	ES−RS NAME
*2 ES−RS TYPE	X(02)	ES−RS TYPE が ESR のとき 'ES' RSR のとき 'RS'
*3 ATTRIBUTE NAME	X(16)	ATTRIBUTE NAME
*4 LOCK	X(03)	ATTRIBUTE MODE (n) } n回目の出力
KEY	X(10)	KEY(n) of ATTRIBUTE } n回目の出力
TID	X(08)	SPACE
FILLER	X(05)	SPACE

図8−22　出力リレーション：HI／PRT／ATTRIBUTE

アウトプット・リレーション名 HI／PRT／MEMBERSHIP	属性 長さ	モジュール・エントリ名 HI／PRT　（201）
*1 ES−RS NAME	X(16)	ES−RS NAME
STORAGE CLASS	X(09)	STORAGE CLASS（DBTG型メンバーシップのうち STORAGE CLASSを指定）　AUTOMATIC／MANUAL
REMOVAL CLASS	X(09)	REMOVAL CLASS（DBTG型メンバーシップのうち REMOVAL CASSを指定）　FIXED／MANDATORY／OPTIONAL
TID	X(08)	
FILLER	X(18)	

図8−23　出力リレーション：HI／PRT／MEMBERSHIP

```
*
*              ***************************
*              *  OUTPUT RELATION NAME   *
*              ***************************
*
***************
*    HI/ESR    *
*************** <RL=80>
*
 01  HI-ESR-PUT.
     03  HI-ESR-ENTITY-NAME              PIC  X(16).
     03  HI-ESR-AREA-ROOT                PIC  X(16).
     03  HI-ESR-NO-OF-KEYS               PIC  9(04).
     03  HI-ESR-MODE                     PIC  X(02).
     03  HI-ESR-DEGREE                   PIC  9(02).
     03  HI-ESR-WIDTH                    PIC  9(04).
     03  HI-ESR-SIZE                     PIC  9(08).
     03  HI-ESR-PROTECTION               PIC  X.
     03  HI-ESR-INTEGRITY                PIC  X.
     03  HI-ESR-CARDINALITY              PIC  9(04).
     03  HI-ESR-TID                      PIC  X(08).
     03  HI-ESR-FILLER                   PIC  X(14).
*
***************
*    HI/ATT    *
*************** <RL=80>
*
 01  HI-ATT-PUT.
     03  HI-ATT-ES-RS-NAME               PIC  X(16).
     03  HI-ATT-ES-RS-TYPE               PIC  X(02).
     03  HI-ATT-ATTNO                    PIC  9(02).
     03  HI-ATT-ATTRIBUTE                PIC  X(16).
     03  HI-ATT-VALUE-SET                PIC  X(12).
     03  HI-ATT-ROLE                     PIC  X(05).
     03  HI-ATT-TYPE                     PIC  X.
     03  HI-ATT-LENGTH                   PIC  9(03).
     03  HI-ATT-PROTECTION               PIC  X.
     03  HI-ATT-INTEGRITY                PIC  X.
     03  HI-ATT-CARDINALITY              PIC  9(04).
     03  HI-ATT-TID                      PIC  X(08).
     03  HI-ATT-SELECTIVITY              PIC  9(03).
     03  HI-ATT-FILLER                   PIC  X(06).
*
***************
*    HI/RSR    *
*************** <RL=120>
*
 01  HI-RSR-PUT.
     03  HI-RSR-RELATIONSHIP             PIC  X(16).
     03  HI-RSR-SOURCE-ES                PIC  X(16).
     03  HI-RSR-DEST-ES                  PIC  X(16).
     03  HI-RSR-MODE                     PIC  X(02).
     03  HI-RSR-RS-TYPE.
         05  HI-RSR-RS-T1                PIC  X.
         05  HI-RSR-RS-T2                PIC  X.
         05  HI-RSR-RS-T3                PIC  X.
     03  HI-RSR-DEGREE                   PIC  9(02).
     03  HI-RSR-WIDTH                    PIC  9(04).
     03  HI-RSR-SIZE                     PIC  9(08).
     03  HI-RSR-SOURCE-SET               PIC  X(16).
     03  HI-RSR-DEST-SET                 PIC  X(16).
     03  HI-RSR-PROTECTION               PIC  X.
     03  HI-RSR-INTEGRITY                PIC  X.
     03  HI-RSR-CARDINALITY              PIC  9(04).
     03  HI-RSR-TID                      PIC  X(08).
     03  HI-RSR-FILLER                   PIC  X(07).
```

図８－２４　出力リレーション形式(1)

```
*
*****************
*    HI/ITG.PRT   *
****************** (RL=60)
*
 01  HI-ITG-PRT-PUT.
     03  ITG-PRT-ES-RS-NAME              PIC  X(16).
     03  ITG-PRT-ES-RS-TYPE              PIC  X(02).
     03  ITG-PRT-ITG-LEVEL.
         05  ITG-PRT-ITG-L1              PIC  X.
         05  ITG-PRT-ITG-L2              PIC  X.
         05  ITG-PRT-ITG-L3              PIC  X.
     03  ITG-PRT-ITG-DETAIL              PIC  X(04).
     03  ITG-PRT-PRT-LEVEL.
         05  ITG-PRT-PRT-L1              PIC  X.
         05  ITG-PRT-PRT-L2              PIC  X.
         05  ITG-PRT-PRT-L3              PIC  X.
     03  ITG-PRT-PRT-DETAIL              PIC  X(04).
     03  ITG-PRT-TID                     PIC  X(08).
     03  ITG-PRT-FILLER                  PIC  X(20).
*
*********************
*    HI/ITG.RELATION  *
********************* (RL=60)
*
 01  HI-ITG-REL-PUT.
     03  ITG-REL-ES-RS-NAME              PIC  X(16).
     03  ITG-REL-ES-RS-TYPE              PIC  X(02).
     03  ITG-REL-CDNL-ES-RS              PIC  9(04).
     03  ITG-REL-CDNL-TUPLE              PIC  9(04).
     03  ITG-REL-KEY-TYPE                PIC  X.
     03  ITG-REL-TID                     PIC  X(08).
     03  ITG-REL-FILLER                  PIC  X(25).
*
***********************
*    HI/ITG.ATTRIBUTE  *
*********************** (RL=60)
*
 01  HI-ITG-ATT-PUT.
     03  ITG-ATT-ES-RS-NAME              PIC  X(16).
     03  ITG-ATT-ES-RS-TYPE              PIC  X(02).
     03  ITG-ATT-ATTRIBUTE               PIC  X(16).
     03  ITG-ATT-CARDINALITY             PIC  9(04).
     03  ITG-ATT-TID                     PIC  X(08).
     03  ITG-ATT-FILLER                  PIC  X(14).
*
*********************
*    HI/ITG.VALUE   *
********************* (RL=60)
*
 01  HI-ITG-VAL-PUT.
     03  ITG-VAL-ES-RS-NAME              PIC  X(16).
     03  ITG-VAL-ES-RS-TYPE              PIC  X(02).
     03  ITG-VAL-ATTRIBUTE               PIC  X(16).
     03  ITG-VAL-VALUE-KEY               PIC  9(03).
     03  ITG-VAL-VALUE                   PIC  9(07).
     03  ITG-VAL-TID                     PIC  X(08).
     03  ITG-VAL-FILLER                  PIC  X(08).
```

図８－２５　出力リレーション形式(2)

```
*
**************************
*    HI/PRT.RELATION     *
************************** <RL=60>
*
 01  HI-PRT-REL-PUT.
     03  PRT-REL-ES-RS-NAME          PIC  X(16).
     03  PRT-REL-ES-RS-TYPE          PIC  X(02).
     03  PRT-REL-LOCK                PIC  X(03).
     03  PRT-REL-KEY                 PIC  X(10).
     03  PRT-REL-TID                 PIC  X(08).
     03  PRT-REL-FILLER              PIC  X(21).
*
**************************
*    HI/PRT.MEMBERSHIP   *
************************** <RL=60>
*
 01  HI-PRT-MEM-PUT.
     03  PRT-MEM-ES-RS-NAME          PIC  X(16).
     03  PRT-MEM-STORAGE-CLASS       PIC  X(09).
     03  PRT-MEM-REMOVAL-CLASS       PIC  X(09).
     03  PRT-MEM-TID                 PIC  X(08).
     03  PRT-MEM-FILLER              PIC  X(16).
*
**************************
*    HI/PRT.ATTRIBUTE    *
************************** <RL=60>
*
 01  HI-PRT-ATT-PUT.
     03  PRT-ATT-ES-RS-NAME          PIC  X(16).
     03  PRT-ATT-ES-RS-TYPE          PIC  X(02).
     03  PRT-ATT-ATTRIBUTE           PIC  X(16).
     03  PRT-ATT-LOCK                PIC  X(03).
     03  PRT-ATT-KEY                 PIC  X(10).
     03  PRT-ATT-TID                 PIC  X(08).
     03  PRT-ATT-FILLER              PIC  X(05).
```

図８－２６　出力リレーション形式(3)

8.1.2 リレーショナル記述

ＤＢエントリ及びモジュールエントリで記述されたＬＣＳ／ＨＩ情報は、NDD/HIPSサブシステムにより解析され、ＮＤＤのＨＩ情報として新たに生成される。リレーショナル記述、即ち、ＲＤとは事象集合や関係性集合の記述をユーザもしくは問合せ(query)がみるときの視点(view)であり、リレーションの一種である。本サブシステムでは、生成した各リレーションに対して、下記の形式に基づいた問合せ機能を有している。指定項目は、対象リレーション名、リレーションタイプ、編集項目名等である。

形式：

$$\mathrm{HI/RELATION}\left(\left\{\begin{matrix}\mathrm{REL-TYPE}\\ \mathrm{TYPE}\end{matrix}\right\}=\left\{\begin{matrix}\mathrm{ES}\\ \mathrm{RS}\end{matrix}\right\},\ \text{schema-name},\text{database-name},\text{host-id}\right)$$

$$\left(\left\{\begin{matrix}[\,\text{rel-name}[\,,\text{rel-name}\,]\cdots\,]\\ \mathrm{REL}\\ \mathrm{RELATION}\end{matrix}\right\}[\,,\left\{\begin{matrix}\mathrm{ATT}\\ \mathrm{ATTRIBUTE}\end{matrix}\right\}[\,,\left\{\begin{matrix}\mathrm{ROL}\\ \mathrm{ROLE}\end{matrix}\right\}\,]\,]\right)$$

解説：

- $\left\{\begin{matrix}\mathrm{REL-TYPE}\\ \mathrm{TYPE}\end{matrix}\right\}=\left\{\begin{matrix}\mathrm{ES}\\ \mathrm{RS}\end{matrix}\right\}$

  編集するＲＤのタイプを指定する。

- schema-name

  編集対象のＥＳ／ＲＳが記述されているスキーマ名を指定する。

- database-name

  編集対象のスキーマに相当するデータベース名を指定する。

- host-id

  編集対象のデータベースが格納されているホストＩＤを指定する。

- $\left\{\begin{matrix}[\,\text{rel-name}[\,,\text{relname}\,]\cdots\,]\\ \mathrm{REL}\\ \mathrm{RELATION}\end{matrix}\right\}$

  編集するリレーションを具体的に記述する。データベース全部を対象にする場合は、ＲＥＬ又はＲＥＬＡＴＩＯＮと指定する。

- $\left\{\begin{matrix}\mathrm{ATT}\\ \mathrm{ATTRIBUTE}\end{matrix}\right\}$

  リレーション名とともに、属性名も編集する場合は、ＡＴＴ又はＡＴＴＲＩＢＵＴＥと指定する。

- $\left\{\begin{matrix}\mathrm{ROL}\\ \mathrm{ROLE}\end{matrix}\right\}$

さらに、各属性の役割について編集する場合は、ＲＯＬＥまたはＲＯＬＥと指定する。この場合、先行するＡＴＴＲＩＢＵＴＥ指定の省略は許されない。

次の例は、ＡＣＯＳ(host-id)にあるＰＲＯＪ(database-name)のＰＲＯＪＥＣＴ(schema-name)について、全ての関係性集合（ＲＥＬ－ＴＹＰＥ＝ＲＳ）を求めている。また、合わせてその属性構成と役割についても要求している。

```
JIPDEC  NDD/HIPS    ( RELATIONAL DESCRIPTION )          DATE 03/13/80      PAGE    1

PARAMETER CARD LIST :    HI/ATT    ( TYPE = RS, PROJECT , PROJ ,   ACOS  )
                                   ( RELATION, ATT, ROLE )



JIPDEC  NDD/HIPS    ( RELATIONAL DESCRIPTION )          DATE 03/13/80      PAGE    2

RD/RSR : SCHEMA NAME = PROJECT          DATABASE NAME = PROJ          MODEL =          HOST = ACOS

    RELATIONSHIP SET    ATTRIBUTE NAME     ROLE OF KEY    ATTRIBUTE NAME     ROLE OF KEY
                        (DOMAIN)                          (DOMAIN)

    REPR-PROJ           REPR-NO            K0101          PROJ-NO            K0201
                        (REPR-NO    )                     (PROJ-NO    )

    PROJ-SPON-LINK      SPONNAME           K0101          PROJ-NO            K0201
                        (SPONNAME   )                     (PROJ-NO    )

                        BUDGET
                        (BUDGET     )

    PROJ-KEYW-LINK      KEYWORD            K0101          PROJ-NO            K0201
                        (KEYWORD    )                     (PROJ-NO    )

                        DUMMY
                        (DUMMY      )

    PROJ-LINK           PROJ-NO                           PRIOR-NO
                        (PROJ-NO    )                     (PRIOR-NO   )

                        POST-NO
                        (POST-NO    )

    REPR-ENGI           REPR-NO            K0101          ENGINAME           K0201
                        (REPR-NO    )                     (ENGINAME   )

    PROJ-ENGI           PROJ-NO            K0101          ENGINAME           K0201
                        (PROJ-NO    )                     (ENGINAME   )
```

図８－２７　ＲＤ指定と表示の例

8.1.3 例 題

本項では、ＤＢＴＧデータモデルに基づいて記述したスキーマ（ＬＣＳ）を採り上げ、ＮＤＤ／ＨＩＰＳサブシステムによりＨＩ情報が生成される過程を示す。引用するデータベースは、プロジェクト・データベース（ＰＲＤＢ）である。ＰＲＤＢのスキーマ・データ構造図を、図８－２９に、Ｅ－Ｒモデルに基づいたＥ－Ｒ図を図８－２８に示す。

また、「８.１.１　ＬＣＳ／ＨＩ記述」に基づいて記述した、例題のＬＣＳ／ＨＩ記述を図８－30に示す。

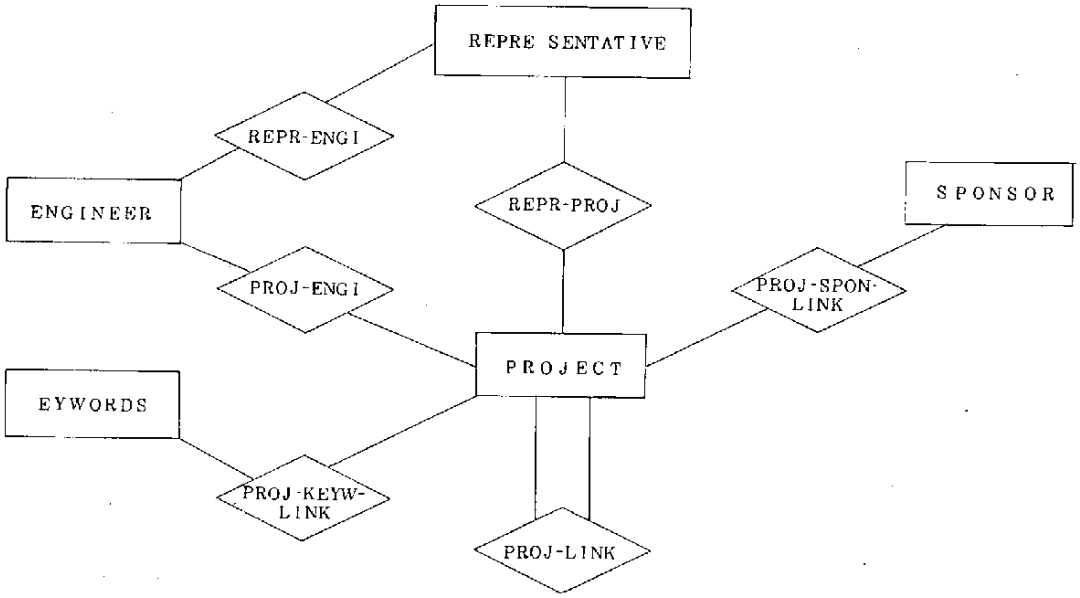


図８－２８　プロジェクト・データベース：ＬＣＳ（Ｅ－Ｒ図）

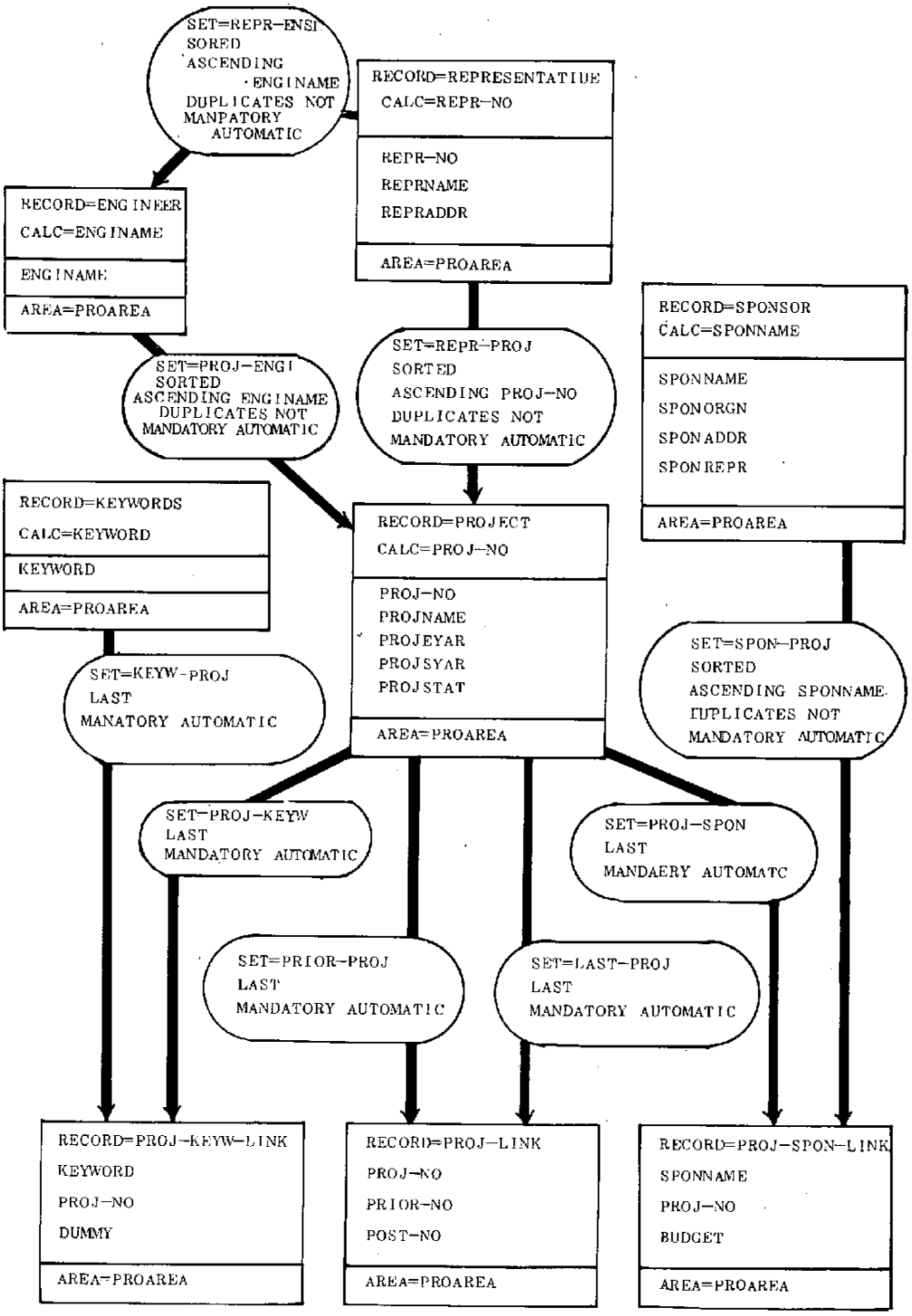


図8－29　PRDBのスキーマデータ構造図

```
##DB/PROJECT/PROJ/DSTG/ACDS/C
ESRC REPRESENTATIVE   101 PROAREA          CD PN IN 0010
ESRC REPRESENTATIVE   201 REPR-NO          C 020 K0001 PN IN 0010
ESRC REPRESENTATIVE   202 REPRNAME         C 020       PN IN 0010
ESRC REPRESENTATIVE   203 REPRADDR         C 020       PN IN 0005
ESRC PROJECT          101 PROAREA          CD PN IN 0050
ESRC PROJECT          201 PROJ-NO          C 020 K0001 PN IN 0050
ESRC PROJECT          202 PROJNAME         C 020       PN IN 0050
ESRC PROJECT          203 PROJEYAR         D 006       PN IN 0005
ESRC PROJECT          204 PROJSYAR         D 006       PN IN 0010
ESRC PROJECT          205 PROJSTAT         D 002       PN IN 0002
ESRC SPONSOR          101 PROAREA          CD PN IN 0010
ESRC SPONSOR          201 SPONNAME         C 020 K0001 PN IN 0010
ESRC SPONSOR          202 SPONORGN         C 020       PN IN 0010
ESRC SPONSOR          203 SPONADDR         C 020       PN IN 0005
ESRC SPONSOR          204 SPONREPR         C 020       PN IN 0004
ESRC KEYWORDS         101 PROAREA          CD PN IN 0020
ESRC KEYWORDS         201 KEYWORD          C 020 K0001 PN IN 0005
ESRC ENGINEER         101 PROAREA          CD PN IN 0020
ESRC ENGINEER         201 ENGINAME         C 020 K0001 PN IN 0010
RSRC REPR-PROJ        101 REPRESENTATIVE   PROJECT       VS S  PN IN 0005
RSRC REPR-PROJ        301 REPR-NO          C 020 K0101 PN IN 0005
RSRC REPR-PROJ        401 PROJ-NO          C 020 K0201 PN IN 0005
RSRC PROJ-SPON-LINK   101 SPONSOR          PROJECT       VS LN PN IN 0006
RSRC PROJ-SPON-LINK   201 SPON-PROJ        PROJ-SPON
RSRC PROJ-SPON-LINK   301 SPONNAME         C 020 K0101 PN IN 0006
RSRC PROJ-SPON-LINK   401 PROJ-NO          C 020 K0201 PN IN 0006
RSRC PROJ-SPON-LINK   501 BUDGET           D 010       PN IN 0006
RSRC PROJ-KEYW-LINK   101 KEYWORDS         PROJECT       VS LN PN IN 0010
RSRC PROJ-KEYW-LINK   201 KEYW-PROJ        PROJ-KEYW
RSRC PROJ-KEYW-LINK   301 KEYWORD          C 020 K0101 PN IN 0005
RSRC PROJ-KEYW-LINK   401 PROJ-NO          C 020 K0201 PN IN 0005
RSRC PROJ-KEYW-LINK   501 DUMMY            C 010       PN IN 0001
RSRC PROJ-LINK        101 PROJECT          PROJECT       VS LNC PN IN 0004
RSRC PROJ-LINK        201 PRIOR-PROJ       LAST-PROJ
RSRC PROJ-LINK        301 PROJ-NO          C 020       PN IN 0004
RSRC PROJ-LINK        401 PRIOR-NO         C 020       PN IN 0002
RSRC PROJ-LINK        501 POST-NO          C 020       PN IN 0002
RSRC REPR-ENGI        101 REPRESENTATIVE   ENGINEER      VS S  PN IN 0005
RSRC REPR-ENGI        301 REPR-NO          C 020 K0101 PN IN 0005
RSRC REPR-ENGI        401 ENGINAME         C 020 K0201 PN IN 0005
RSRC PROJ-ENGI        101 PROJECT          ENGINEER      VS S  PN IN 0005
RSRC PROJ-ENGI        301 PROJ-NO          C 020 K0101 PN IN 0005
RSRC PROJ-ENGI        401 ENGINAME         C 020 K0201 PN IN 0005
PRTC PROJECT          101 ESR DEL JIPDEC       REM JIPDEC
PRTC PROJECT          301 ESR PROJ-NO          MOD JIPDEC
PRTC REPR-PROJ        101 RSR FIN JIPDEC
PRTC REPR-PROJ        201 RSR AUTOMATIC MANDATORY
PRTC PROJ-SPON-LINK   101 RSR FIN JIPDEC
PRTC PROJ-SPON-LINK   201 RSR AUTOMATIC MANDATORY
PRTC PROJ-SPON-LINK   301 RSR SPONNAME         MOD JIPDEC
PRTC PROJ-SPON-LINK   302 RSR PROJ-NO          MOD JIPDEC
PRTC PROJ-SPON-LINK   901 RSR THIS RECORD TYPE MUST BE HANDLED BY DBA.
PRTC SPONSOR          101 ESR DEL JIPDEC       REM JIPDEC
PRTC SPONSOR          102 ESR INS JIPDEC
PRTC SPONSOR          201 ESR AUTOMATIC MANDATORY
PRTC SPONSOR          301 ESR SPONNAME         MOD JIPDEC    GET JIPDEC    C
PRTC SPONSOR          302 ESR                  STO JIPDEC
PRTC REPRESENTATIVE   101 ESR DEL JIPDEC
ITGC PROJECT          101 ESR M 0100 0100
ITGC PROJECT          201 ESR PROJ-NO          0100 R 0000001 0000100
ITGC PROJECT          901 ESR A PROJ-NO CORRESPONDS TO ONE OR MORE PROJECT.
ITGC REPRESENTATIVE   101 ESR U 0100 0100
ITGC REPRESENTATIVE   201 ESR REPR-NO          0100 E 0000004 0000005 0000020  C
ITGC REPRESENTATIVE   202 ESR                                 0000100
ITGC REPRESENTATIVE   901 ESR A REPRESENTATIVE MUST LIVE IN TOKYO.
ITGC KEYWORDS         101 ESR M 0100 0100
ITGC SPONSOR          101 ESR U 0050 0050
ITGC SPONSOR          201 ESR PROJ-NO          0100 R 0000001 0000100
##END
```

図8－30　LCS／HIの記述例

図８－３１以降は、ＮＤＤ／ＨＩＰＳにより生成されたＨＩ情報の各リレーションの内容を示したものである。また、図８－４２に本サブシステムが生成したリレーションのディレクトリを示す。

JIPDEC NDD/HIPS (ENTITY SET RELATION) DATE 03/10/80 PAGE 1

HI/ESR : SCHEMA NAME = PROJECT DATABASE NAME = PROJ MODEL = DBTG HOST = M

ENTITY SET NAME	AREA ROOT NAME	NO. OF KEYS	MODE	DEGREE	WIDTH	SIZE	PRT	ITG	CARDINALITY	TID
REPRESENTATIVE	PROAREA	1	CD	3	60	600	N	N	10	
PROJECT	PROAREA	1	CD	5	54	2,700	N	N	50	
SPONSOR	PROAREA	1	CD	4	80	800	N	N	10	
KEYWORDS	PROAREA	1	CD	1	20	400	N	N	20	
ENGINEER	PROAREA	1	CD	1	20	400	N	N	20	

図８－３１　リレーション：ＨＩ／ＥＳＲ

JIPDEC NDD/HIPS (RELATIONSHIP SET RELATION) DATE 03/10/80 PAGE 1

HI/RSR : SCHEMA NAME = PROJECT DATABASE NAME = PROJ MODEL = DBTG HOST = M

RELATIONSHIP SET	SOURCE ES	DESTINATION ES	MODE	TYPE	DEG	WIDTH	SIZE	SOURCE SET	DESTINATION SET
REPR-PROJ	REPRESENTATIVE	PROJECT	VS	S	2	40	200		
PROJ-SPON-LINK	SPONSOR	PROJECT	VS	LN	3	50	300	SPON-PROJ	PROJ-SPON
PROJ-KEYW-LINK	KEYWORDS	PROJECT	VS	LN	3	50	500	KEYW-PROJ	PROJ-KEYW
PROJ-LINK	PROJECT	PROJECT	VS	LNC	3	60	240	PRIOR-PROJ	LAST-PROJ
REPR-ENGI	REPRESENTATIVE	ENGINEER	VS	S	2	40	200		
PROJ-ENGI	PROJECT	ENGINEER	VS	S	2	40	200		

図８－３２　リレーション：ＨＩ／ＲＳＲ

JIPDEC NDD/HIPS (ATTRIBUTE) DATE 03/10/80 PAGE 1

HI/ATT : SCHEMA NAME = PROJECT DATABASE NAME = PROJ MODEL = DBTG HOST = M

ES-RS NAME	ES/RS	ATTNO	ATTRIBUTE NAME	VALUE SET	ROLE-OF-KEY	TYPE	LENGTH	PRT	ITG	CARDINALITY	FID	SELECTIVITY
REPRESENTATIVE	ES	01	REPR-NO	REPR-NO	K0001	C	20	N	N	10		10
		02	REPRNAME	REPRNAME		C	20	N	N	10		10
		03	REPRADDR	REPRADDR		C	20	N	N	5		20
PROJECT	ES	01	PROJ-NO	PROJ-NO	K0001	C	20	N	N	50		2
		02	PROJNAME	PROJNAME		C	20	N	N	50		2
		03	PROJEYAR	PROJEYAR		D	6	N	N	5		20
		04	PROJSYAR	PROJSYAR		D	6	N	N	10		10
		05	PROJSTAT	PROJSTAT		D	2	N	N	2		50
SPONSOR	ES	01	SPONNAME	SPONNAME	K0001	C	20	N	N	10		10
		02	SPONORGN	SPONORGN		C	20	N	N	10		10
		03	SPONADDR	SPONADDR		C	20	N	N	5		20
		04	SPONREPR	SPONREPR		C	20	N	N	4		25
KEYWORDS	ES	01	KEYWORD	KEYWORD	K0001	C	20	N	N	5		20
ENGINEER	ES	01	ENGINAME	ENGINAME	K0001	C	20	N	N	10		10
REPR-PROJ	RS	01	REPR-NO	REPR-NO	K0101	C	20	N	N	5		20
		02	PROJ-NO	PROJ-NO	K0201	C	20	N	N	5		20
PROJ-SPON-LINK	RS	01	SPONNAME	SPONNAME	K0101	C	20	N	N	6		16
		02	PROJ-NO	PROJ-NO	K0201	C	20	N	N	6		16
		03	BUDGET	BUDGET		D	10	N	N	6		16

図8－33 リレーションシ：HI／ATT (1)

JIPDEC NDD/HIPS (ATTRIBUTE) DATE 03/10/80 PAGE 2

HI/ATT : SCHEMA NAME = PROJECT DATABASE NAME = PROJ MODEL = DBTG HOST = M

ES-RS NAME	ES/RS	ATTNO	ATTRUBUTE NAME	VALUE SET	ROLE-OF-KEY	TYPE	LENGTH	PRT	ITG	CARDINALITY	TID	SELECTIVITY
PROJ-KEYW-LINK	RS	01	KEYWORD	KEYWORD	K0101	C	20	N	N	5		20
		02	PROJ-NO	PROJ-NO	K0201	C	20	N	N	5		20
		03	DUMMY	DUMMY		C	10	N	N	1		100
PROJ-LINK	RS	01	PROJ-NO	PROJ-NO		C	20	N	N	4		25
		02	PRIOR-NO	PRIOR-NO		C	20	N	N	2		50
		03	POST-NO	POST-NO		C	20	N	N	2		50
REPR-ENGI	RS	01	REPR-NO	REPR-NO	K0101	C	20	N	N	5		20
		02	ENGINAME	ENGINAME	K0201	C	20	N	N	5		20
PROJ-ENGI	RS	01	PROJ-NO	PROJ-NO	K0101	C	20	N	N	5		20
		02	ENGINAME	ENGINAME	K0201	C	20	N	N	5		20

図8－34 リレーション：HI／ATT (2)

JIPDEC NDD/HIPS (INTEGRITY , PROTECTION) DATE 03/10/80 PAGE 1

HI/ITG,PRT : SCHEMA NAME = PROJECT DATABASE NAME = PROJ MODEL = DBTG HOST = M

ES-RS NAME	ES/RS	INTEGRITY LEVEL	DETAIL	PROTECTION LEVEL	DETAIL	TID	FILLER
PROJECT	ES	ETA	Y	E A	N		
REPRESENTATIVE	ES	ETA	Y	E	N		
KEYWORDS	ES	ET	N				
SPONSOR	ES	ETA	N	EMA	N		
REPR-PROJ	RS			RM	N		
PROJ-SPON-LINK	RS			RMA	Y		

図8－35 リレーション：HI／ITG・PRT

JIPDEC NDD/HIPS (HI/PRT/RELATION) DATE 03/10/80 PAGE 1

HI/PRT : SCHEMA NAME = PROJECT DATABASE NAME = PROJ MODEL = DBTG HOST = M

ES-RS NAME	ES/RS	LOCK	KEY	LOCK	KEY	LOCK	KEY
PROJECT	ES	DEL	JIPDEC	REM	JIPDEC		
REPR-PROJ	RS	FIN	JIPDEC				
PROJ-SPON-LINK	RS	FIN	JIPDEC				
SPONSOR	ES	DEL	JIPDEC	REM	JIPDEC	INS	JIPDEC
REPRESENTATIVE	ES	DEL	JIPDEC				

図8－36 リ レ ー シ ョ ン : H I ／ P R T ／ R E L A T I O N

JIPDEC NDD/HIPS (HI/PRT/MEMBERSHIP) DATE 03/10/80 PAGE 1

HI/PRT : SCHEMA NAME = PROJECT DATABASE NAME = PROJ MODEL = DBTG HOST = M

ES-RS NAME	ES/RS	STORAGE CLASS	REMOVAL CLASS	TID	FILLER
REPR-PROJ	RS	AUTOMATIC	MANDATORY		
PROJ-SPON-LINK	RS	AUTOMATIC	MANDATORY		
SPONSOR	RS	AUTOMATIC	MANDATORY		

図8－37 リ レ ー シ ョ ン : H I ／ P R T ／ M E M B E R S E I P

JIPDEC NDD/HIPS (HI/PRT/ATTRIBUTE) DATE 03/10/80 PAGE 1

HI/PRT : SCHEMA NAME = PROJECT DATABASE NAME = PROJ MODEL = DBTG HOST = M

ES-RS NAME	ES/RS	ATTRIBUTE NAME	LOCK	KEY	LOCK	KEY
PROJECT	ES	PROJ-NO	MOD	JIPDEC		
PROJ-SPON-LINK	RS	SPONNAME	MOD	JIPDEC		
PROJ-SPON-LINK	RS	PROJ-NO	MOD	JIPDEC		
SPONSOR	ES	SPONNAME	MOD	JIPDEC	GET	JIPDEC
			STO	JIPDEC		

図8－38 リレーションス：HI／PRT／ATTRIBUTE

JIPDEC NDD/HIPS (HI/ITG/RELATION) DATE 03/10/80 PAGE 1

HI/ITG : SCHEMA NAME = PROJECT DATABASE NAME = PROJ MODEL = DBTG HOST = M

ES-RS NAME	ES/RS	CARDINALITY ES-RS	CARDINALITY TUPLE	KEY TYPE	TID	FILLER
PROJECT	ES	100	100	M		
REPRESENTATIVE	ES	100	100	U		
KEYWORDS	ES	100	100	M		
SPONSOR	ES	50	50	U		

図8－39 リレーションス：HI／ITG／RELATION

JIPDEC NDD/HIPS (HI/ITG/ATTRIBUTE) DATE 03/10/80 PAGE 1

HI/ITG : SCHEMA NAME = PROJECT DATABASE NAME = PROJ MODEL = DBTG HOST = M

ES-RS-NAME	ES/RS	ATTRIBUTE NAME	CARDINALITY	TID	FILLER
PROJECT	ES	PROJ-NO	100		
REPRESENTATIVE	ES	REPR-NO	100		
SPONSOR	ES	PROJ-NO	100		

図8－40 リレーション：HI／ITG／ATTRIBUTE

JIPDEC NDD/HIPS (HI/ITG/VALUE) DATE 03/10/80 PAGE 1

HI/ITG : SCHEMA NAME = PROJECT DATABASE NAME = PROJ MODEL = DBTG HOST = M

ES-RS-NAME	ES/RS	ATTRIBUTE NAME	VALUE KEY	VALUE	VALUE KEY	VALUE	VALUE KEY	VALUE
PROJECT	ES	PROJ-NO	000	1	999	100		
REPRESENTATIVE	ES	REPR-NO	001	4	002	5	003	20
			004	100				
SPONSOR	ES	PROJ-NO	000	1	999	100		

図8－41 リレーション：HI／ITG／VALUE

JIPDEC NDD/HIPS (DIRECTORY OF LCS/HI) DATE 03/10/80 PAGE 1

SCHEMA NAME	DATABASE NAME	MODEL	HOST	ESR	ATT	RSR	ITG REL	ITG ATT	ITG VAL	PRT REL	PRT ATT	PRT MEM
PROJECT	PROJ	DBTG	ACOS	YES	YES	YES	YES	YES	YES	YES	YES	YES

図8－42 LCS／HIディレクトリ

8.2 ＱＴＰ （Query Translation Processor）

ＱＴＰ（query translation processor）は、問合せ変換プログラムであり、ＱＵＥＬによって書かれたＬＣＳ問合せ（ＬＣＳ query or ＬＱ）を入力として、ＤＢＴＧ ＤＭＬ プログラムを出力するものである〔図８－４３〕。本章では、第５章で論じた問合せ変換アルゴリズムに基づいたＱＴＰの概要について述べる。また、ＱＴＰが必要とする変換情報は、ＨＩＰＳ〔 8.1を参照〕で生成された異種性情報（ＨＩ）を用いる。

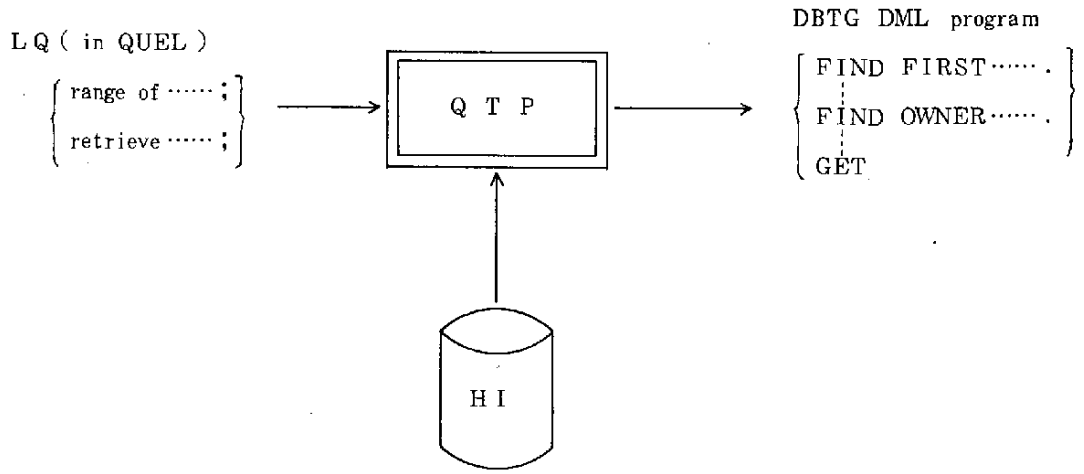


図８－４３ ＱＴＰの概要

8.2.1　ＱＴＰの構成

ＱＴＰは大きく分けて、４つのモジュールから成る〔図８－４４〕。

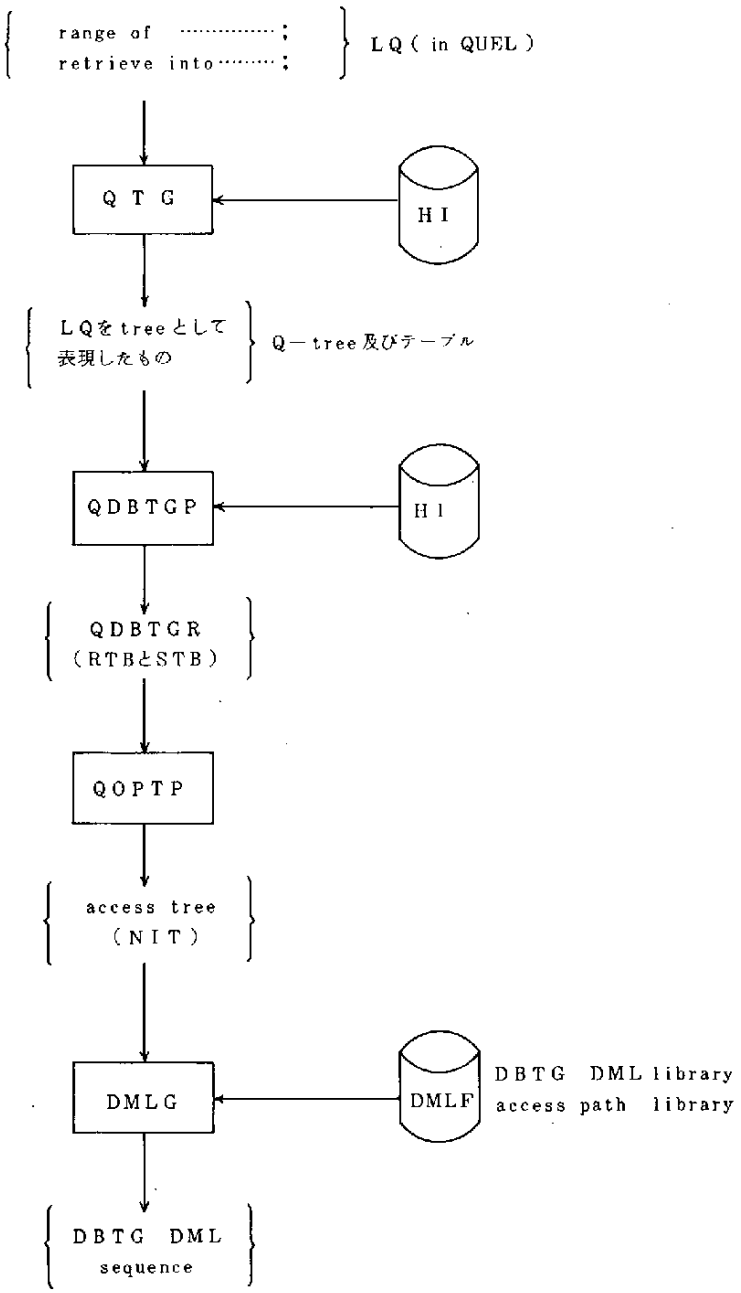


図８－４４　ＱＴＰの構成

ＱＴＧ（Ｑ－tree generator）は、ＱＵＥＬによって書かれた問合せ（ＬＱ）の文字列を入力として、対応する内部表現としての２分木（Ｑ－tree）及び、３つのテーブルを生成するものである。

ＱＤＢＴＧＰは、木構造で表わされた問合せから、ＤＢＴＧモデルの構文構造に基づいた問合せ表現（ＤＢＴＧＲ）を生成するプロセスである。ＤＢＴＧＲは、ＳＴＢ及びＲＴＢと呼ぶ、２つのテーブルからなる。ＤＢＴＧＲは、5.3.3で述べたＤＢＴＧ問合せグラフ（ＤＱＧ）の内部表現である。

ＱＯＰＴＰは、ＱＤＢＴＧＲから最適なアクセス・パスを決定するプロセスである。アクセスのシーケンスは、ＮＩＴと呼ばれる領域に木構造として表わされる。但し、頻繁に用いられるアクセス・パターンについては、ＮＩＴに必要な情報をセットしてから、そのパターン番号を次のＤＭＬＧプロセスへ通知する。アクセス・パスの決定には、２つのアルゴリズムが組込まれている。一つは、ＢＦＡ（breadth－first algorithm）であり、もう一つは、ＤＦＡ（depth－first algorithm）である。コマンドにより選択が可能となっている。

ＤＭＬＧ（ＤＭＬ generator）は、ＱＯＰＴＰにより最適化されたアクセス・パスから、目標ＤＢＭＳのＤＭＬを生成するプロセスである。ただし、アクセス・パスが代表的問合せに一致した場合は、パターン番号に従って、既に登録されているライブラリからＤＭＬを生成する。

8.2.2　ＬＱの内部表現

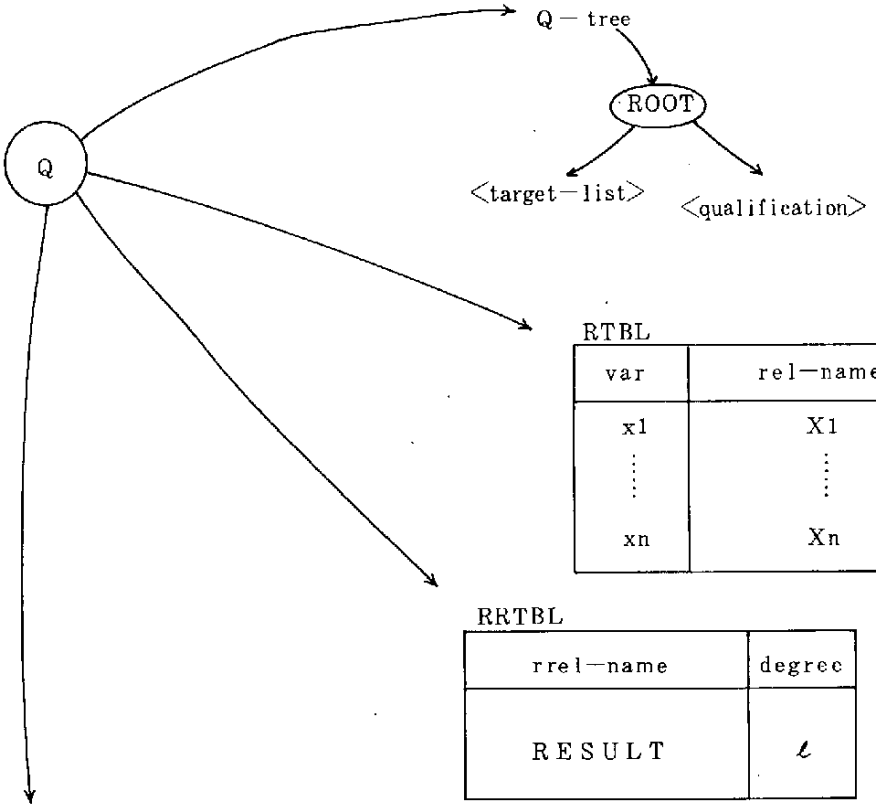
ＬＣＳ問合せ（ＬＱ）は、ＱＴＧによって、内部表現としての２分木（Ｑ－tree）と、３つのテーブルに変換される。

例えば、問合せ（Ｑ）は、次のように変換される。

問合せＱ：

```
range of (x1、x2、……、xn)(X1、X2、……、Xn);
retrieve into RESULT (<target－list>)
  where <qualification>;
```

内部表現：



RTBL

var	rel－name
x1	X1
⋮	⋮
xn	Xn

RRTBL

rrel－name	degree
RESULT	ℓ

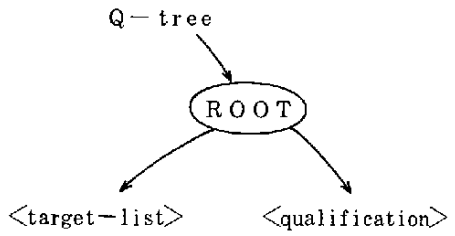
RATBL

rrel－name	att－no	att－name	role	type	length	domain
RESULT	1	A1	N			
〃	2	A2	N			
〃	⋮	⋮	⋮			
〃	ℓ	Aℓ	N			

HIから得る

A. Q－tree

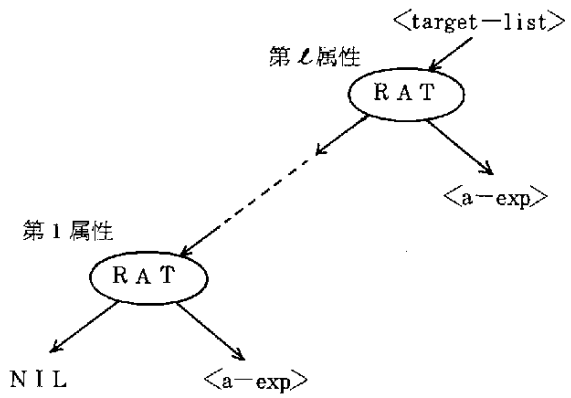
Q－treeは、次のような構造をしている。



<target－list>::= <t－clause>{、<t－clause>}

<t－clause>::= <attribute> = <a－exp> | <v－attribute>

であるから、



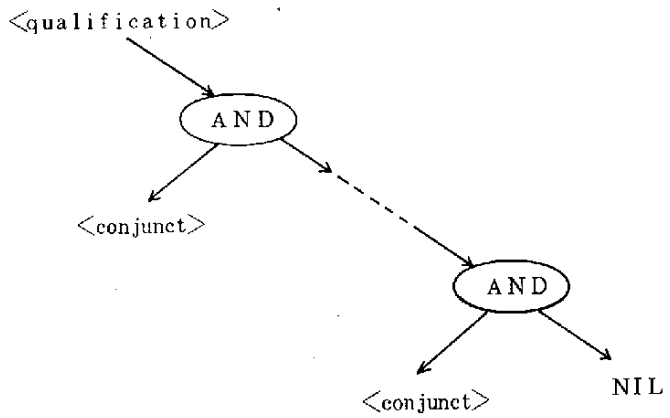
と表わせる。ここで、j番目の結果属性（result－attribute）に当る算術式（<a－exp>）と、RATBL内の結果属性名との対応は、RATノード内の属性番号とRATBL内の属性番号を介してなされる。

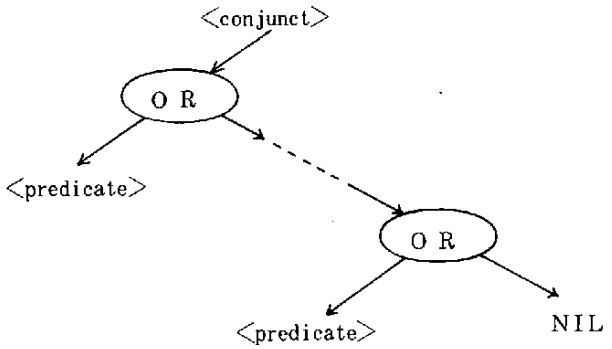
<qualification>部は、積正規形（conjunctive normal form）であると仮定する。即ち、

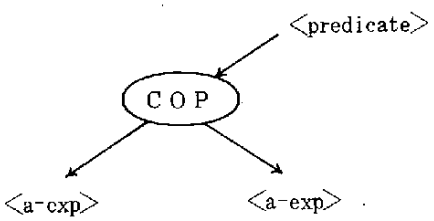
<qualification>::= <conjunct> | <conjunct>{AND <conjunct>}

<conjunct>::= <predicate> | <predicate>{OR <predicate>}

という形に限られる。ただし、<conjunct>内の全ての<predicate>は、同一の組変数を参照していなければならない（modified積正規形）〔7.11を参照のこと〕。





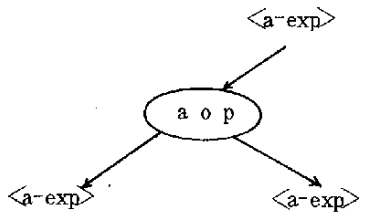


ここで、＜cop＞は比較演算子を表わすノードである。また、＜a−exp＞は、属性と定数上に定義された算術式（算術関数とaggregate関数を含む）の２分木表現である。

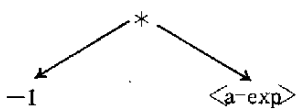
＜cop＞ ::= LT | LE | EQ | GE | GT | NE | =

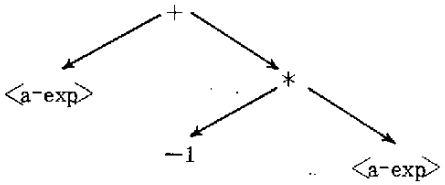
＜a−exp＞ ::= ＜constant＞ | ＜v−attribute＞ | −＜a−exp＞ |
(＜a−exp＞) | ＜a−exp＞ ＜aop＞ ＜a−exp＞ |
＜a−function＞ | ＜g−function＞

＜aop＞ ::= + | − | * | / | **



但し、－＜a－exp＞及び＜a－exp＞－＜a－exp＞は、次のように表わす。

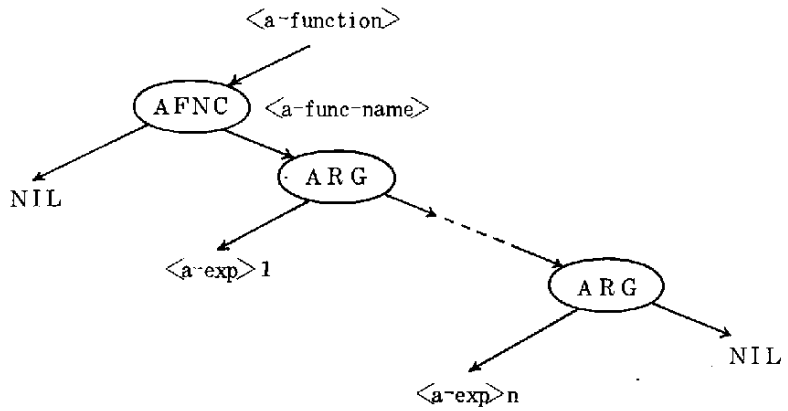




関数には、算術関数とaggregate関数の２種類がある。算術関数は次のような形式をもつ。

＜a－function＞::＝＜a－func－name＞（＜a－exp＞$_1$、……、＜a－exp＞n）

対応するtreeは次のように表わされる。

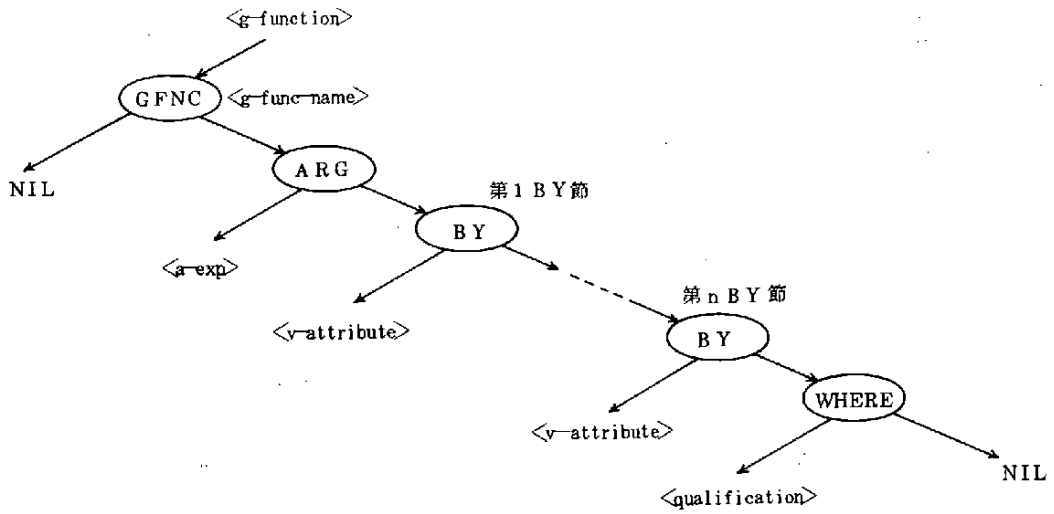


AFNC は算術関数の種類を表わすノードである。

＜a－exp＞$_i$ はｉ番目のパラメータを示している。

＜a－func－name＞::＝ＡＢＳ | ＭＯＤ | ……

aggregate関数も同様に表わされる。



<g-function> ::= <g-func-name> (<a-exp>[BY <v-attribute>
{ BY <v-attribute> }]
[WHERE <qualification>])

<g-func-name> ::= COUNT | MAX | MIN | SUM | AVER |
UCOUNT | USUM | UAVER | ANY

B．Q－treeにおけるノード構造

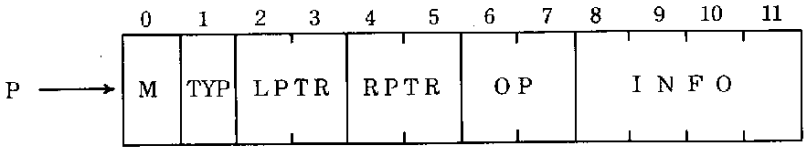
ここでは、Q－treeを構成するノードの構造について述べる。ノードは、主記憶内の連続領域の連続した3ワード（12バイト）から成る。このノードが存在する連続領域を自由ノード空間（FNS）と呼ぶ。

ノードの種類は大別すると、原子ノードと演算子ノードに分けられる。

演算子ノードは主に、2つのポインタ・フィールドと演算子フィールドから成っている。これに対して原子ノードは、1つのポインタ・フィールドと原子情報フィールドから成る。

原子ノードとしては、属性ノード、定数ノード、オーバーフロー領域としてのatomicノードの3種類がある。また、演算子ノードとしては、ROOT、論理演算子、算術演算子、関数パラメータ（BY、WHERE、ARG）、関数、結果属性ノード等がある。

木（tree） 内のノードは、次のような連続した3ワード（12バイト）から構成される。



Pはノードのポインタを示す。ノードの各フィールドの値は、それぞれ、M(P)、TYP(P)、LPTR(P)、RPTR(P)、INFO(P) として参照される。各フィールドの概略については、表8－3に示す。また、ノードのタイプとノード名は、表8－4に列挙されている。

表8－3　ノードのフィールド構成

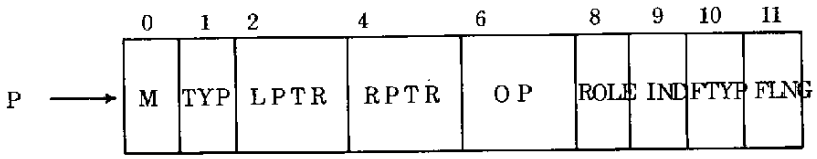
フィールド名	SIZE (バイト)	説　　明
M	1	garbage collection (GBC) において、使用中のノードをマークするためのfieldである。 M(P)＝2：GBCでノードPは使用中treeのルートノード M(P)＝1：GBCでノードPは使用中tree内のノード M(P)＝0：GBCでノードPは未使用 GBC以外では常にM〔P〕＝0である。
TYP	1	ノードの種類を表わす。 TYP(P)＝ 0：attribute ノード, 1：constant ノード, 2：atomic ノード ｝原子ノード ／ 5：operator ノード ｝演算子ノード
LPTR	2	pointer の値が格納される。 TYP(P)＝5 (OPN) の場合は、左手の subtree への pointer が入る。
RPTR	2	TYP(P)＝0、1、2 (原子ノード) のとき、固有な原子情報が入る。 TYP(P)＝5の場合は、右手の subtree への pointer が入る。
OP	2	TYP(P)＝5のとき、operator code が入る。
INFO	4	ノード・タイプ毎に固有な情報が格納される。

表８－４　ノード・タイプとノード名

ノード・タイプ	ノード名	説　　明
原子ノード	ＡＴＴＮ	attribute ノード。 <target－list> 又は <qualification> 内で参照される attribute を表わすノード。
	ＣＮＴＮ	constant ノード。整数値と文字定数を表わすノード。
	ＡＴＭＮ	ＡＮＤノードとＲＯＯＴの bit－map 情報を格納するために使用される。ＡＮＤ／ＲＯＯＴノードの overflow ノード。
演算子ノード	ＡＯＰＮ	算術演算子（＋、＊、／、＊＊）ノード。
	ＣＯＰＮ	比較演算子（＜、≦、＝、≧、＞、≠）ノード。
	ＡＮＤＮ	論理演算子ＡＮＤを表わすノード
	ＯＲＮ	論理演算子　ＯＲを表わすノード
	ＮＯＴＮ	論理演算子ＮＯＴを表わすノード
	ＡＦＮＣＮ	算術関数ノード
	ＧＦＮＣＮ	aggregate関数ノード
	ＡＲＧＮ	argument　ノード。関数のパラメータ・リストを表わす。
	ＢＹＮ	ＢＹ　clause ノード。
	WHEREN	ＷＨＥＲＥ　clause ノード。
	ＲＡＴＮ	result－attribute ノード
	ＲＯＯＴＮ	<target－list> と <qualification> のＲＯＯＴとなるノード。

Ｃ．属性ノード（attribute node ：ＡＴＴＮ）

属性ノードは、目標リスト及び条件部の算術式（<a－exp>）内に現われる属性を表わしている。



フィールド名	説　　明
ＴＹＰ	ＴＹＰ(Ｐ)＝０　（属性ノードであることを示す。）
ＬＰＴＲ	tuple － substitute 時に、この属性がとる値を格納した定数ノードへのポインタが入る。それ以外ではＮＩＬである。
ＲＰＴＲ	ＲＴＢＬ（range table）へのポインタが入る。
ＯＰ	この属性の属性番号（ＡＴＴＮＯ）が入る。
ＲＯＬＥ	この属性のリレーション内での役割を示す。 ＲＯＬＥ(Ｐ)＝｛０（ＮＫＥＹ）……non key １０（Ｋ Ｅ Ｙ）……primary key ２０（ＣＫＥＹ）……composite key
ＩＮＤ	この属性上のインデックス情報
ＦＴＹＰ	この属性のとり得る値のデータ・タイプを示す。 （整数タイプ＝２、文字タイプ＝１）
ＦＬＮＧ	この属性のとり得る値の長さを示す。（０～６３）

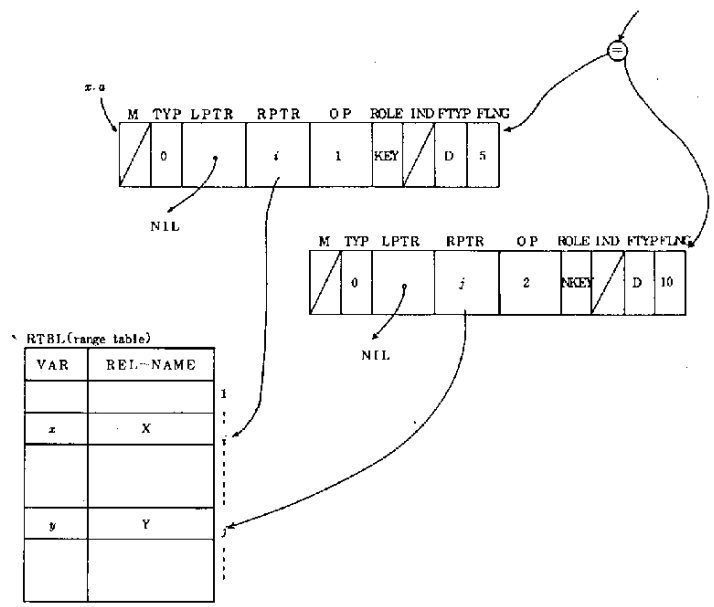
（例）

リレーションのスキームを

Ｘ（ａ、ｂ）、Ｙ（ｃ、ｄ、ｅ）

ｘとｙは、それぞれＸとＹの組変数（tuple variable）とする。

1）　ｘ．ａ＝ｙ．ｄ



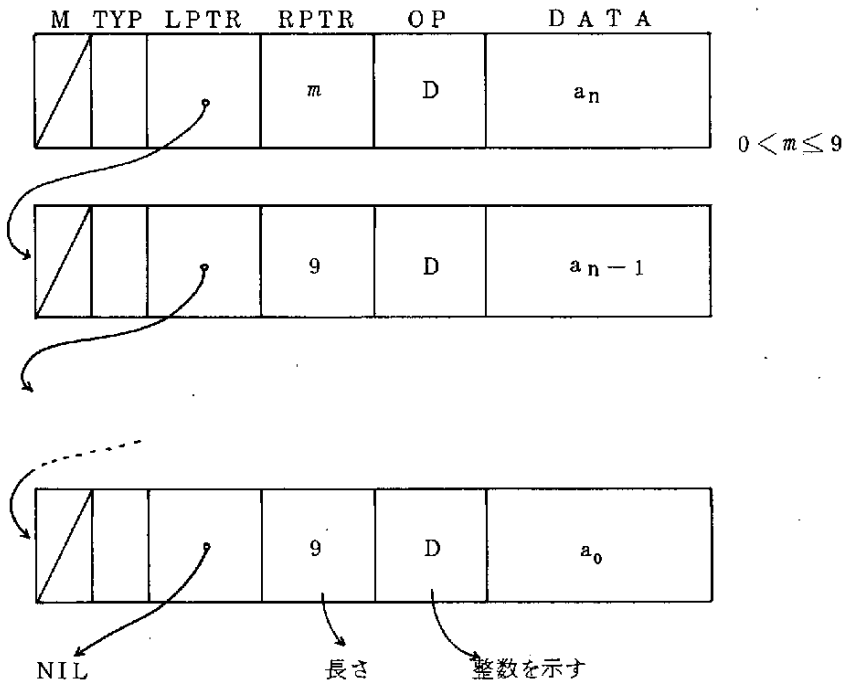
D．定数ノード（constant node：ＣＮＴＮ）

定数ノードは、整数、又は、文字定数を表わす。１つのノードでは、９桁までの整数、又は、４文字までの文字を格納できる。これを超えるものは、複数のＣＮＴＮをチェインして表わしている。

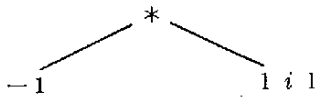
整数の場合、整数ｉを次のように展開する。

$$|\ i\ | = a_0 + a_1 \cdot \alpha + a_2 \cdot \alpha^2 + \cdots\cdots + a_n \cdot \alpha^n$$

ここで、$\alpha = 10$ MAXNN、$0 \leq a_j < \alpha$ である。また、$2^{31} > 10^{10}$ であるから、ここでは、ＭＡＸＮＮ＝９としている。従って、整数値ｉは、次のように複数の定数ノードを用いて表わされる。



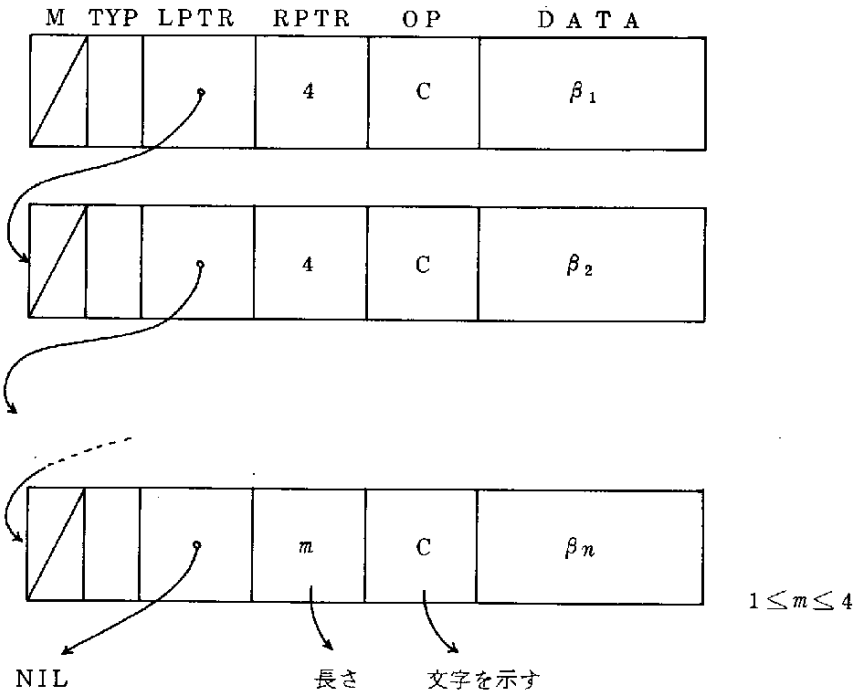
但し、負数は次のように表わされる。



一方、文字定数の場合は、βi（i=1、…、n－1）を４文字の文字列、βn を１～４文字の文字列とするとき、

$$C = "\beta_1\ \beta_2\ \cdots\cdots\ \beta n"$$

と書けるので、複数の定数ノードを用いて次のように表わされる。



また、逆に、ある整数値以下の整数については定数ノードを生成せずに、ポインタフィールドに値を埋め込む方式をとっている。即ち、１つのポインタフィールド（ＬＰＴＲ、ＲＰＴＲ）として２バイト（１６ビット）を使用し、ポインタの値はアドレスではなく、自由ノード空間（ＦＮＳ）のノード番号を用いている。従って、ｐをポインタ値とすると

ＭＩＮＦ ≦ ｐ ≦ ＭＡＸＦ となる。

ここで、ＭＩＮＦとＭＡＸＦは、ＬＰＴＲとＲＰＴＲがポインタとしてとり得る最小値、最大値である。（例えば、ＭＩＮＦ＝１、ＭＡＸＦ＝２０００とすれば、ＦＮＳのノード数は２０００となる。）

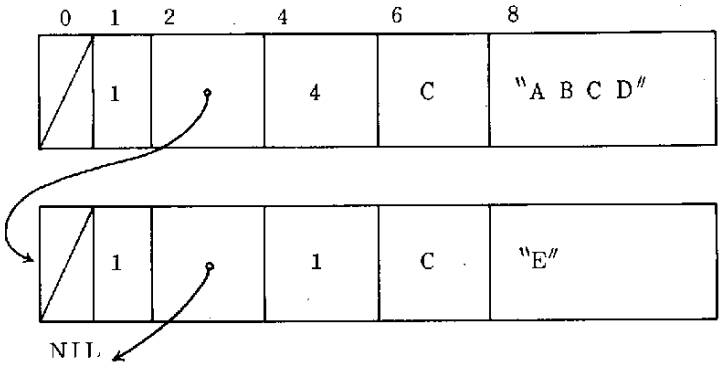
一般に、ＭＡＸＦ＜2^{16}であるから、正整数ｉがＭＡＸＦ＋ｉ＋α＜2^{16} を満足すれば、このｉを格納する定数ノードを生成せずに、ｉを参照するノードのＬＰＴＲ又はＲＰＴＲに埋め込めばよい。こうすることによって、ｉ＜2^{16} －ＭＡＸＦ－αの正整数については、ノードを生成せずにすむとともに、処理速度を向上できる。ここで、αは１０程度の整数とする。

例えば、ＭＡＸＦ＝２０００、α＝１０とすると、整数ｉ＝５６８０は5680+2000+１０＝７６９０という値として、ポインタフィールドにセットされる。

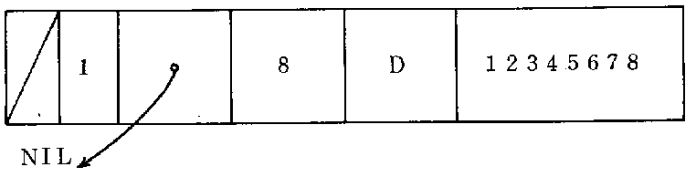
フィールド名	説　　明
ＴＹＰ	ＴＹＰ（Ｐ）＝１　（定数ノードであることを示す）
ＬＰＴＲ	多倍長データを格納する場合、次のＣＮＴＮへのポインタ。 他の場合はＮＩＬ
ＲＰＴＲ	ＤＡＴＡ部に入る値の有効長
ＯＰ	値のタイプが入る。 ＯＰ（Ｐ）＝{ １（Ｃ）……文字 ２（Ｄ）……整数
ＤＡＴＡ	値が格納される。 整数のときは、$10^{10}-1$までの値、文字のときは４文字まで格納できる。

（例）

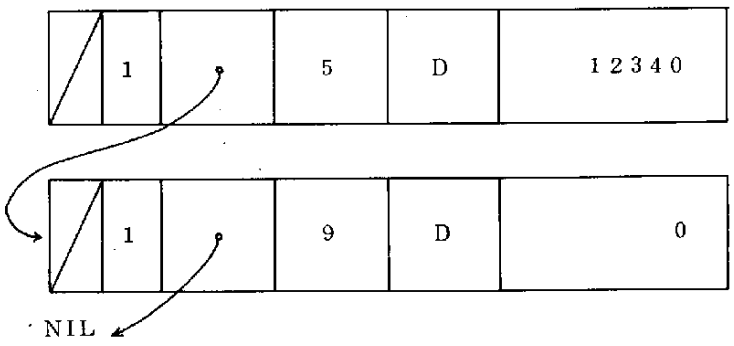
1）　文字例　"ＡＢＣＤＥ"



2）　整　数　12345678

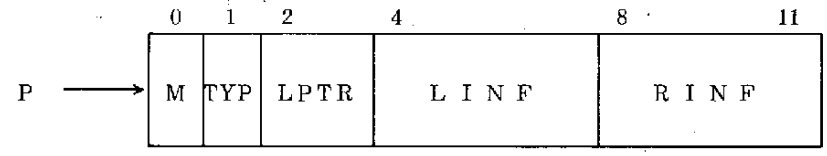


3）　整　数　12340000000000



E．原子ノード（atomic node：ＡＴＭＮ）

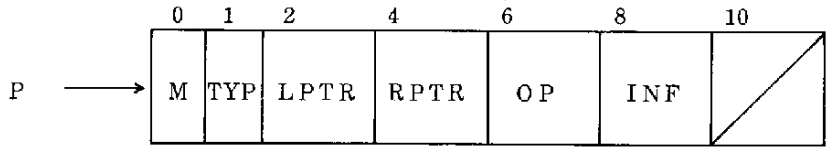
原子ノードは、ＡＮＤノードとＲＯＯＴノードのオーバーフロー・ノードとして用いられる。



フィールド名	説　　明
Ｔ Ｙ Ｐ	ＴＹＰ（Ｐ）＝２　（原子ノードであることを示す）
ＬＰＴＲ	次のＡＴＭＮへのポインタ
ＬＩＮＦ	任意の情報が格納される。
ＲＩＮＦ	詳細は、ＡＮＤ／ＲＯＯＴノードの項に記述する。

Ｆ．結果属性ノード（result attribute node：ＲＡＴＮ）

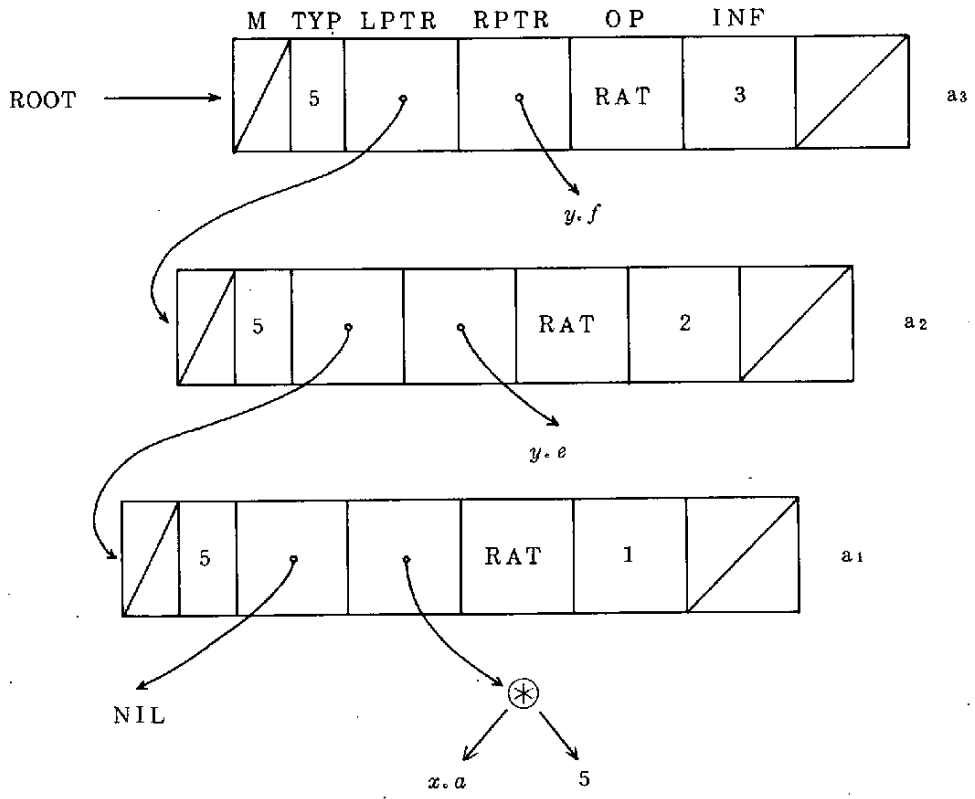
結果属性ノードは、目標リスト（target－list）の分岐ノードであり、結果属性の属性番号（ＡＴＴ－ＮＯ）を示している。



フィールド名	説　　明
Ｔ Ｙ Ｐ	ＴＹＰ（Ｐ）＝５　（＝ＯＰＮ）
ＬＰＴＲ	１つ前のＲＡＴＮへのポインタ このＲＡＴＮのＡＴＴ－ＮＯ＝ｎとすると、ＬＰＴＲは ＡＴＴ－ＮＯ＝ｎ－１のＲＡＴＮを指す。 ＡＴＴ－ＮＯ＝１のとき、ＬＰＴＲ（Ｐ）＝ＮＩＬ
ＲＰＴＲ	＜a－exp＞ tree へのポインタ
ＯＰ	ＯＰ（Ｐ）＝ＲＡＴ（＝３２）
Ｉ Ｎ Ｆ	この結果属性の属性番号　（ＡＴＴ－ＮＯ）

（例）　＜target－list＞が次のように記述されているとすると、＜target－list＞treeは次のようになる。

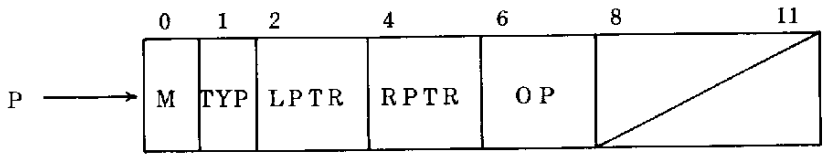
(a_1 =x. a*5、 a_2 =y. e、 a_3 =y. f)



また、a_1 、a_2 、a_3 は、属性番号とともにＲＡＴＢＬに格納される。

G. 単項演算子ノード(unary operator node)

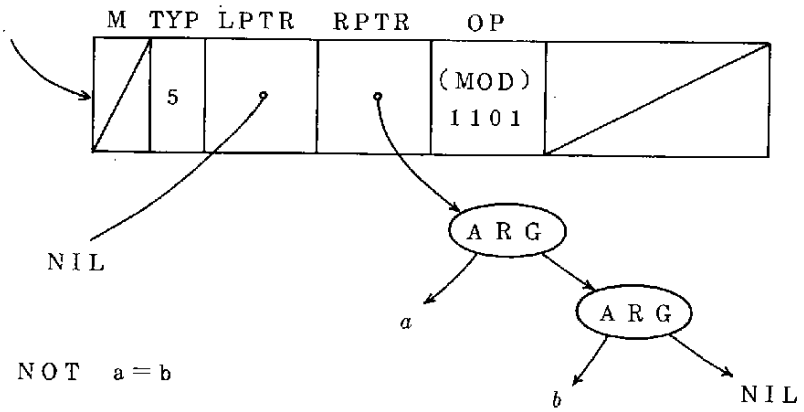
ＮＯＴノード	（ＮＯＴＮ）
算術関数ノード	（ＡＦＣＮ）
aggregate関数ノード	（ＧＦＣＮ）



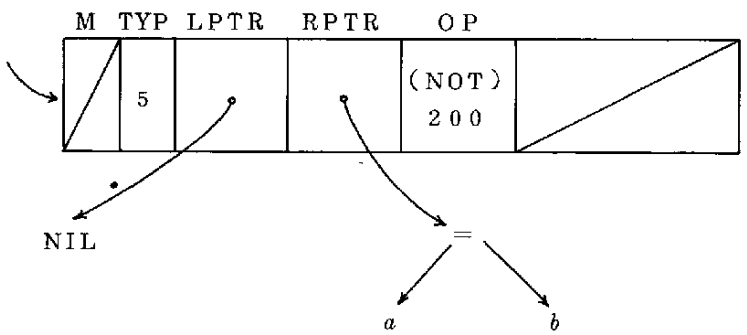
フィールド名	説　　明	
T Y P	TYP(P)＝5　(＝OPN)	
L P T R	LPTR(P)＝NIL	
R P T R	ポインタ。関数の場合は、パラメータ・リストへのポインタ	
O P	OP(P)＝NOT　(200)	aggregate関数
	ANY　(1000)	
	UAVER　(1001)	
	UCOUNT(1002)	
	USUM　(1003)	
	AVER　(1020)	
	COUNT　(1021)	
	SUM　(1022)	
	MAX　(1023)	
	MIN　(1024)	
	ABS　(1100)	算術関数
	MOD　(1101)	

(例)

1)　MOD(a、b)

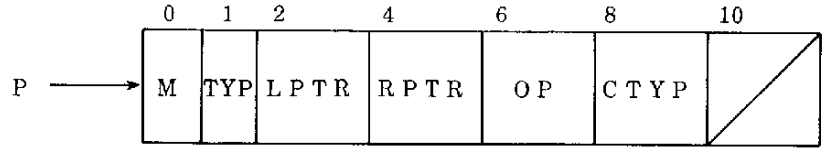


2)　NOT　a＝b



H．比較演算子ノード（ＣＯＰＮ）

ＣＯＰＮは、比較演算子を表わす。この他に、比較述語の種類を示す情報（ＣＴＹＰフィールド）をもっている。

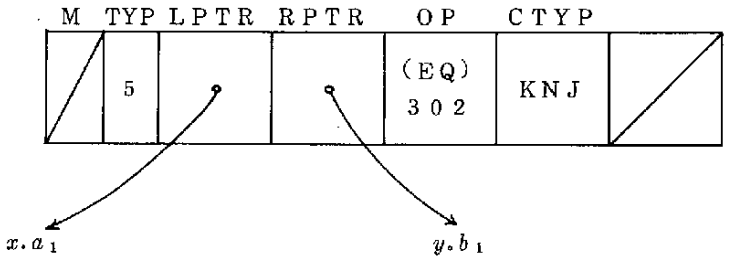


フィールド名	説　　明
ＴＹＰ	ＴＹＰ(Ｐ)＝５　（＝ＯＰＮ）
ＬＰＴＲ	１ｓｔ ｏｐｅｒａｎｄへのポインタ
ＲＰＴＲ	２ｎｄ ｏｐｅｒａｎｄへのポインタ
ＯＰ	ＯＰ(Ｐ)＝ＬＴ(＜)　（３００） ＬＥ(≦)　（３０１） ＥＱ(＝)　（３０２） ＧＥ(＞)　（３０３） ＧＴ(＞)　（３０４） ＮＥ(≠)　（３０５）
ＣＴＹＰ	ＣＴＹＰ(Ｐ)＝ＫＫＪ　（０） ＫＮＪ　（１） ＮＫＪ　（２） ＮＮＪ　（３）　} ｊｏｉｎ ｃｌａｕｓｅ ＫＲ　（１０） ＮＲ　（１１）　} ｒｅｓｔｒｉｃｔｉｏｎ ｃｌａｕｓｅ

比較述語は、$<a-exp_1>$ $<cop>$ $<a-exp_2>$という形式をしている。ここで、$<a-exp_1>$、$<a-exp_2>$は算術式であり、$<cop>$は比較演算子である。また、$<a-exp_2>$が定数であるとき、この句を制限（ｒｅｓｔｒｉｃｔｉｏｎ）述語と呼び、$<a-exp_1>$、$<a-exp_2>$ともに属性をもつとき結合（ｊｏｉｎ）述語と呼ぶ。$<a-exp_1>$、$<a-exp_2>$がキー属性をもつとき"Ｋ"、もたないときを"Ｎ"で表わすと、ＣＴＹＰ(Ｐ)は、次表で決められる。

	$<a-exp_1>$	$<a-exp_2>$	ＣＴＹＰ
ｊｏｉｎ ｃｌａｕｓｅ	Ｋ	Ｋ	ＫＫＪ
	Ｋ	Ｎ	ＫＮＪ
	Ｎ	Ｋ	ＮＫＪ
	Ｎ	Ｎ	ＮＮＪ
ｒｅｓｔｒｉｃｔｉｏｎ ｃｌａｕｓｅ	Ｋ	—	ＫＲ
	Ｎ	—	ＮＲ

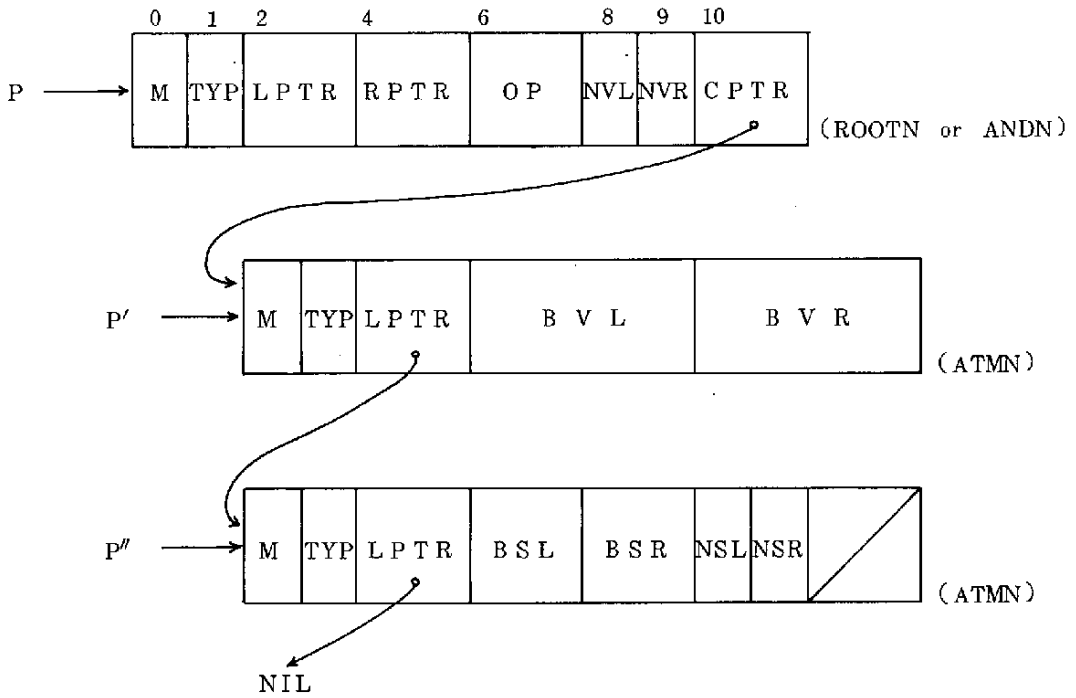
（例）　a_1 はxのキー属性、b_1 はyのキー属性ではないとする。比較句 $x.a_1 = y.b_1$ は、次のようになる。



I．ROOTノード（ROOTN）

　ANDノード（ANDN）

　ROOTNは、＜target－list＞treeと＜qualification＞treeとの根（root）を表わす。ANDNは、AND演算子を表わす。ROOTNとANDNは、そのノードの左手と右手の部分木（subtree）内の変数とサイト情報を保有している。このため、これらのノードは、原子ノード（ATMN）がチェインされた多変長ノードとなっている。



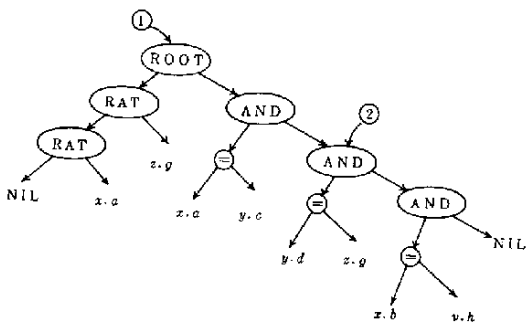
bit－map（BVLとBVR）の各ビット位置は、RANGEテーブルの組（tuple）番号に対応しており、ビットがONであることは、その木（tree）内に対応する変数が存在することを表わしている。同様に、bit－map（BSLとBSR）はSITE－NOテーブルに対応している。（QTPでは、ある1つのサイト内だけの処理であるので、サイトに関するbit－mapは使用しない。）

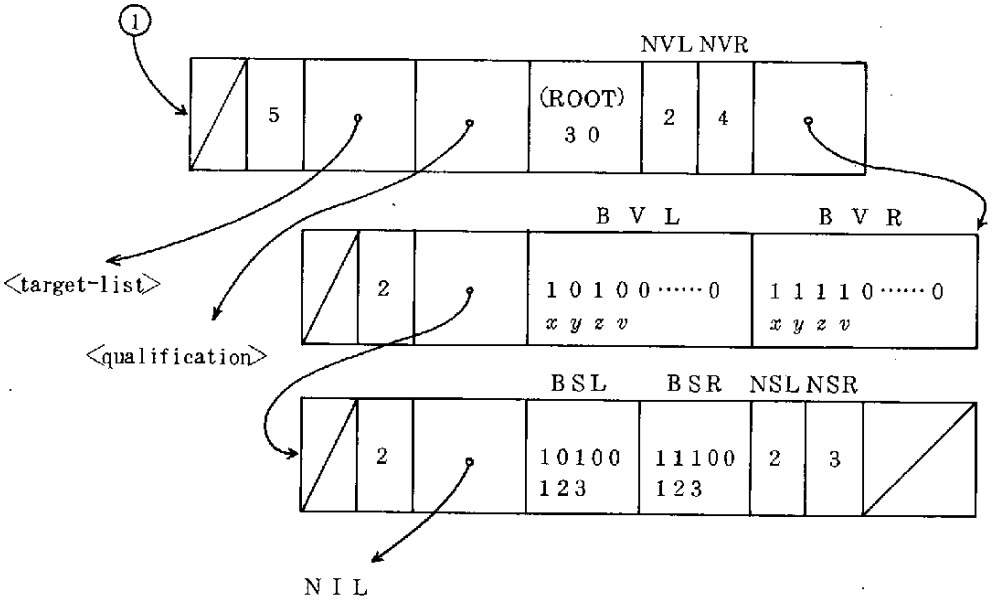
フィールド名	説　　明
T Y P	TYP(P)＝5　(＝OPN)
L P T R	A N D N　：1 st operandへのポインタ
	R O O T N：target－list treeへのポインタ
R P T R	A N D N　：2nd operandへのポインタ
	R O O T N：qualification treeへのポインタ
O P	O P (P)　＝　A N D　　(１００) R O O T　(　３０)
N V L	このノードの左手のsubtree内の変数の数(≦３２)
N V R	このノードの右手のsubtree内の変数の数(≦３２)
C P T R	C P T R (P)＝P′
B V L	B V L (P′)＝左手のsubtree内の変数のbit－map　(32ビット)
B V R	B V R (P′)＝右手のsubtree内の変数のbit－map　(32ビット)
B S L	B S L (P″)＝左手のsubtree内のサイトのbit－map(16ビット)
B S R	B S R (P″)＝右手のsubtree内のサイトのbit－map(16ビット)
N S L	N S L (P″)＝左手のsubtree内のサイト数(≦１６)
N S R	N S R (P″)＝右手のsubtree内のサイト数(≦１６)

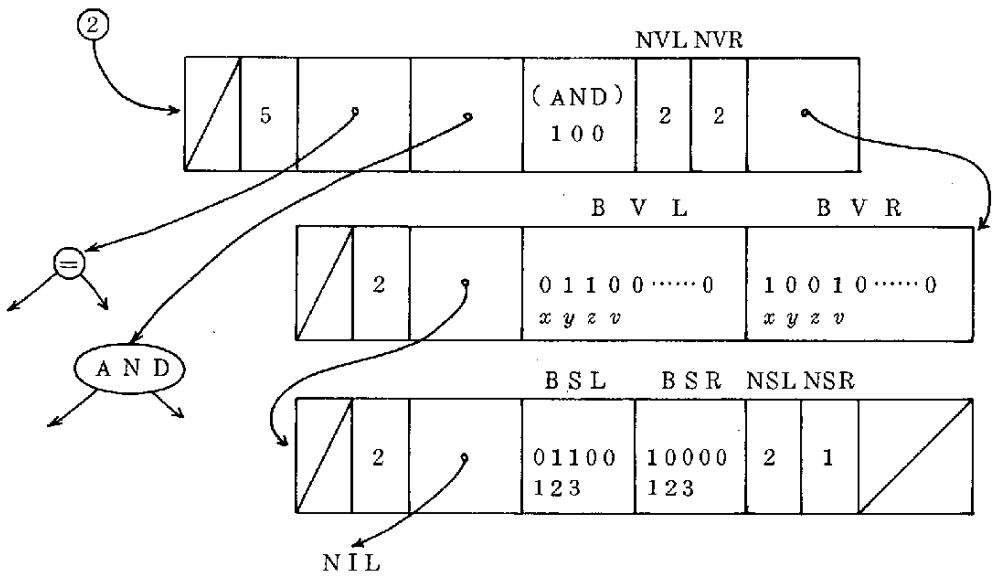
(例)　リレーションスキーム　X(a、b　　)…… サイト1
Y(c、d、e)…… サイト2
Z(f、g　　)…… サイト3
V(h、i　　)…… サイト1

x、y、z、vは、各々X、Y、Z、Vの組変数(tuple variable)。

1)　retrieve (a=x. a　b=z.g)
WHERE　x.a＝y.c　AND　y.d＝z.g　AND　x.b＝v.h





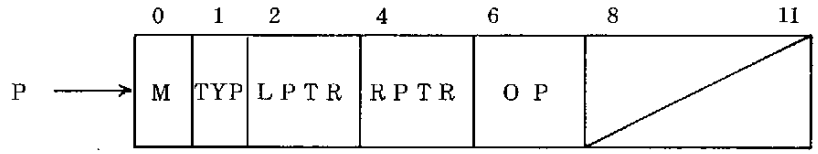


J．算術演算子ノード（AOPN）

ORノード　　（ORN）

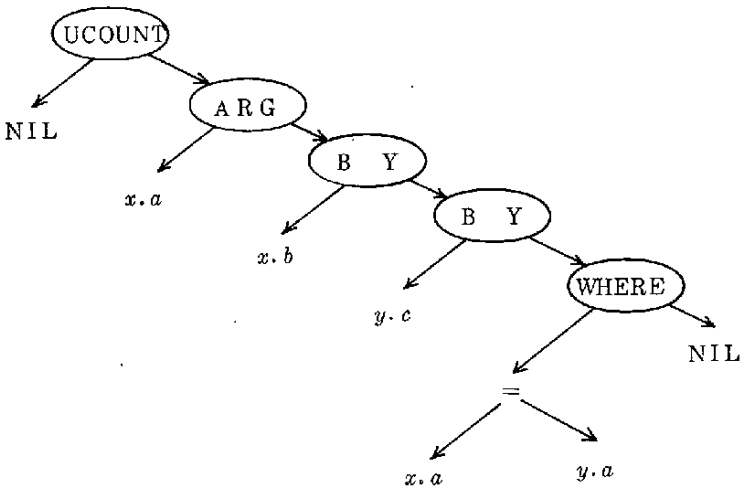
BY、WHEREノード

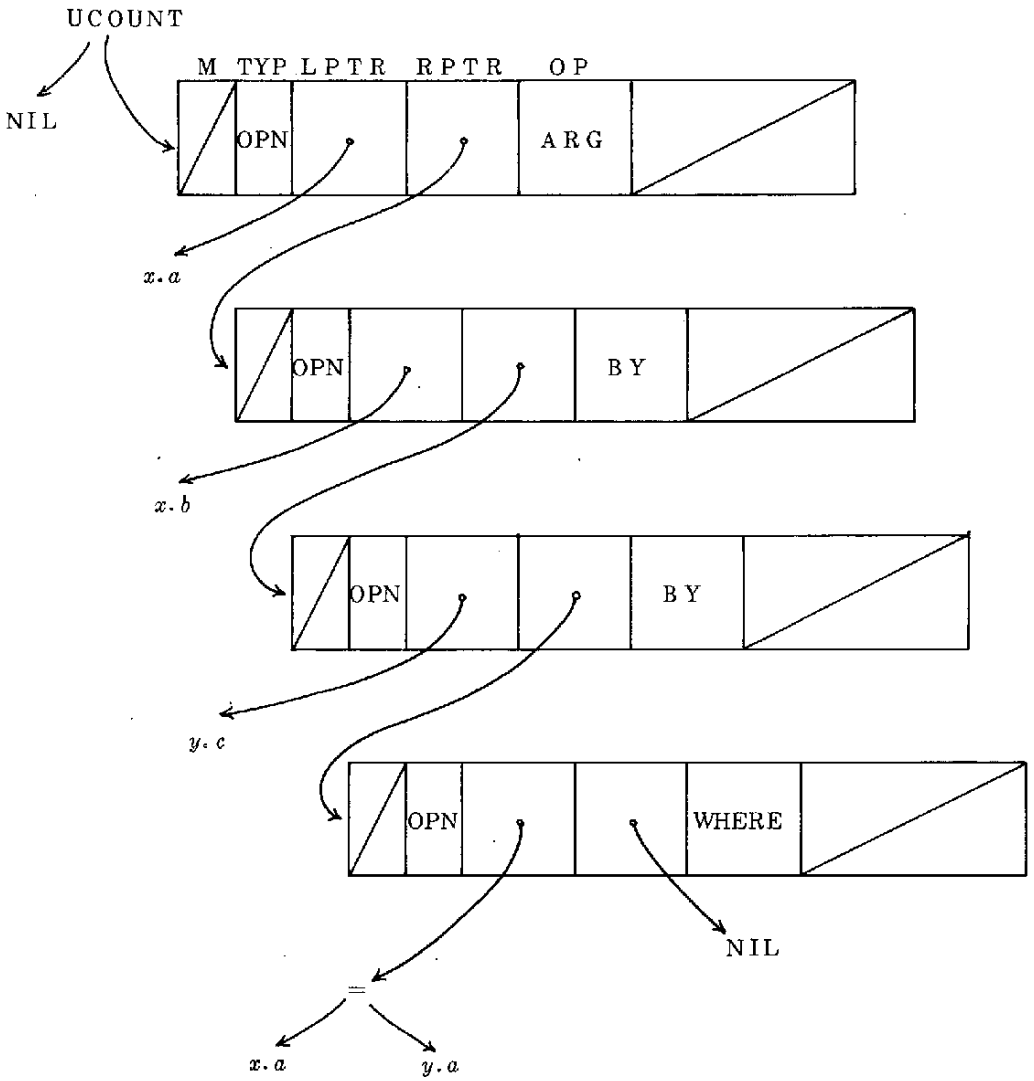
argumentノード（ARGN）



フィールド名	説　　明
T Y P	TYP(P)＝5　(＝OPN)
L P T R	1 st operandへのポインタ
R P T R	2 nd operandへのポインタ
O P	OP(P)＝OR　　(　101)
	PLUS　(　400)
	TIMS　(　600)
	DIV　(　601)
	POW　(　700)
	BY　(1300)
	WHERE　(1400)
	ARG　(1200)

(例)　UCOUNT(x. a BY x. b BY y. C WHERE x. a＝y. a)





8.2.3 QTG（Query Tree Generator）

QTGは、QUEL仕様に基づいたQUEL問合せの文字列を入力として、NDD／HI／ESR、RSR、ATTのスキーマ情報を参照しながら、対応するQ－tree、RTBL（range table）、RRTBL（result－relation table）、RATBL（result－attribute table）を生成する。

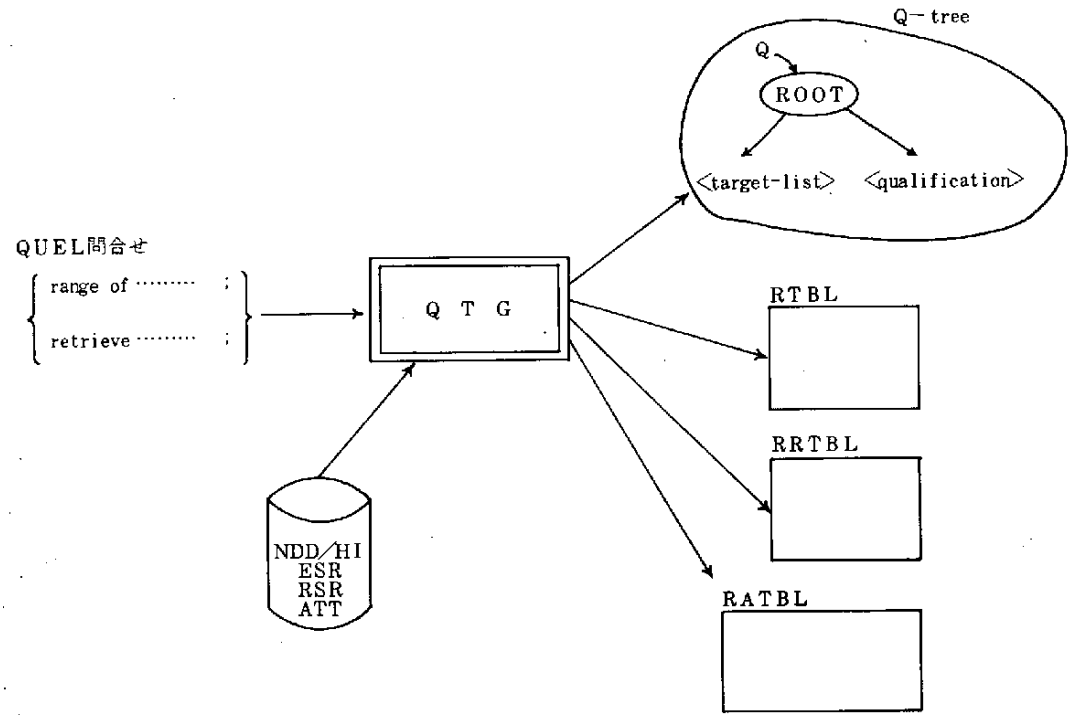


図８－４５　ＱＴＧの概要

Ａ．ＱＴＧの構成

ＱＴＧは、３つのサブプロセス（ＲＡＮＧＥＰ、ＲＥＴＲＶＰ、ＱＵＡＬＰ）から構成されている。ＲＡＮＧＥ文を処理するのがＲＡＮＧＥＰであり、ＲＥＴＲＩＥＶＥ文の＜target－list＞までを処理するのがＲＥＴＲＶＰであり、＜qualification＞を処理するのがＱＵＡＬＰである。

ＱＴＧの入力となるＱＵＥＬ問合せの形式的な定義を、付記１に示す。ＲＡＮＧＥ文は、この問合せ内で用いられる組変数（tuple variable）を宣言するものである。＜target－list＞は、結果リレーションの属性の構成を定めるものであり、＜qualification＞は、問合せの条件を表わしている。

また、問合せの内部表現としては、Ｑ－tree の他に、３つのテーブル（ＲＴＢＬ、ＲＲＴＢＬ、ＲＡＴＢＬ）から成る。これらのテーブルは、複数の問合せに対して共通に使用される。

ＲＴＢＬ（range table）は、１つのセッション内において、ユーザがＲＡＮＧＥ文を用いて宣言した組変数とリレーション名との対応表である。ＲＴＢＬの構成は、表８－５に示す。

ＲＲＴＢＬ（result－relation table）は、この問合せによって生成されるリレーション

名と、その次数（属性の数）が格納される。ＲＲＴＢＬの構成は、表８－６に示す。

ＲＡＴＢＬ（result－attribute table）は、結果リレーションを構成する結果属性の名前と、その属性番号とが格納される。属性番号は、＜target－list＞において定義された結果属性の左からの順番である。Ｑ－treeの処理において、結果属性は全て、この属性番号によって参照される。ＲＡＴＢＬの構成を、表８－７に示す。

表８－５　ＲＴＢＬの構成

属　性　名	属性番号	ROLE	TYPE	LENGTH	説　　明
VAR	1	K	C	4	組変数（tuple variable）名
REL－NAME	2	N	C	16	リレーション名
SITE－NO	3	N	D	4	所在サイト番号

表８－６　ＲＲＴＢＬの構成

属　性　名	属性番号	ROLE	TYPE	LENGTH	説　　明
REL－NAME	1	K	C	16	リレーション名
DEGREE	2	N	D	2	リレーションの属性数

表８－７　ＲＡＴＢＬの構成

属　性　名	属性番号	ROLE	TYPE	LENGTH	説　　明
REL－NAME	1	C	C	16	リレーション名
ATT－NO	2	C	D	2	属性番号
ATT－NAME	3	N	C	16	属性名（attribute name）
ROLE	4	N	C	1	役割（role）K or N
TYPE	5	N	C	1	データタイプ C or D
LENGTH	6	N	D	4	属性の長さ
DOMAIN	7	N	C	16	領域（domain）名

B．ＲＡＮＧＥＰ

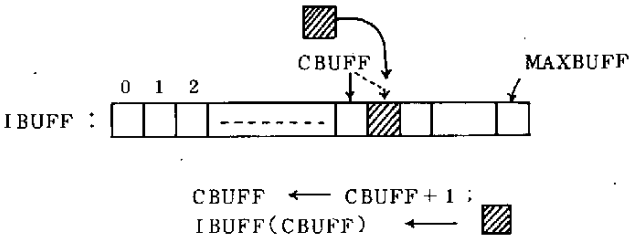
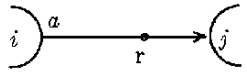
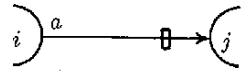
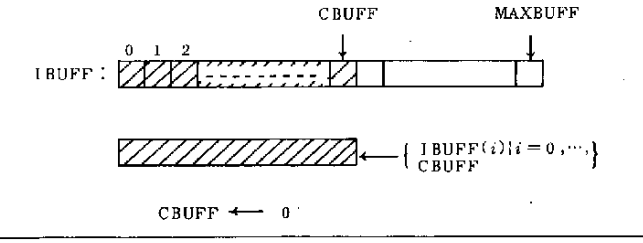
ＲＡＮＧＥＰは、文字列としてのＲＡＮＧＥ文を入力として、対応するＲＴＢＬ（range table）を生成するプロセスである。ＲＡＮＧＥＰでは、入力されたＲＡＮＧＥ文について次のようなチェックを行う。

組変数名（tuple variable name）とリレーション名は、それぞれ、入力順に対応づけられるので、その数は同じでなければならない。また、一つのリレーション名に対して複数の組変数を定義することは可能であるが、逆に、一つの組変数を二つ以上のリレーション名に対して用いることはできない。既に宣言されている変数に対して、新しいリレーションを用いようとしたときは、

ＲＴＢＬのリレーション名の置き換えとなる。リレーション名は、ＨＩ／ＥＳＲ、又は、ＨＩ／ＲＳＲにあるものでなければならない。

ＲＡＮＧＥＰの状態遷移図を、図８－４６に示す。状態遷移図の中の記号については、表８－８に述べてある。

表８－８　状態遷移図の記述形式

図　　式	説　　明
	状態ｉを表わす。 ｉは状態番号（ｓｔａｔｅ　ｎｕｍｂｅｒ）である。
	状態ｉから状態ｊへの遷移（ｔｒａｎｓｉｔｉｏｎ）を表わす。 即ち、状態ｉにおいて文字ａが読込まれた時、又は条件式ａを満足する時、状態ｊに遷移することを表わす。
	状態ｉからｊへの遷移の際に、１文字読込むことを表わす。 即ち、ｔｉｍｅ　ｓｅｑｕｅｎｃｅは ｓｔａｔｅ(ｉ)→ｅｖｅｎｔ　ａ → ｒｅａｄ→ｓｔａｔｅ(ｊ)
	状態ｉからｊへの遷移の際に、現在読込まれた文字を入力バッファ（ＩＢＵＦＦ）にｐａｃｋする。 
	ｅｖｅｎｔ　ａによって、状態ｉからｊへ遷移する際にあるａｃｔｉｏｎ(ｒ)を行うことを表わす。 ｒはａｃｔｉｏｎの説明、又はサブルーチン名である。
	ｅｖｅｎｔ　ａによって状態ｉからｊへ遷移する際に現在まで入力バッファ（ＩＢＵＦＦ）にｐａｃｋされていた文字列を取出すことを表わす。 

また、状態遷移図において

αはアルファベット１文字（Ａ～Ｚ、￥）

βはアルファベット or 数字 or 特殊文字（－、＃）のうち１文字

ｄは数字（０～９）１文字

を、各々示している。

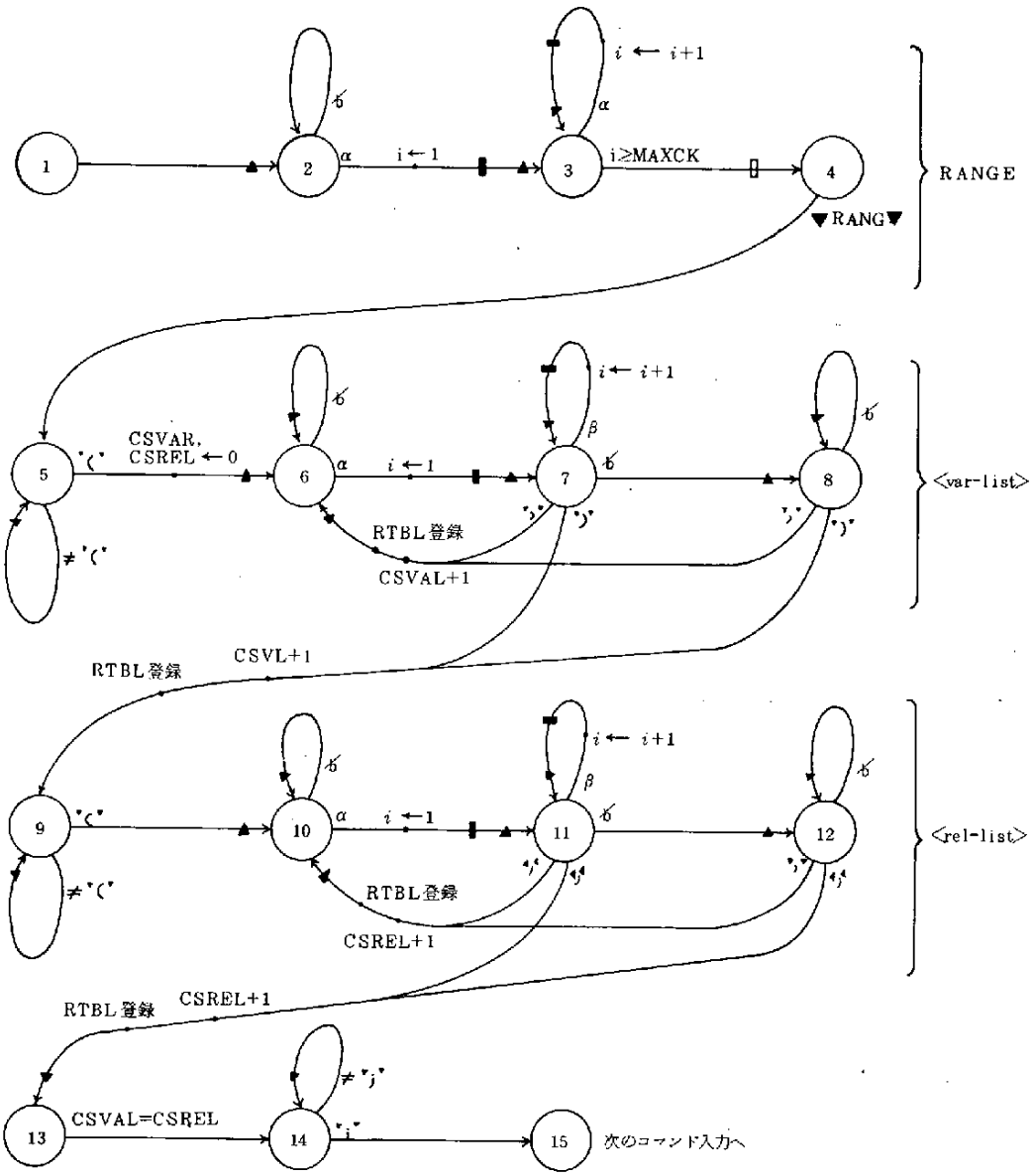


図８－４６　ＲＡＮＧＥＰの状態遷移図

C．RETRVP

RETRVPは、RETRIEVE文の<target－list>までを入力として、対応する<target－list> tree及び、RRTBL、RATBLの2つのテーブルを生成する。<target－list>内の算術式の処理には、サブルーチンQUALを用いる。"INTOリレーション名"が省略された場合は、仮のリレーション名がとられる。

リレーション名が既にRRTBLに登録されているときは、ユーザに問合せる。以前のリレーションに置き換えてもよいときは、RRTBLを置き換え、同時に、以前のリレーションに対応するRATBLの組（tuple）をテーブルから削除する。

<v-attribute>については、次のチェックを行う。組変数名（tuple variable name）はRTBLに宣言されていること、また、組変数に対応するリレーション名と属性名をキーとして、HI／ATTをサーチし、HI／ATTになければエラーとする。

RETRVPの状態遷移図は、図8－47に示す。

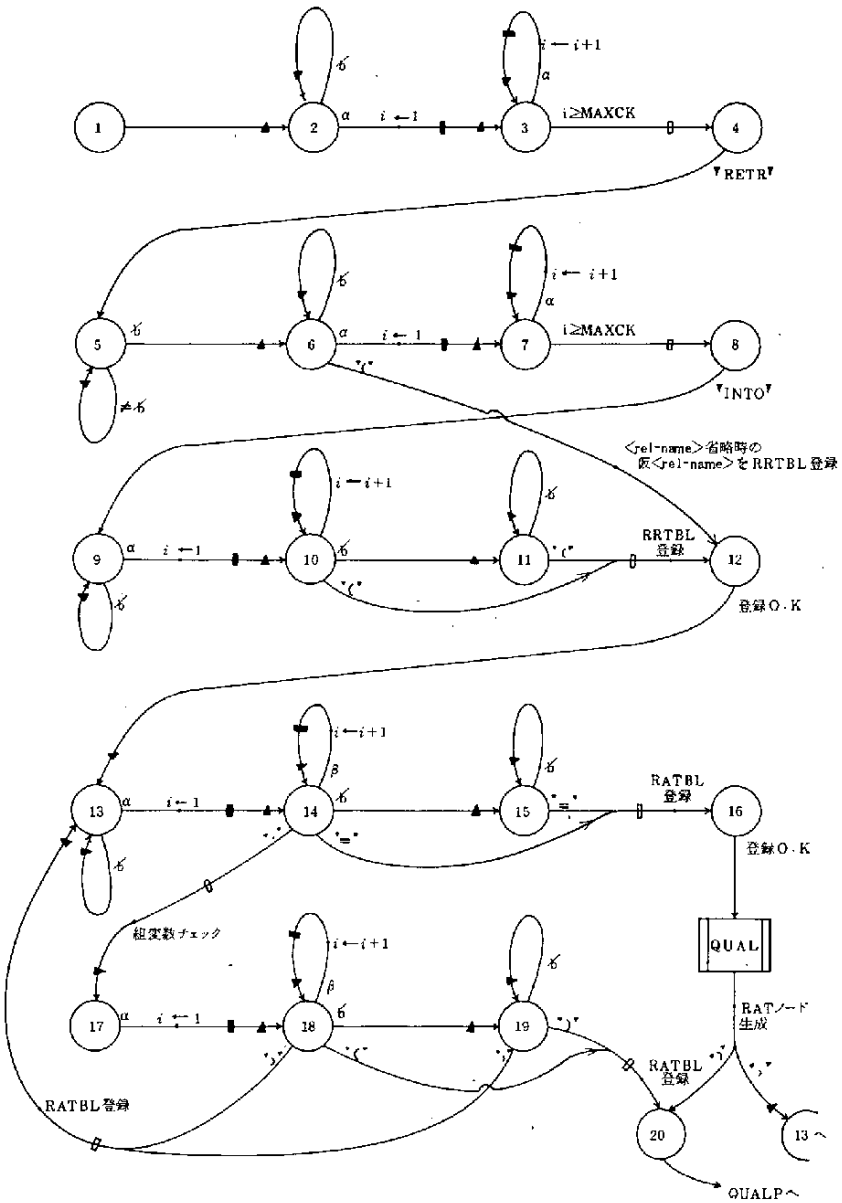


図８－４７　ＲＥＴＲＶＰの状態遷移図

D. ＱＵＡＬＰ

ＱＵＡＬＰは、"ＷＨＥＲＥ<qualification>"を入力として、対応する<qualification> tree を生成するプロセスである。<qualification>部の処理には、ＲＥＴＲＶＰと同じくサブルーチンＱＵＡＬを用いる。問合せの条件が無い場合は、<qualification> tree としてはＮＩＬとなる。

また、同一の属性（attribute）について唯一つの属性ノード（ＡＴＴＮ）しか生成しないようにするため、ＡＴＴＢＬ（attribute table）〔表８－１１〕を生成・参照する。ＡＴＴＢＬは、属性名と、対応する属性ノード（ＡＴＴＮ）との対応表となっている。

ＱＵＡＬＰの処理が終ったら、<target－list> tree と<qualification> treeからＱ－tree を生成する。

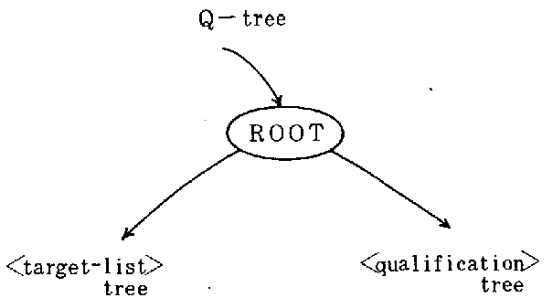


図８－４８　Ｑ－tree

ＱＵＡＬＰの状態遷移図は、図８－４９に示す。

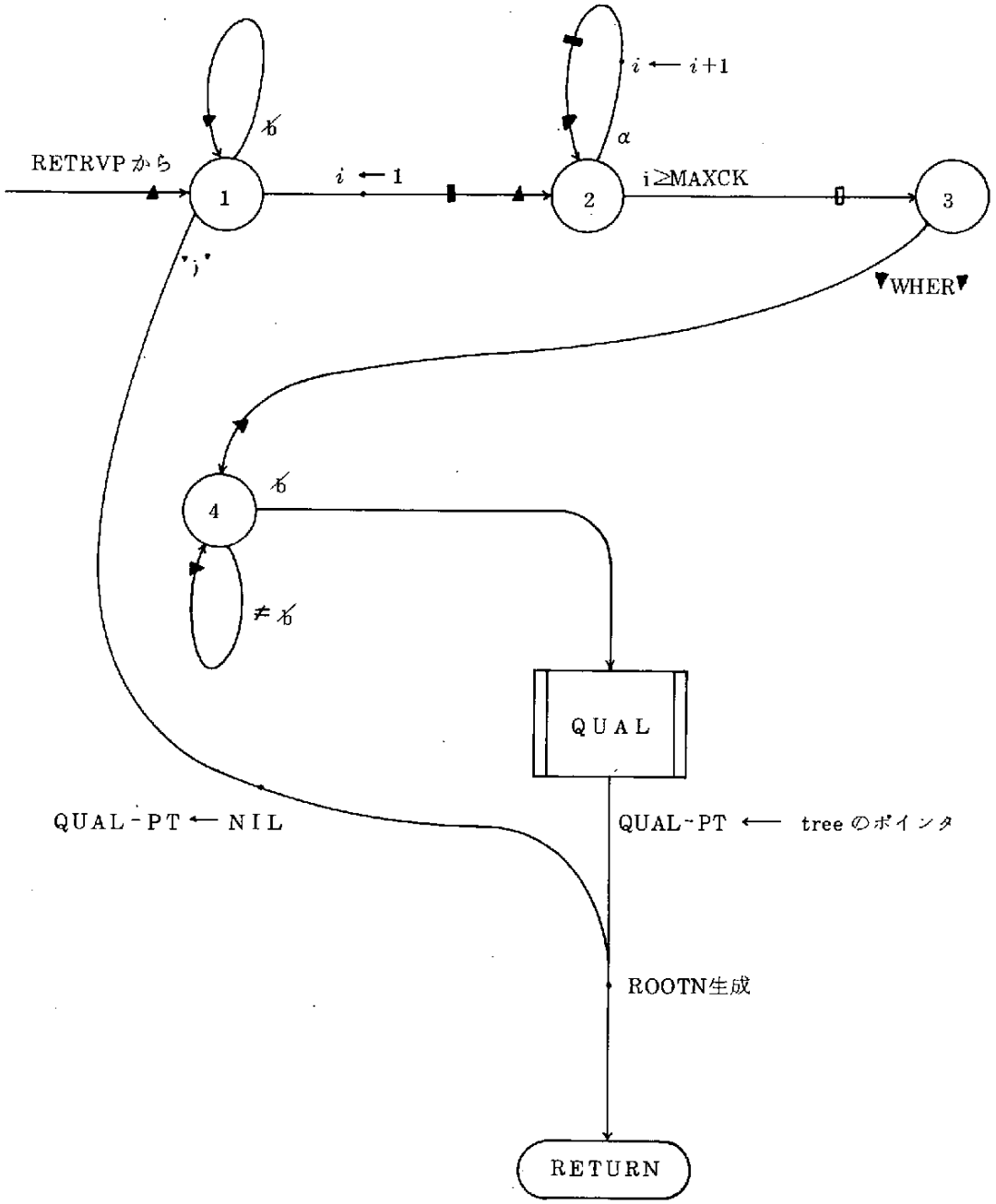


図8－49　QUALPの状態遷移図

E．サブルーチン　ＱＵＡＬ

ＱＵＡＬは、通常の式表現（infix notation）の算術式（<a−exp>）又は、条件式（<qualification>）を入力として、対応する２分木（binary tree）を生成する手続きである。

文字列から２分木を生成するには、次の点を考慮する必要がある。

1）　通常の式表現（infix notation）から前置記法（prefix notation）への変換

2）　定数と属性に対応するノードの生成

3）　演算子記号から内部コード（operator code）への変換

1)の変換は、ＴＴＢＬ（transformation table）〔表８－１０〕を用いて行われる。入力される文字列は、一般に次のパターンから成っている。

<opr> <ope> <cpr> <cpe>

ここで、<opr> と <cpr>は、オペランド（ＮＩＬ又はポインタ）を表わし、<ope> と <cpe>は、演算子コード（operator code）を示す。また、<ope> <cpe> と続いたときは、間の<cpr>はＮＩＬとみなして処理する。そうすると、以前の演算子<ope>と、いま読込まれた演算子<cpe>とから、要求される処理のプロセス番号がＴＴＢＬを用いて決められる。

opec ← <ope> ／100

cpec ← <cpe> ／100

プロセス番号 ← ＴＴＢＬ（opec、cpec）

但し、プロセス番号が0（表で空欄の部分）となったときは、構文エラーであることを示す。また、プロセス番号が９のとき、そのときの<cpr>を木の先頭ノードとして通知し処理を終る。

3)の変換は、ＯＰＴＢＬ（operator table）〔表８－９〕を用いて行われる。ＯＰＴＢＬは、演算子記号と演算子コード、演算子の優先度との対応表である。優先度は、２分木から文字列へ逆に変換するときに用いられる。

属性ノードを生成するときは、属性ノードを冗長に生成させないために、ＡＴＴＢＬ（attribute table）を用いる。ＡＴＴＢＬの構成は、表８－１１に示す。ＡＴＴＢＬは、<target−list>と<qualification> 内で参照される属性と、この属性の木表現との対応表となっている。このＡＴＴＢＬの目的は、Ｑ－tree の生成時に１つの属性に対して、冗長に属性ノード（ＡＴＴＮ）を生成させないことであるとともに、問合せ処理において、tuple－substitution時に、ある属性ノードへの値の代入も１つの属性ノードに対して行えばよいようにすることである。ＡＴＴＢＬを設けることにより、ノードの有効利用と処理の効率化を図ることができる。従って、木（tree）生成時に、ある属性のノードを生成しようとする場合、まず、このＡＴＴＢＬをサーチする。もし、ＡＴＴＢＬ内にあれば、新たにノードを生成せずにテーブル内のポインタを使用する。なければ、属性ノードを生成するとともにＡＴＴＢＬに追加をしておく。

参照する属性が多くなる場合には、ハッシングによる検索を用いることも考えている。しかし、１つの問合せの参照する属性数は１０～３０程度であるから、シーケンシャル・サーチで十分と考えている。

表8－9　OPTBL(operator table)

	記　　号	operator code	優 先 度	
1	AND	100	11	
2	OR	101	10	
3	NOT	200	20	
4	LT (<)	300	30	
5	LE (≤)	301	30	
6	EQ (=)	302	30	
7	=	302	30	
8	GE (≥)	303	30	
9	GT (>)	304	30	
10	NE (≠)	305	30	
11	+	400	40	
12	−	500	40	
13	*	600	50	
14	/	601	50	
15	**	700	60	
16	(	800		
17	)	900		
18		0		
19	,	1200		
20	BY	1300		
21	WHER(E)	1400		
22		1500		
23		1500		
24	ANY	1000		unique－aggregate function
25	UAVE(R)	1001		
26	UCOU(NT)	1002		
27	USUM	1003		
28	AVER	1020		non－unique aggregate funiction
29	COUN(T)	1021		
30	SUM	1022		
31	MAX	1023		
32	MIN	1024		
33	ABS	1100		arithmetic function
34	MOD	1101		

表8－10　TTBL（transfomation table）

	CPEC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ŌPEC	<cpe> <ope>	AND OR	NOT	<cop>	+	－	＊ ／	＊＊	(	)	g f	a f	,	BY	WHERE	⊣
1	AND OR	2	2	2	2	2	2	2	2	1	2	2				1
2	NOT	1	2	2	2	2	2	2	2	1	2	2				1
3	LT、LE EQ、＝ GE、GT NE(<cop>)	1		1	2	2	2	2	2	1	2	2				1
4	＋	1		1	1	1	1	2	2	1	2	2	1	1	1	1
5	－	3		3	3	3	3	3	3	3	2	2	3	3	3	3
6	＊ ／	1		1	1	1	1	2	2	1	2	2	1	1	1	1
7	＊＊	1		1	1	1	1	1	2	1	2	2	1	1	1	1
8	(	2	2	2	2	2	2	2	2	4	2	2	2	5	5	
9	)															
10	g—func								2							
11	a—func								2							
12	,				2	2	2	2	2	8	2	2	2	5	5	
13	BY									7				6	6	
14	WHERE	2	2	2	2	2	2	2	2	7	2	2				
15	⊢	2	2	2	2	2	2	2	2		2	2				9

ＱＵＡＬの各プロセスの概要

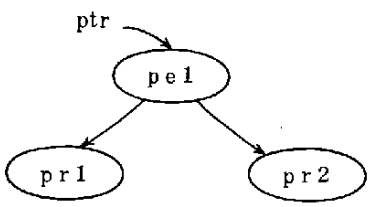
プロセス番号	<opr> <ope> <cpr> <cpe>	処　理　概　要
1		cpr ← st (ope、 opr、 cpr)； PPUP (ope、 opr、 dummy)；
2		PDWN (ope、 opr、 NIL)； opr ← cpr；　　ope ← cpe； read (cpr)；　　read (cpe)；
3	／　— a　—	cpr ← st (TIMES、 MONE、 cpr)； if　opr＝NIL、 　then　PPUP (ope、 opr、 dummy)； 　else　ope ← PLUS (+)；
4	／　(　b　)	PPUP (ope、 opr、 dummy)； if　ope＝a—func or g—func、 　then begin 　　if　OP (cpr)≒ARG， 　　then 　　cpr ← st (ARG、 cpr、 NIL)； 　　cpr ← st (ope、 opr、 cpr)； 　　PPUP (ope、 opr、 dummy)； 　　end； read (cpe)；
5	／　(　b　BY ／　(　b　WHERE a　,　b　BY a　,　b　WHERE	PDWN (ope、 opr、 NIL)； PDWN (ARG、 cpr、 NIL)； opr ← cpr；　　ope ← cpe； read (cpr)；　　read (cpe)；
6	／　BY　b　BY ／　BY　b　WHERE	PDWN (ope、 cpr、 NIL)； opr ← NIL；　　ope ← cpe； read (cpr)；　　read (cpe)；
7	／　BY　b　) ／　WHERE　b　)	t ← NIL； LOOP： 　　t ← st (ope、 cpr、 t)；

プロセス番号	<opr> <ope> <cpr> <cpe>	処理概要
		PPUP(ope、cpr、dummy)； if ope ≠ '(' , then go to LOOP； cpr ← t； go to process (4)；
8	a , b)	t ← st(ope、cpr、NIL)； cpr ← opr； LOOP： t ← st(ope、cpr、t)； PPUP(ope、cpr、dummy)； if ope ≠ '(' , then go to LOOP； cpr ← t； go to process (4)；
9	／ ⊢ b ⊣	QUAL ← cpr；

read(cpr)；は、文字列からオペランドを判定し、属性ノードあるいは定数ノードを生成して、そのポインタを<cpr>として通知することを示す。

read(cpe)；は、文字列から演算子を判別し、その演算子コード(operator code)を<cpe>として知らせる。

ptr ← st(pe_1、pr_1、pr_2)；は、pr_1をLPTR、pr_2をRPTRとする演算子ノード(pe_1)を生成し、その演算子ノードへのポインタをptrに格納する処理を表わしている。



PPUPおよびPDWNは、それぞれ、スタック領域からのポップアップおよびプッシュダウンを行うサブルーチンを表わしている。

次に、read(cpr)およびread(cpe)内で用いられている、サブルーチンSCANについて述べる。SCANは、文字列から演算子又はオペランドを取り出し、それが演算子のときは対応する演算子コードを返し、オペランド(定数又は属性)のときは、その定数ノード又は属性ノードを生成し、そのノードへのポインタを返す。SCANの状態遷移図を、図8－50に示す。

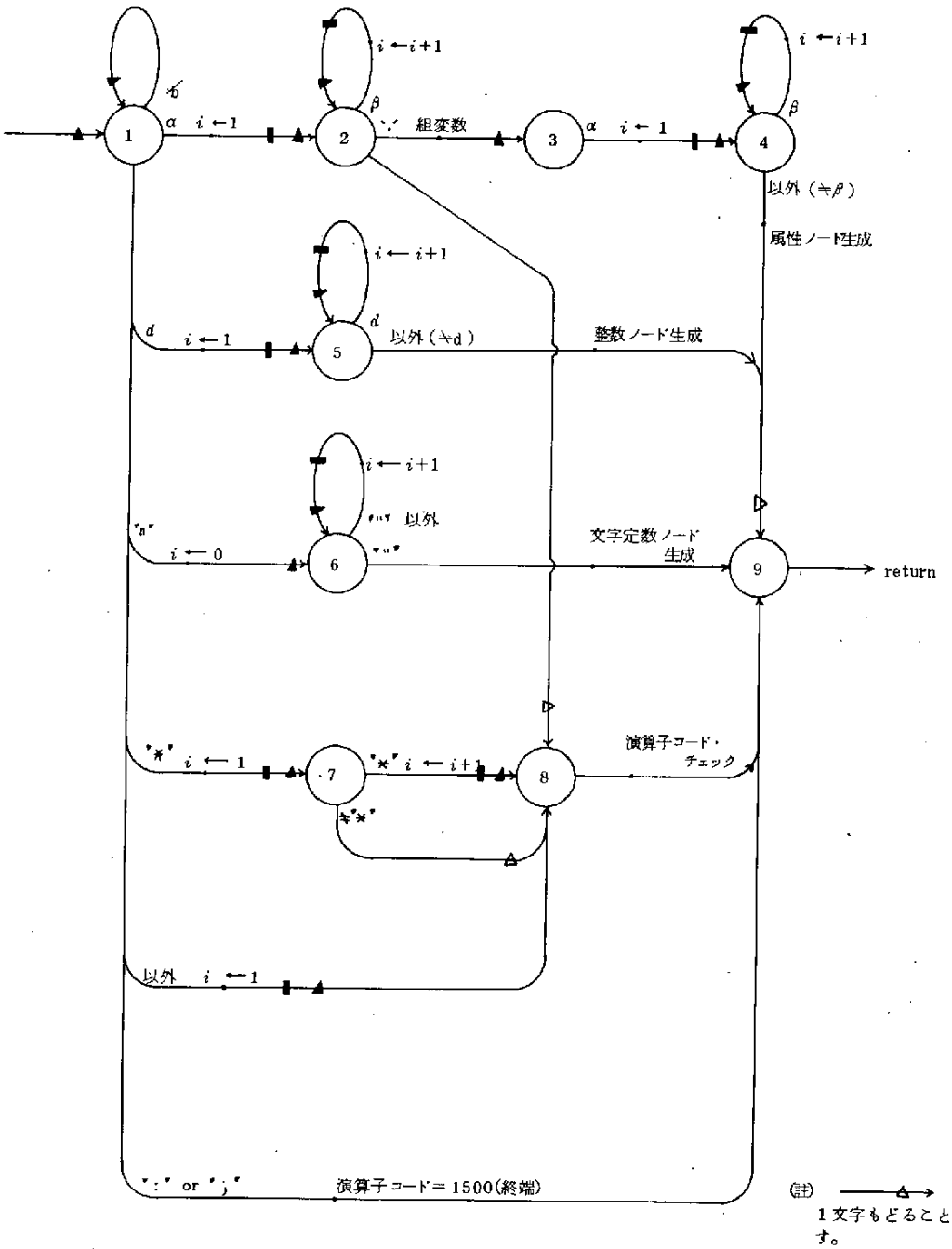


図８－５０　ＳＣＡＮの状態遷移図

表8－11　ATTBLの構成

属性名	属性番号	ROLE	TYPE	LENGTH	説明
VAR	1	K	C	4	tuple variable name
ATTNAME	2	K	C	16	attribute name
ATTNP	3	N	D	4	属性ノードのポインタ

F．QTGの例

QUEL問合せ：

```
@T;
RANGE ( E , P                    )   ( ENGINEER, PROJECT   );
RANGE ( PE, RE, RP, PKL  )   ( PROJ-ENGI, REPR-ENGI, REPR-PROJ,
                                 PROJ-KEYW-LINK );

RETRIEVE INTO  RESULT ( P.PROJNAME , E.ENGINAME   )
WHERE      PKL.KEYWORD   = "DATABASE"      AND  PKL.PROJ-NO   = P.PROJ-NO      AND
           P.PROJ-NO     = RP.PROJ-NO      AND  RP.REPR-NO    = RE.REPR-NO     AND
           RE.ENGINAME   = E.ENGINAME      AND  E.ENGINAME    = PE.ENGINAME    AND
           PE.PROJ-NO    = P.PROJ-NO       AND  P.PROJSYAR  GE 1975            AND
           P.PROJSTAT    = "ON" ;
```

RTBL、RATBLは、次のように生成される。

RTBL（range table）

SEQ	VAR	RELNAME
1	E	ENGINEER
2	P	PROJECT
3	PE	PROJ-ENGI
4	RE	REPR-ENGI
5	RP	REPR-PROJ
6	PKL	PROJ-KEYW-LINK

RATBL（result － attribute table）

SEQ	RELNAME	ATTNO	ATTNAME	ROLE	TYPE	LNG	DOMAIN
1	PROJECT	2	PROJNAME	N	1	20	PROJNAME
2	ENGINEER	1	ENGINAME	N	1	20	ENGINAME

また、ATTBLは次のように生成され、ある属性については属性ノードが１つしか生成されないようにしている。

ATTBL（attribute table）

SEQ	VAR	ATTNAME	ATTNP
1	P	PROJNAME	1
2	E	ENGINAME	3
3	PKL	KEYWORD	5
4	PKL	PROJ-NO	9
5	P	PROJ-NO	10
6	RP	PROJ-NO	12
7	RP	REPR-NO	14
8	RE	REPR-NO	15
9	RE	ENGINAME	17
10	PE	ENGINAME	19
11	PE	PROJ-NO	21
12	P	PROJSYAR	23
13	P	PROJSTAT	25

Q－treeは、図8－51のようになる。

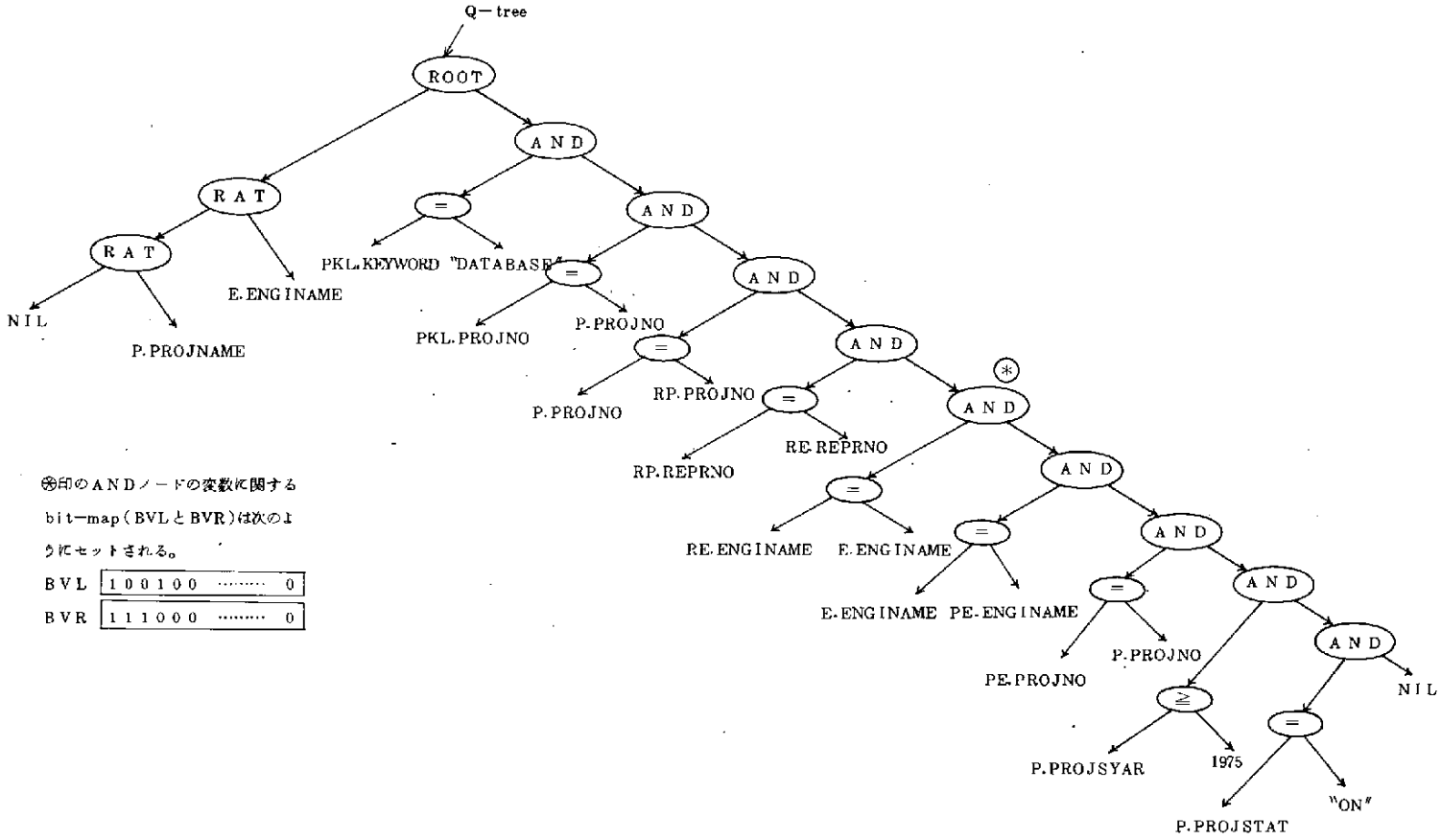


図８－５１　Ｑ－ｔｒｅｅの例

8.2.4 QDBTGP

ＱＤＢＴＧＰは、木構造で表わされた問合せ（Ｑ－ｔｒｅｅ）から、ＤＢＴＧモデルの構文構造に基づいた問合せ表現（ＱＤＢＴＧＲ）を生成するプロセスである〔図８－５２〕。ＱＤＢＴＧＲは、6.3.3におけるＤＢＴＧ問合せグラフ（ＤＱＧ）に対応している。また、ＱＤＢＴＧＲは、ＤＢＴＧ固有の構文構造によって問合せを非手続き的に表わしたものである。

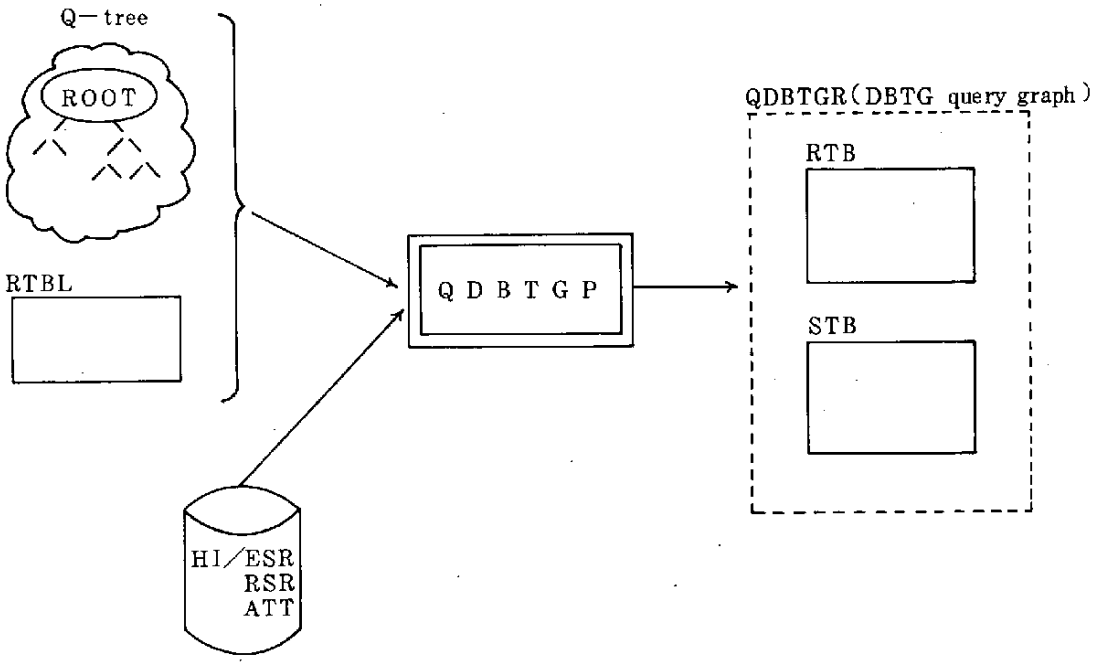


図８－５２　ＱＤＢＴＧＰ の 概 要

A．ＱＤＢＴＧＰの構成

ＱＤＢＴＧＰは、ＶＢＭＧ、ＢＭＧ、ＲＴＢＧ、ＲＳＴＢＧの、４つのサブ・プロセスから成っている〔図８－５３〕。

ＶＢＭＧは、Ｑ－ｔｒｅｅのＲＯＯＴノード及びＡＮＤノードにチェインされているｂｉｔ－ｍａｐ〔8.2.2〕をもとにして、各節（ｃｌａｕｓｅ）とその節の参照する変数との対応表をＶＢＭテーブルとして生成する。

ＢＭＧは、ＶＢＭテーブルの各節を、ｔａｒｇｅｔ－ｌｉｓｔ、ｒｅｓｔｒｉｃｔｉｏｎ ｃｌａｕｓｅ、ｊｏｉｎ ｃｌａｕｓｅ に分類する。これによって作られたテーブルを、それぞれ、ＴＢＭ（ｔａｒｇｅｔ－ｌｉｓｔ ｂｉｔ－ｍａｐ）、ＲＢＭ（ｒｅｓｔｒｉｃｔｉｏｎ ｂｉｔ－ｍａｐ）、ＪＢＭ（ｊｏｉｎ ｂｉｔ－ｍａｐ）と呼ぶ。更に、ＲＢＭとＪＢＭで同一の変数を参照する節を接合（ｃｏｎｊｕｎｃｔｉｏｎ）する。

ＲＴＢＧは、こうして作られたｂｉｔ－ｍａｐ とＲＴＢＬをもとにして、この問合せ内に現われる

変数に対応するレコード・タイプを明らかにする。更に、各レコード・タイプに関連する制限（restriction）と結果属性（result－attribute）及び、ＱＯＰＴＰで使用される選択度（selectivity）、カーディナリティ等のパフォーマンス情報も明らかにする。

ＲＳＴＢＧは、主に、隠れ構造（hidden structure：ＩＨＳとＥＨＳ）〔5.3.2を参照〕を明らかにし、完全なＲＴＢ（record－type table）とＳＴＢ（set－type table）を生成する。ＲＴＢはレコード型についての情報（名前、カーディナリティ、条件式等）を保持し、ＳＴＢはセット型（親子レコード型）についての情報を保持している。この２つによって、レコード型をノード、セット型をリンクとするグラフを表わしている。このＲＴＢとＳＴＢの２つのテーブルを総称してＱＤＢＴＧＲと呼ぶ。join clauseに関する情報が格納されているＪＢＭをアクセスしながらＲＴＢからセット型を除去し、ＥＨＳを加え、ＳＴＢにＩＨＳ及びＲＴＢからのセット型を加えてＱＤＢＴＧＲを生成する。

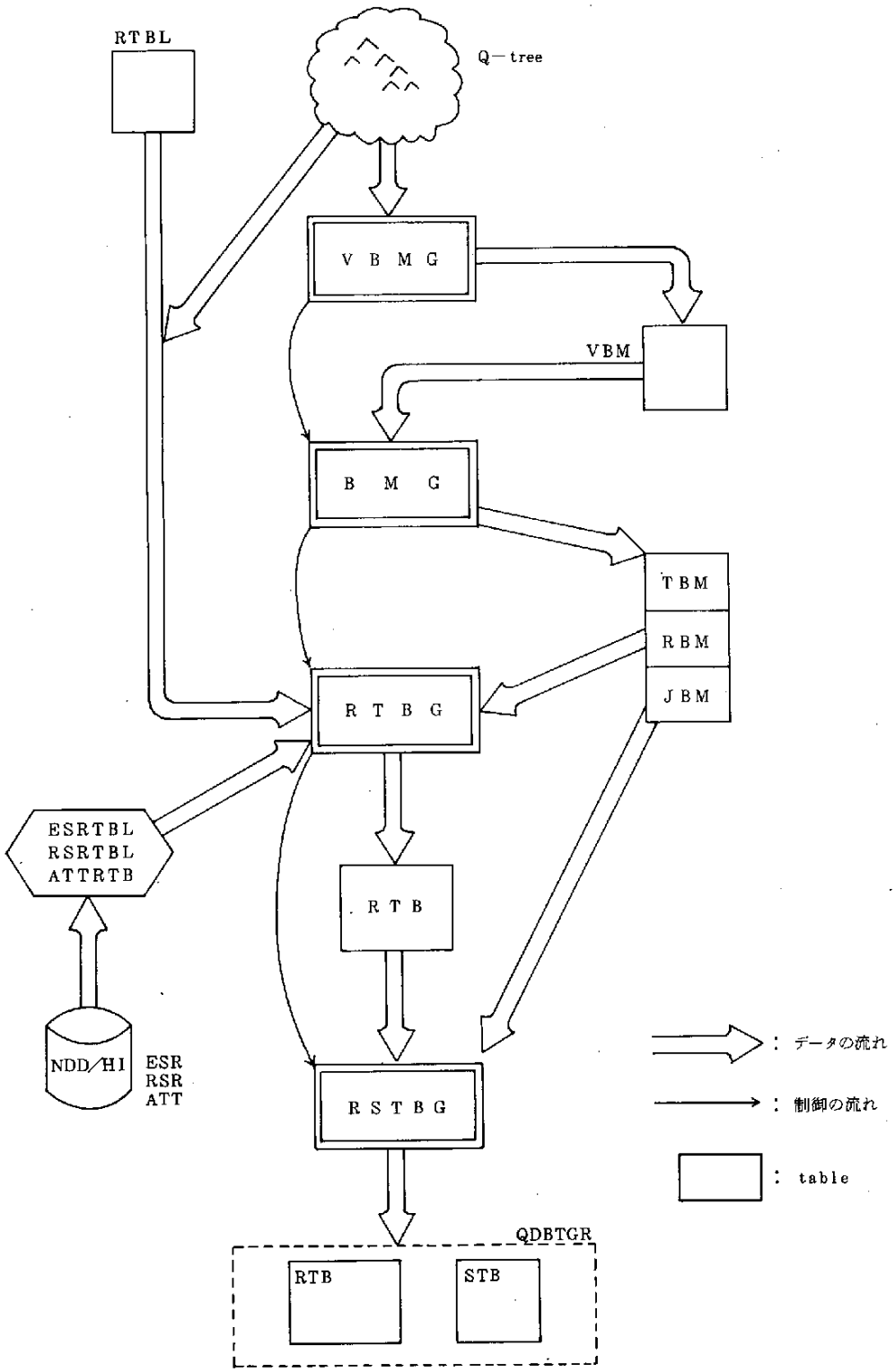


図８－５３　ＱＤＢＴＧＰ の 構 成

B. ＱＤＢＴＧＲ

ＱＤＢＴＧＲは、問合せのＤＢＴＧモデルによる非続き的な表現であり、ＲＴＢとＳＴＢと呼ぶ２つのテーブルより構成される。ＲＴＢは、問合せ内のＤＢＴＧのレコード型に関する情報（名前、カーディナリティ、条件式）についてのテーブル表現である。ＳＴＢは、セット型に関する情報（親子レコード型）についてのテーブル表現である。

表８－１２　ＲＴＢの構成

属　性　名	属性番号	ROLE	TYPE	LENGTH	説　　明
ＲＴＹＰＥ	1	K	C	16	レコード型の名前
ＣＡＲＤ	2	N	D	4	レコード型の全オカーレンス数
ＬＩＮＫ	3	N	D	2	normal　（＝０） link－type（＝１） set　－type（＝２）
ＣＲＳＴＲ	4	N	D	2	ＣＡＬＣ項目をもつ equi － restriction tree へのポインタ。なければ、ＮＩＬ
ＲＳＴＲ	5	N	D	2	ＣＲＳＴＲを除いた restriction tree へのポインタ。なければ、ＮＩＬ
ＣＳＬＣＴ	6	N	F	4	ＣＲＳＴＲの選択度　（≦１） ＣＲＳＴＲ＝ＮＩＬなら　１
ＲＳＬＣＴ	7	N	F	4	ＲＳＴＲの選択度　（≦１） ＲＳＴＲ＝ＮＩＬなら　１
ＲＳＬＴ	8	N	D	2	結果属性リストへのポインタ
ＯＣＡ	9	N	D	4	レコード型内でアクセスされるべきオカーレンス数
ＮＰＴＲ	１０	N	D	2	ＮＩＴノードへのポインタ
ＵＳＥＤ	１１	N	D	2	ＯＮ　（＝１）：この tuple は使用中 ＯＦＦ（＝０）：この tuple は未使用
ＭＡＲＫ	１２	N	D	2	ＣＮ　（≧２）：合流ノード ＮＮ　（＝１）：合流ノード以外のノード ＱＯＰＴＰで使用される。

表8－13　ＳＴＢの構成

属性名	属性番号	ROLE	TYPE	LENGTH	説明
STYPE	1	K	C	16	セット型の名前
CARD	2	N	D	4	セット型の親レコード型の1つのオカーレンスがもつ平均の子のオカーレンス数
OREC	3	N	D	2	親のレコード型が格納されているＲＴＢのtuple番号
MREC	4	N	D	2	子のレコード型が格納されているＲＴＢのtuple番号
MARK	5	N	D	2	ON（＝1）：このセット型は、もうアクセス・パス内にセットされた。 OFF（＝0）：まだセットされていない。

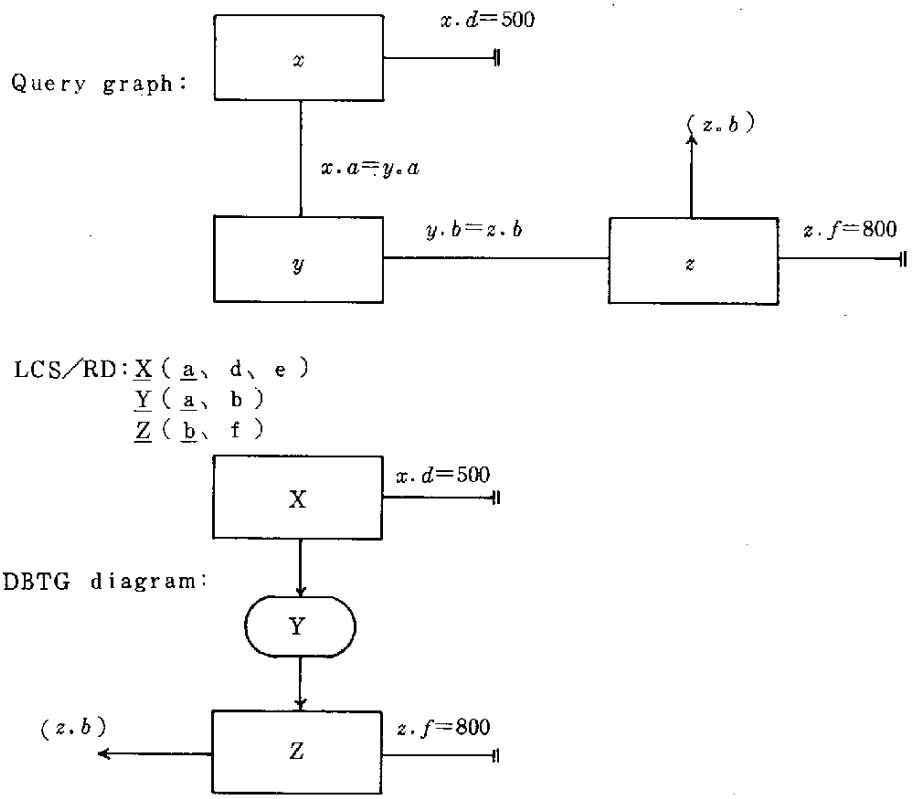


図8－54　ＲＴＢとＳＴＢの例(1)

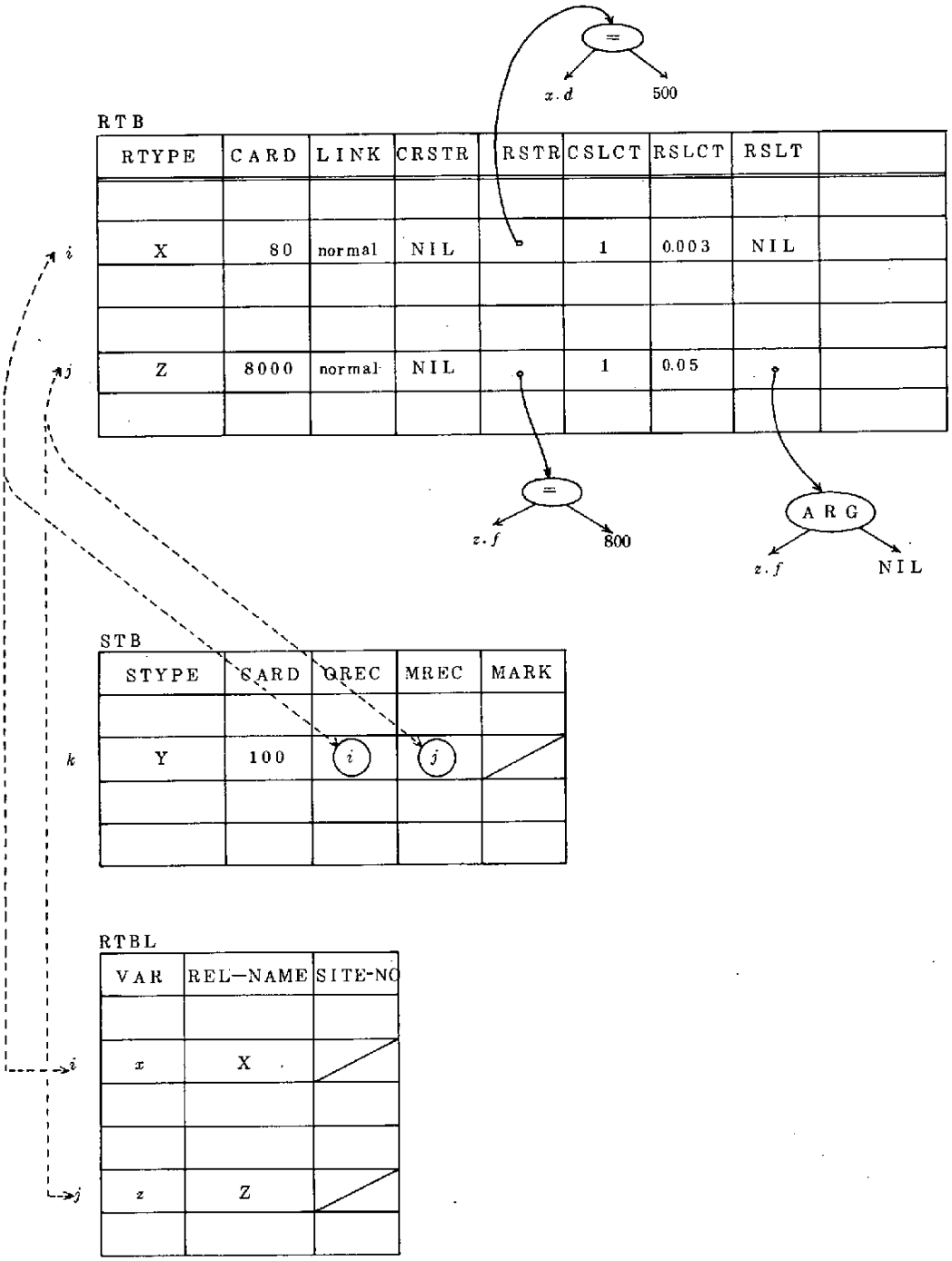


図8－55　RTBとSTBの例 (2)

C．ＶＢＭＧ

ＶＢＭＧは、Ｑ－ｔｒｅｅのＲＯＯＴノード及びＡＮＤノードにチェインされている変数に関するｂｉｔ－ｍａｐ（ＢＶＬ）を用いてＶＢＭテーブルを生成するプロセスである。

ＶＢＭテーブルは、<target－list>及び<qualification>内の各<conjunct>と、これが参照する組変数との対応表である。ＶＢＭの構成は、表8－14に示す。即ち、ＶＢＭの１つの組（tuple）は、対応する節（clause）のＲＯＯＴＮ、又は、ＡＮＤＮにチェインされたbit－map（ＢＶＬ）を表わし、ＰＴＲには、このＲＯＯＴＮ又はＡＮＤＮの左手の部分木（subtree）へのポインタがセットされる。ＶＢＬは３２ｂｉｔからなり、各ｂｉｔの位置はＲＴＢＬ(range table)の組番号に対応している。これにより、問合せの参照し得る組変数（tuple variable）数は、３２までとなる。

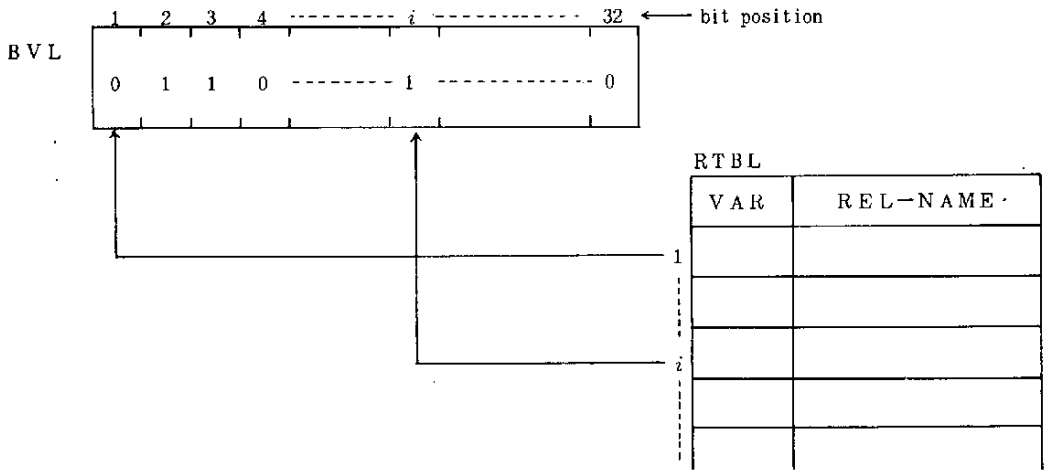


図８－５６　ＲＴＢＬとＢＶＬのビット位置の対応

表８－１４　ＶＢＭの構成

属性名	属性番号	ROLE	TYPE	LENGTH	説明
PTR	1	N	D	2	部分木へのポインタ
VBL	2	N	C	4	部分木の変数（variable）についてのbit－map
MARK	3	N	D	2	

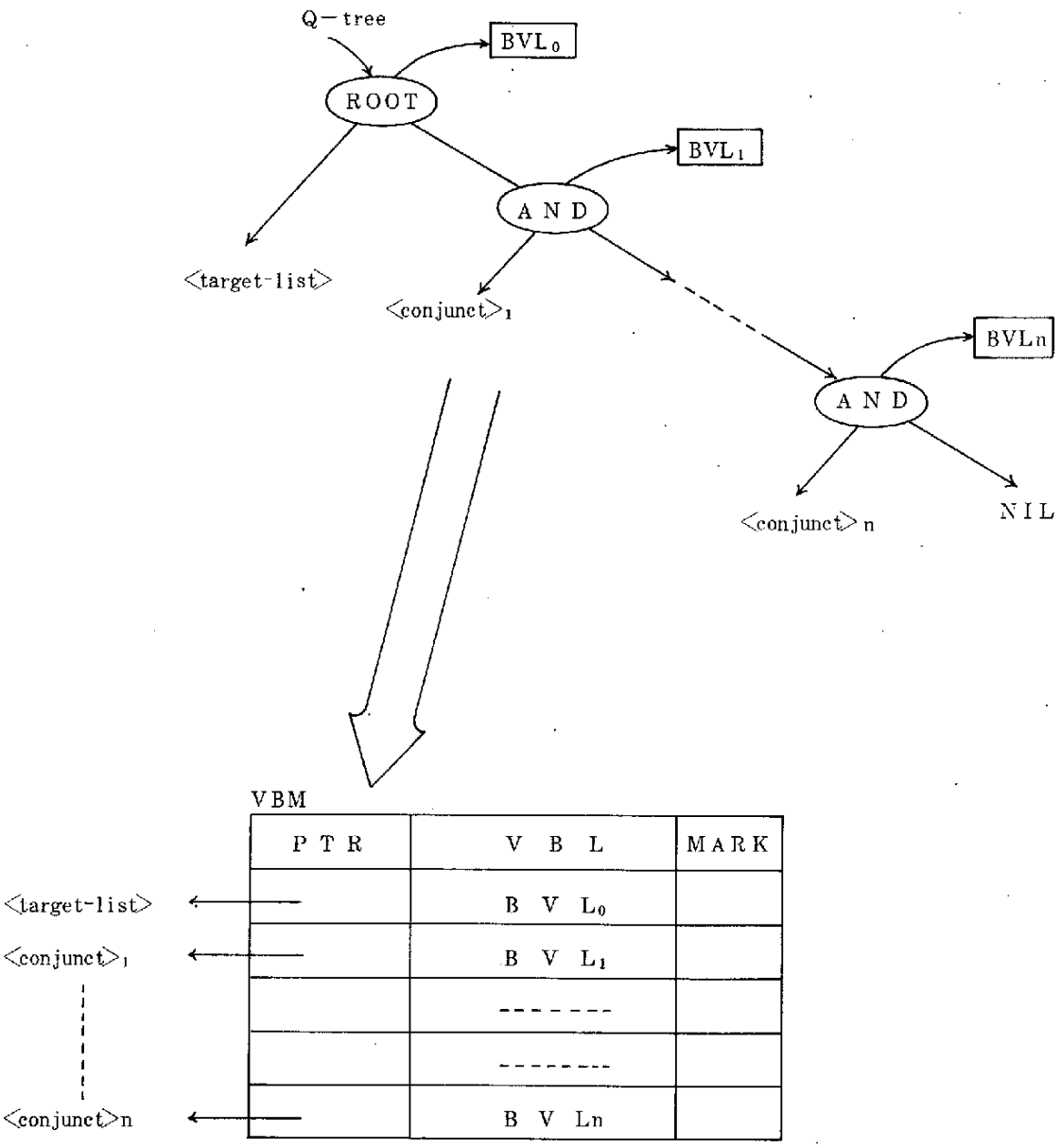


図８－５７　ＶＢＭＧの処理概要

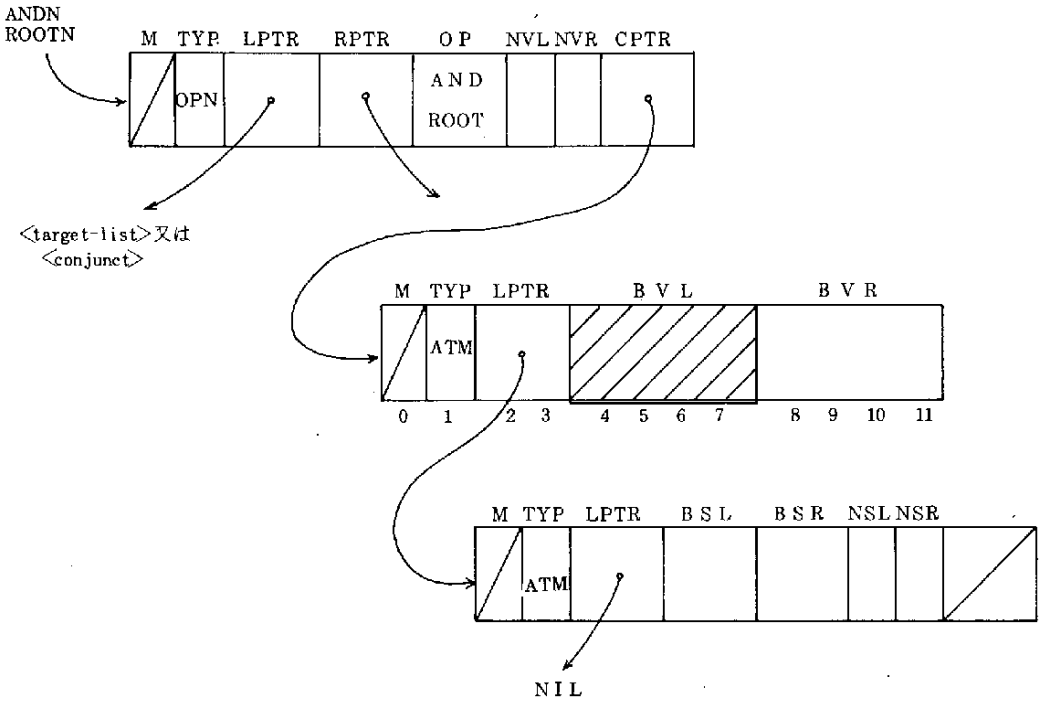


図８－５８　ＲＯＯＴＮとＡＮＤＮの構成

Ｄ．ＢＭＧ

ＢＭＧは、ＶＢＭテーブルを入力として、各<conjunct>を制限節（restriction clause）と結合節（join clause）に分類し、それぞれ、ＲＢＭ、ＪＢＭとしてまとめる。また、<target－list>からＴＢＭを作る。あるＶＢＭの組（tuple）のＶＢＬで、ＯＮであるビットの数が１つであるとき、その節は制限節であり、２つあるときが結合節である。

更に、ＲＢＭ、ＪＢＭで、それぞれ、同一の変数を参照するものの論理積をつくる。

ＴＢＭ、ＲＢＭ、ＪＢＭは、ＶＢＭテーブルの領域内に作られる。即ち、ＶＢＭの先頭の組はそのままＴＢＭとなり、ＲＢＭはＶＢＭの第２番目から順に、ＪＢＭは逆にＶＢＭの底から上へと順に格納される〔図８－５９〕。従って、ＴＢＭ、ＲＢＭ、ＪＢＭの構成はＶＢＭと同じである〔表８－１５〕。また、ＲＢＭ、ＪＢＭとも、同一のｂｉｔ－ｍａｐ（ＶＢＬ）をもつ組が複数個あるときは、対応する節の論理積をつくり、その新しいポインタをＰＴＲに格納する。従って、最終的には同一のＶＢＬをもつ組は１つしかないことになる。

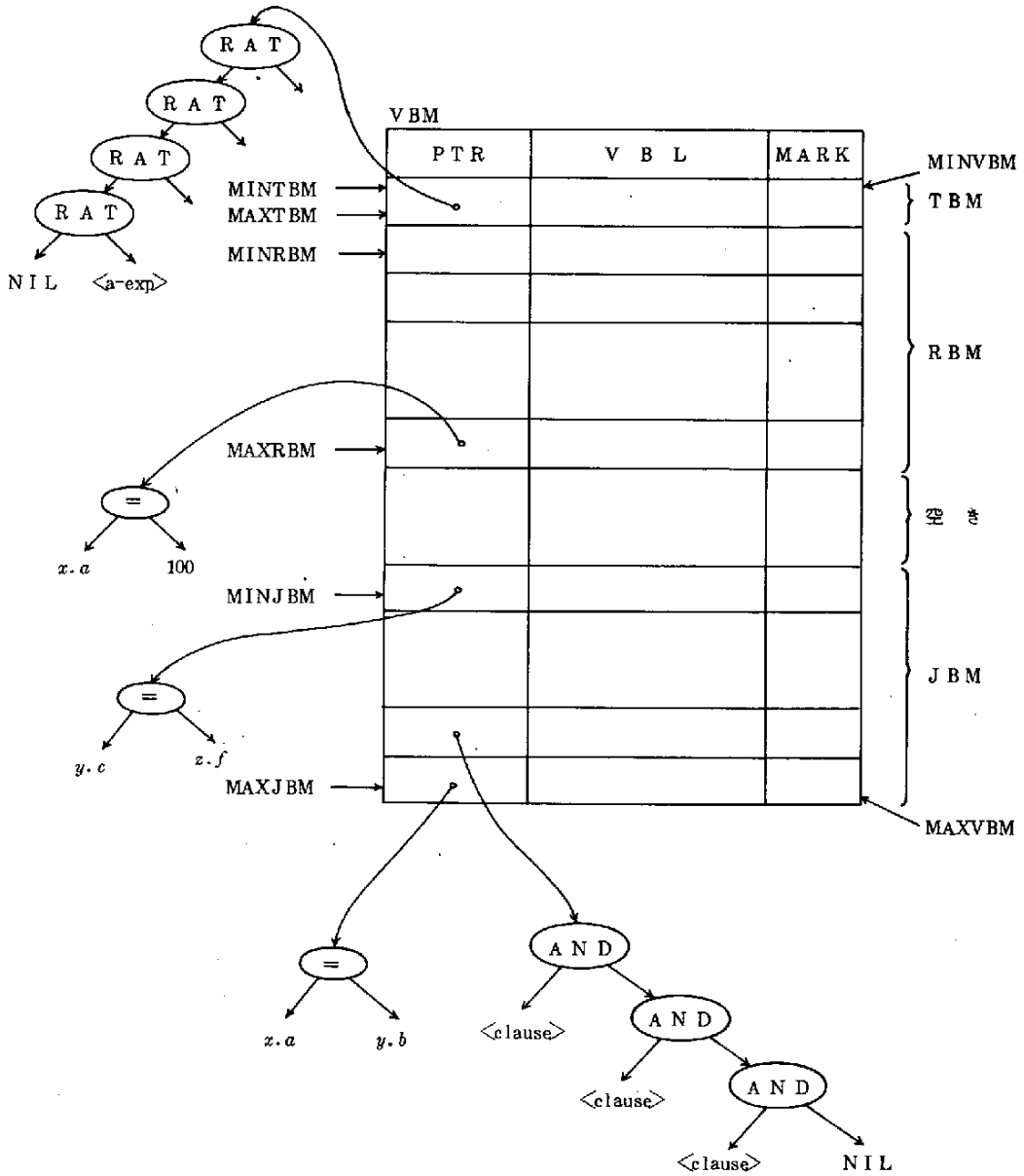


図8－59　TBM、RBM、JBMの構造

表８－１５　ＴＢＭ、ＲＢＭ、ＪＢＭの構成

リレーション名	属性名	属性番号	ROLE	TYPE	LENGTH	説明
ＴＢＭ	ＰＴＲ	1	Ｎ	Ｄ	2	＜target－list＞へのポインタ
	ＶＢＬ	2	Ｎ	Ｃ	4	変数に関するbit－map
	ＭＡＲＫ	3	Ｎ	Ｄ	2	
ＲＢＭ	ＰＴＲ	1	Ｎ	Ｄ	2	制限節（restriction clause）へのポインタ
	ＶＢＬ	2	Ｎ	Ｃ	4	変数に関するbit－map
	ＭＡＲＫ	3	Ｎ	Ｄ	2	
ＪＢＭ	ＰＴＲ	1	Ｎ	Ｄ	2	結合節（join clause）へのポインタ
	ＶＢＬ	2	Ｎ	Ｃ	4	変数に関するbit－map
	ＭＡＲＫ	3	Ｎ	Ｄ	2	

Ｅ．ＲＴＢＧ

ＲＴＢＧは、Ｑ－tree、ＲＴＢＬ、ＲＢＭ及び異種性情報（ＨＩ）〔4.5又は8.1を参照〕をもとにして、問合せで参照される変数に対応するＤＢＴＧの構文構造（レコード型、セット型）をＲＴＢに格納するプロセスである。

問合せ内で参照される全ての変数は、Ｑ－treeのＲＯＯＴノードにチェインされているＢＶＬとＢＶＲの論理和（ＯＲ）をとることによって得られる〔図８－６０〕。即ち、ＶＢ←ＢＶＬ ∨ ＢＶＲ。ここで、ＢＶＬはＲＯＯＴノードの左手の部分木内（＜target－list＞）で参照される変数のbit－map であり、ＢＶＲは右手の部分木（＜qualification＞）で参照される変数のROOTN

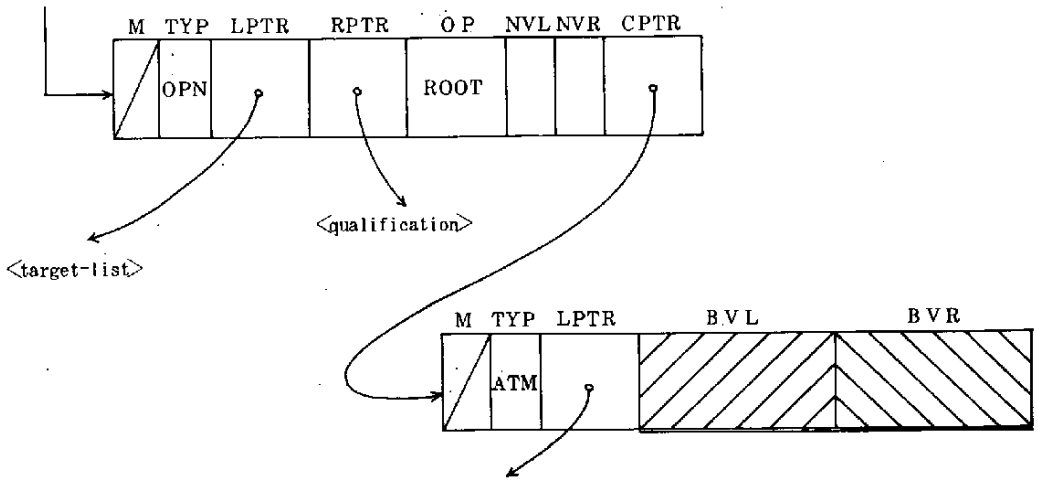


図８－６０　ＢＶＬ と ＢＶＲ

bit－map である。こうして求められたbit－map（ＶＢ）を用いて、ＲＴＢＬより問合せが参照するリレーション名を得ることができる。また、<target－list>内の<a－exp>から属性ノード（ＡＴＴＮ）のみを抽出し、結果属性（result attribute）リストを生成する〔図８－６１〕。

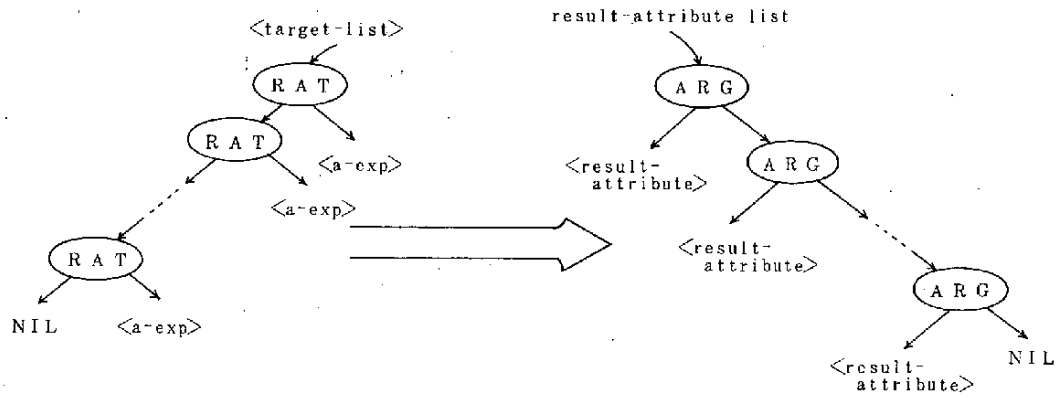


図８－６１　結果属性リストの生成

次に、リレーション名をＲＴＢ内の、bit－map のビット位置に対応した組（tuple）番号に格納する。このとき、ＲＴＢのＬＩＮＫフィールドは、リレーション名がＨＩ／ＥＳＲにあるときはnormal（＝0）とし、ＨＩ／ＲＳＲにあってＨＩ／ＲＳＲのＲＳ－ＴＹＰＥがＳのときセット型（＝２）、Ｓ以外のときlink－type（＝１）とする。また、ＲＴＢのＣＡＲＤフィールドには、それぞれ、ＨＩ／ＥＳＲ又はＨＩ／ＲＳＲのカーディナリティをセットする。

次に、ＲＴＢをサーチし、そのリレーションについての制限節（restriction clause）を作る。これは、ＲＢＭのＶＢＬで同一のビット位置がＯＮの組（tuple）をみつけることにより得られる。このとき、そのＰＴＲフィールドが制限節の木（tree）へのポインタとなっている。制限木（restriction tree）をサーチし、ＣＡＬＣを参照する等価制限節（equi－restriction）のグループとその他のグループに木を分離し、それぞれ、ＲＴＢのＣＲＳＴＲとＲＳＴＲにセットする。同時に、属性ノードの属性番号とリレーション名をもとにＨＩ／ＡＴＴをサーチし、その選択度（selectivity）を用いてＣＲＳＴＲとＲＳＴＲの選択度を計算して、ＣＳＬＣＴとＲＳＬＣＴにセットする。

結果属性リスト（result－attribute list）内の属性ノードからこのリレーションに属するものを抽出して１つの木とし、そのポインタをＲＴＢのＲＳＬＴフィールドに格納する。これは、bit－map（ＶＢ）のビット位置がＲＴＢＬの組番号に等しく、また、属性ノードのＲＰＴＲにＲＴＢＬへのポインタが入っていることにより対応づけられる〔図８－６２〕。

こうして生成されたＲＴＢは、問合せ内に現われた全てのリレーション（ＥＳＲとＲＳＲ）と、

このリレーションに関連する情報とを格納していることになる。次のＲＳＴＢＧで、これらのリレーションのうち、レコード型のみをＲＴＢに残し、セット型はＳＴＢに移すことを行う。

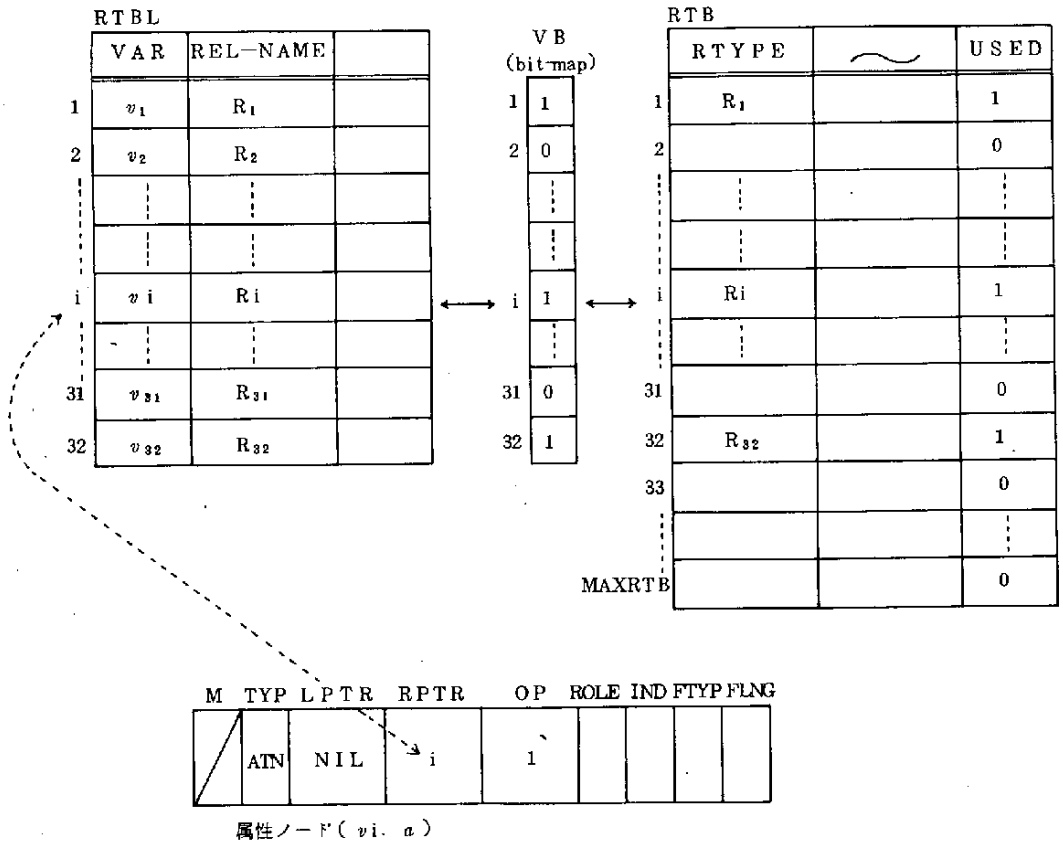


図８－６２　ＲＴＢとＲＴＢＬの関係

Ｆ．ＲＳＴＢＧ

ＲＳＴ-ＢＧは、ＪＢＭ、ＲＴＢ及び異種性情報（ＨＩ）を用いて隠れ構造（hidden structure：ＥＨＳとＩＨＳ）を明らかにしてＱＤＢＴＧＲ（即ち、6.3.3のＤＱＧ）を生成するプロセスである。

ＪＢＭ（join bit－map）は、レコード型の間の join－link を表わしている。

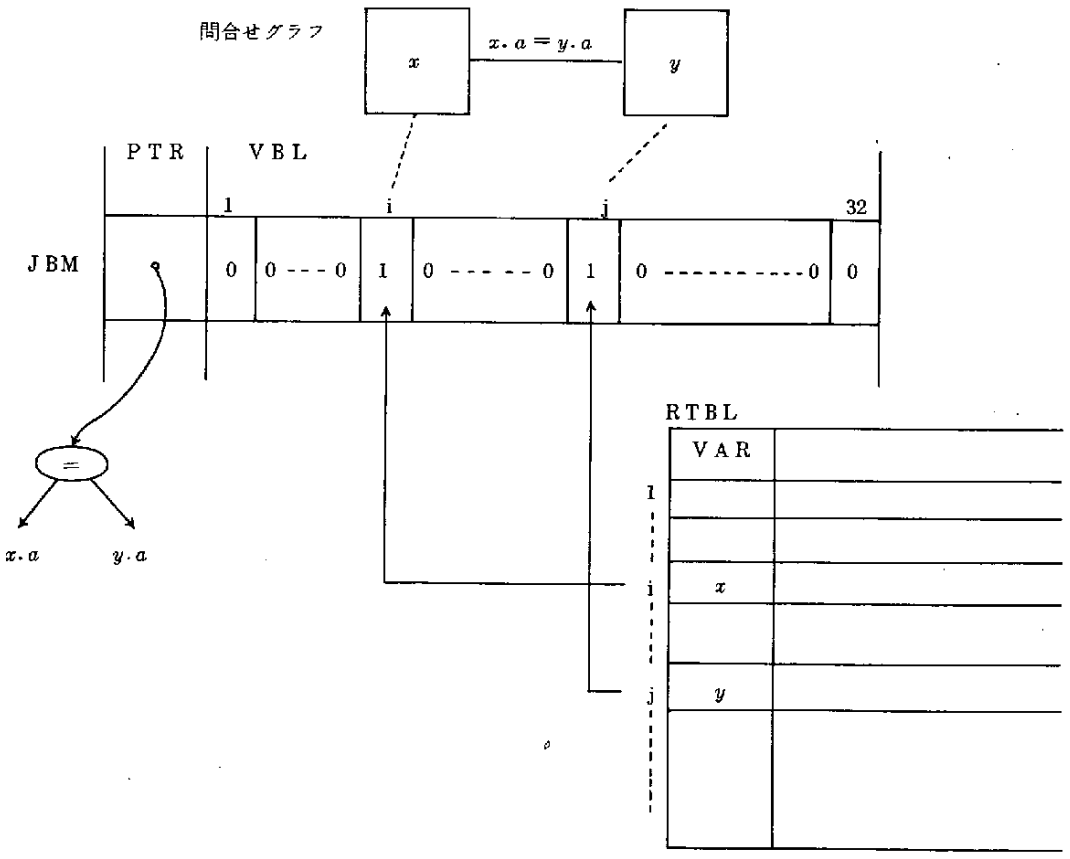


図８－６３　ＪＢＭの構造

ｘとｙはそれぞれ、ＥＳＲかＲＳＲかであり、ｘかｙのどちらか一方は必ずＲＳＲである。従って、結合のパターンとしては、

(1)　ＥＳＲ　－　ＲＳＲ

(2)　ＲＳＲ　－　ＥＳＲ

(3)　ＲＳＲ　－　ＲＳＲ

の３つが考えられる。ＲＳＲ－ＲＳＲの場合には明らかにＥＨＳ（explicitly hidden structure）〔5.3.2を参照〕が存在しているので、ＨＩ／ＲＳＲを用いて２つのＲＳＲ間に存在し得るレコード型（ＥＳＲ）を明らかにする。

手順としては、ＪＢＭをシーケンシャルにアクセスしながら、はじめに、(1)ＥＳＲ－ＲＳＲと(2)ＲＳＲ－ＥＳＲのパターンに対してのみ処理を行う。(1)と(2)の処理後、未処理として残される(3)ＲＳＲ－ＲＳＲの処理を行う。

このようにして生成されたＲＴＢとＳＴＢには、それぞれ、ＤＢＴＧレコード型と、これに関する制限（restriction）、結果属性（result attribute）等の情報と、ＤＢＴＧセット型とこれに関

する情報とが格納されている。これは、局所内部スキーマ（ＬＩＳ）のＤＢＴＧ図式の部分集合に対応している。従って、このＱＤＢＴＧＲはＤＢＴＧ構文構造に基づいた問合せの非手続き的な表現となっている。

ａ）　ＲＳＴＢＧの処理概要

１）〔ＪＢＭを先頭組（tuple）から順にサーチ〕

ＪＢＭをシーケンシャルにアクセスして、join－link のはられている変数を明らかにする。ここで、ＪＢＭのＶＢＬのｉ番目とｊ番目のビットがＯＮであったとする。

２）〔ＥＳＲかＲＳＲかを決定〕

ｉ番目（あるいはｊ番目）のＲＴＢの組（tuple）のＬＩＮＫフィールドがnormal（＝0）ならＥＳＲ、set－type（＝2）、link－type（＝1）ならＲＳＲである。

３）〔ＥＳＲ－ＲＳＲの処理：e－r－kkなるkkをみつける。〕

まず、ＥＳＲ－ＲＳＲの場合を処理する。ｉ番目がＥＳＲ、ｊ番目がＲＳＲとする。逆の場合も同様に処理する。また、ＥＳＲ、ＲＳＲのリレーション名を、それぞれ、ｅ、ｒとする。ＪＢＭをサーチして、ｅ－ｒ－kkなるkkをみつける。即ち、ＪＢＭの中でＶＢＬのｉ番目のビット（ｒに対応）がＯＮであるtupleをみつけ、そのＶＢＬの他のＯＮとなっているビット（kkに対応）をみつける。

４）〔ｋｋはＥＳＲ〕

そのようなtupleがＪＢＭにあったら、kkがＥＳＲかＲＳＲかをＲＴＢのＬＩＮＫフィールドにより知る。kkがnormalなら、ＥＳＲ－ＲＳＲ－ＥＳＲのトリプレットからＤＢＴＧ構文構造に変換してＳＴＢに格納する。

５）〔ｋｋはＲＳＲ〕

kkがset－type、link－typeなら、kkについてＨＩ／ＲＳＲをサーチし、ｒ－ｍ－kkなるＥＳＲ　ｍをみつけＲＴＢに格納する。また、kkとｒの制限と結果属性に関するもののうち、ｍに関するものをｍに移す。ｅ－ｒ－ｍをＳＴＢに格納する。

６）〔ｅ－ｒ－kkなるkkがＪＢＭに見つからない〕

ＪＢＭにｅ－ｒ－kkなるkkがなかったとき、ＨＩ／ＲＳＲをアクセスしてｅ－ｒ－ｍなるＥＳＲ　ｍをみつけ、ＲＴＢに格納する。同時に、ｒのrestrictionとresult－attributeのうち、ｍに関するものをｍに移す。ｒをＳＴＢに格納する。

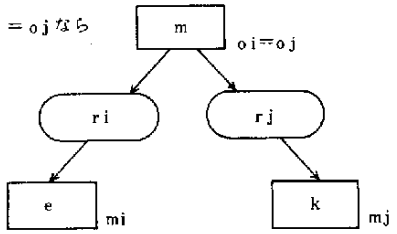
７）〔ＲＳＲ－ＲＳＲの処理〕

ＥＳＲ－ＲＳＲとＲＳＲ－ＥＳＲの処理が終ったら、もう一度ＪＢＭをサーチして、ＲＳＲ－ＲＳＲ（ｒｉ－ｒｊ）なる結合節（join clause）をみつける。ＨＩ／ＲＳＲをサーチして、ｒｉ－ｍ－ｒｊなるＥＳＲ　ｍを求め、ri－mとrj－mというjoin－linkをＪＢＭに追加して１）の処理へ戻る。

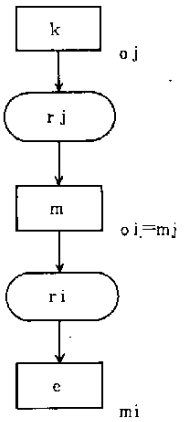
ｒｉとｒｊのsource ＥＳを、各々、ｏｉ、ｏｊ、destination ＥＳを各々ｍｉ、ｍｊとするとき、図８－６４のようになる。この組合せ以外はエラーとなる。これから、ｒｉ－ｍ、ｒｊ－ｍというＥＳＲ－ＲＳＲのjoin clause　をＪＢＭに追加し処理する。

概略のフローチャートを、図８－６５～図８－６７に示す。

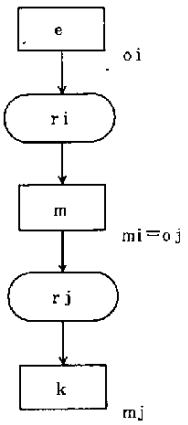
1) oi＝ojなら



2) oi＝mjなら



3) mi＝ojなら



4) mi＝mjなら

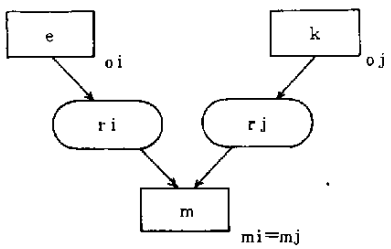


図８－６４　oi－ri－miとoj－rj－mjの関係

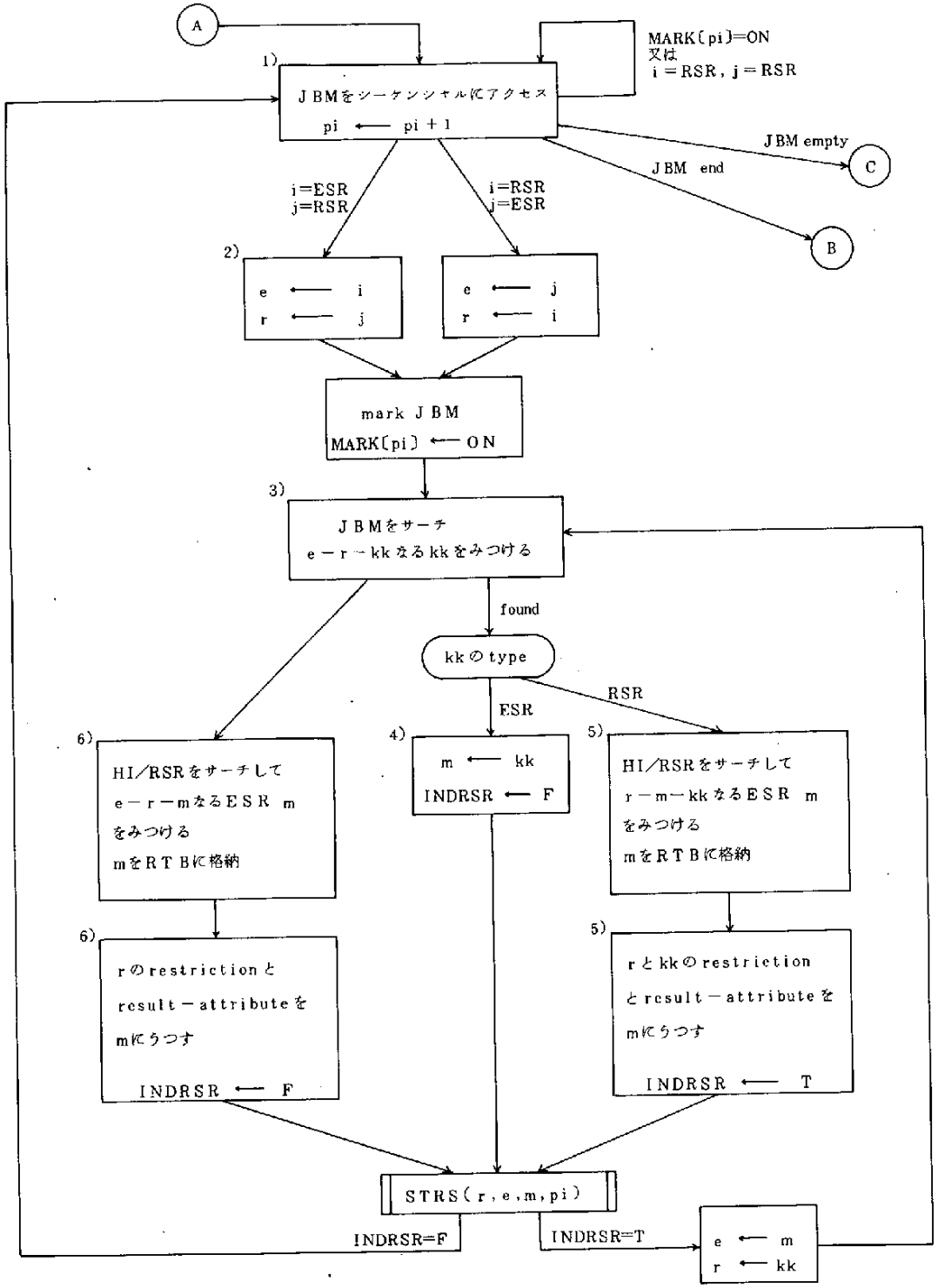


図８－６５　ＲＳＴＢＧの処理概要 (1)

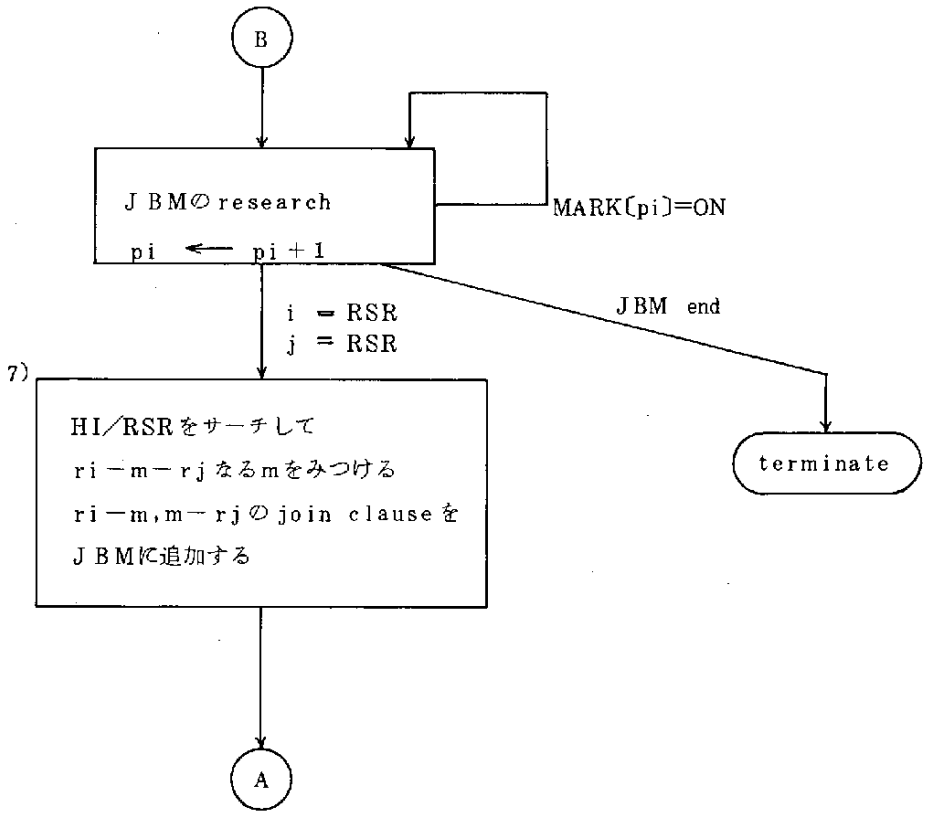


図８－６６　ＲＳＴＢＧの処理概要 (2)

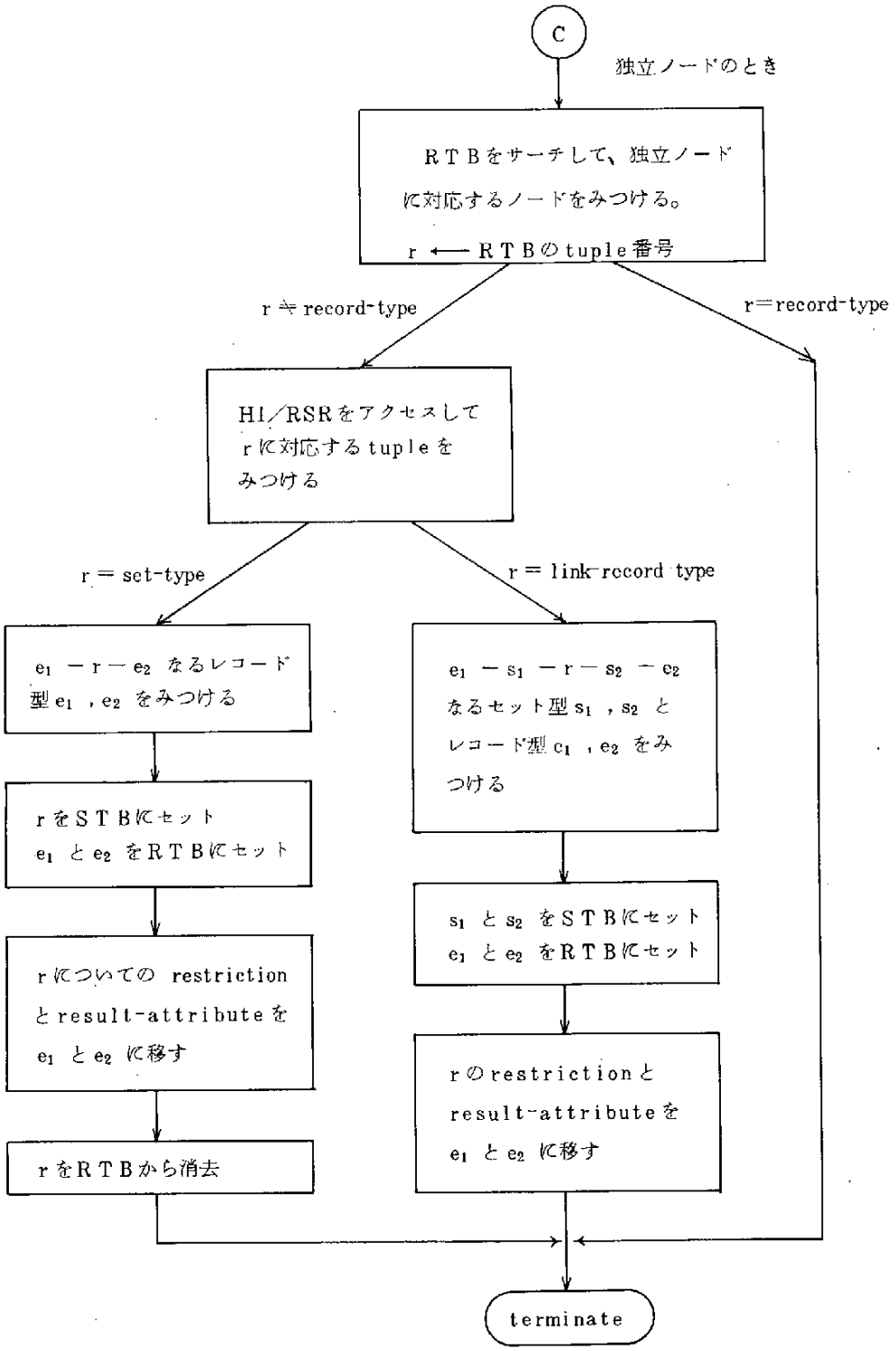


図８－６７　ＲＳＴＢＧの処理概要 (3) ： 独立ノードの処理

G．ＱＤＢＴＧＲの例

8.2.3（Ｅ）の例にもとづいて、ＱＤＢＴＧＲ（ＳＴＢとＲＴＢ）が生成される様子を以下に示す。

ＱＵＥＬ問合せ：

```
@T;
RANGE ( E , P                )   ( ENGINEER, PROJECT   );
RANGE ( PE, RE, RP, PKL      )   ( PROJ-ENGI, REPR-ENGI, REPR-PROJ,
                                   PROJ-KEYW-LINK );
RETRIEVE INTO  RESULT ( P.PROJNAME , E.ENGINAME  )
 WHERE    PKL.KEYWORD  = "DATABASE"      AND  PKL.PROJ-NO  = P.PROJ-NO      AND
          P.PROJ-NO    = RP.PROJ-NO      AND  RP.REPR-NO   = RE.REPR-NO     AND
          RE.ENGINAME  = E.ENGINAME      AND  E.ENGINAME   = PE.ENGINAME    AND
          PE.PROJ-NO   = P.PROJ-NO       AND  P.PROJSYAR  GE 1975           AND
          P.PROJSTAT   = "ON";
```

ＱＴＧにより、ＲＴＢＬは次のように作成されている。

ＲＴＢＬ（range table）

SEQ	VAR	RELNAME
1	E	ENGINEER
2	P	PROJECT
3	PE	PROJ-ENGI
4	RE	REPR-ENGI
5	RP	REPR-PROJ
6	PKL	PROJ-KEYW-LINK

ＱＤＢＴＧＰは、ＳＴＢとＲＴＢを次のように生成する。

ＳＴＢ

SEQ	SET-TYPE	CARD	OREC	MREC	MARK
1	PROJ-ENGI	5	2	1	1
2	REPR-ENGI	5	7	1	1
3	REPR-PROJ	5	7	2	1
4	PROJ-KEYW	10	2	6	1
5	KEYW-PROJ	1	8	6	1
6		0	0	0	0

ＲＴＢ

SEQ	RECORD-TYPE	CARD	LINK	CRSTR	RSTR	CSLCT	RSLCT	RSLT	DCA
1	ENGINEER	20	0	0	0	1.000	1.000	62	20
2	PROJECT	50	0	0	64	1.000	0.050	65	50
3	PROJ-ENGI	5	2	0	0	1.000	1.000	0	5
4	REPR-ENGI	5	2	0	0	1.000	1.000	0	5
5	REPR-PROJ	5	2	0	0	1.000	1.000	0	5
6	PROJ-KEYW-LINK	10	1	0	0	1.000	1.000	0	10
7	REPRESENTATIVE	10	0	0	0	1.000	1.000	0	10
8	KEYWORDS	20	0	8	0	0.200	1.000	0	4
9		0	0	0	0	1.000	1.000	0	0

ここで、ＲＴＢＬ内に無く、ＲＴＢ内にあるＲＥＰＲＥＳＥＮＴＡＴＩＶＥとＫＥＹＷＯＲＤＳは、ＲＳＴＢＧにより見つけられた隠れ構造である。

8.2.5 QOPTP

ＱＯＰＴＰは、ＱＤＢＴＧＰによって生成された問合せのＤＢＴＧ表現（ＲＴＢとＳＴＢ）から、ヒューリスティックス（heuristics）によって、アクセスされるオカーレンス数をより少くするようなアクセスパスを決定するプロセスである。

ＱＤＢＴＧＲは、局所問合せ内の隠れ構造（hidden structure）を明らかにし、各レコード型に関する制限節（restriction clause）、結果属性（result－attribute）及び、各レコード型を結びつけるセット型を明らかにしたものである。従って、このＱＤＢＴＧＲからいくつかのアクセスパスが考えられるが、ＱＯＰＴＰはこのうちで適当なアクセスパスを決定する。この決定は、アクセスするレコード型のオカーレンス数が最小となるようになされる〔図８－６８〕。

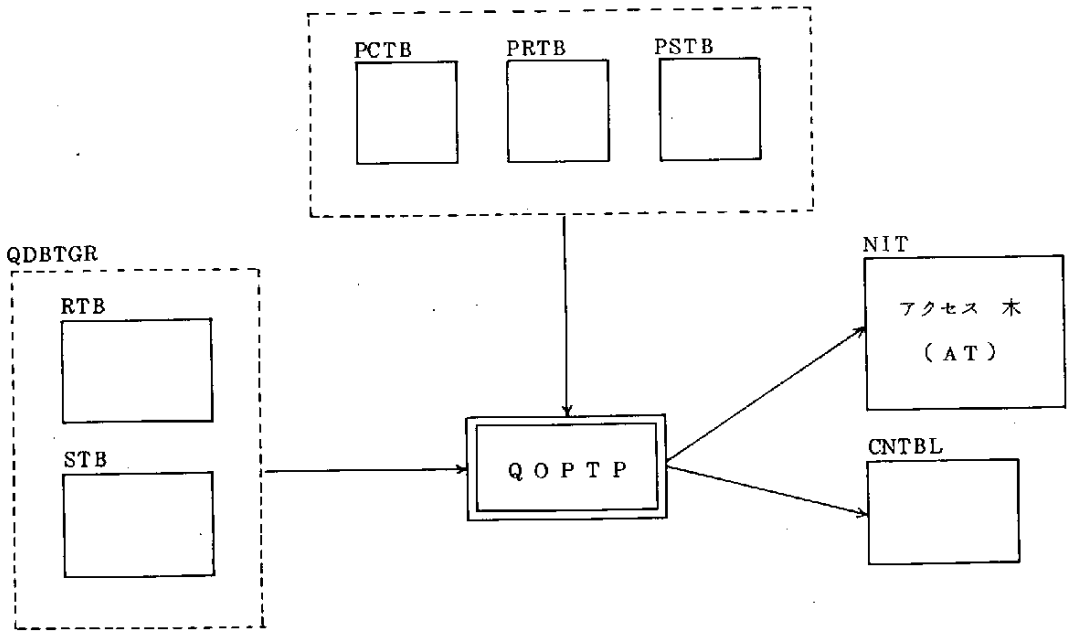


図８－６８　ＱＯＰＴＰ の 概 要

アクセスのシーケンスは、ＮＩＴ内に木構造として表わされる。アクセスパスのこの木構造表現をアクセス木（ＡＴ）と呼ぶ。アクセスパスはこの２分木をhierarchical orderに基づいてアクセスすることにより得られる。

更に、ＱＯＰＴＰは頻繁に用いられるアクセスパターンについては、ＱＤＢＴＧＲからＮＩＴを生成し、そのパターン番号を次のプロセスに知らせる。

A．ＱＯＰＴＰの構成

ＱＯＰＴＰは、ＰＭＡＴＣＨとＯＰＴＰの２つのサブ・プロセスから構成される〔図８－６９〕。ＰＭＡＴＣＨは、ＰＴＣＢ、ＰＲＴＢ、ＰＳＴＢの３つのテーブル内に格納されているアクセスパターンとの照合を行う。また、ＯＰＴＰは、ＰＭＡＴＣＨの照合でアクセスパターンが見つからなかったとき、ＱＤＢＴＧＲから最適なアクセスパスを決定するサブプロセスである。

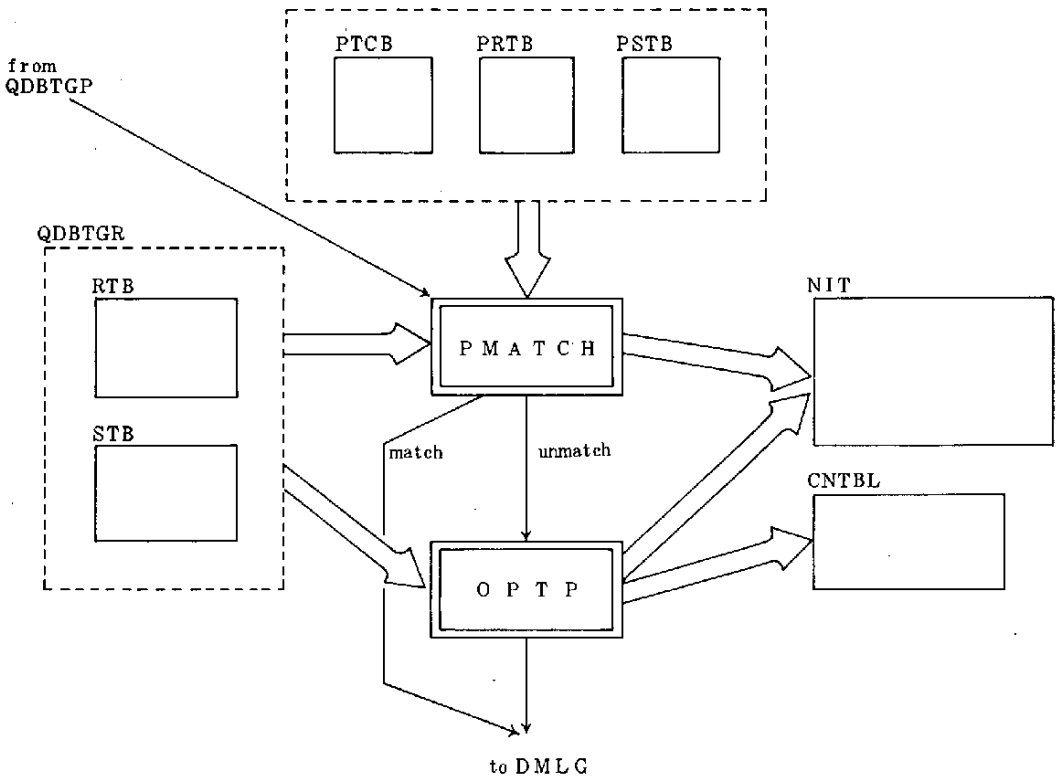


図８－６９　ＱＯＰＴＰ の 構 成

表８－１６　ＰＴＣＢ の 構 成

属 性 名	属性番号	ROLE	TYPE	LENGTH	説　　明
P＃	1	K	D	2	パターン番号
ＮＳＴＢ	2	N	D	2	このパターンがもつセット型の数
ＨＳＴＢ	3	N	D	2	ＰＳＴＢのhead pointer
ＴＳＴＢ	4	N	D	2	ＰＳＴＢのtail pointer
ＮＲＴＢ	5	N	D	2	このパターンのレコード型の数
ＨＲＴＢ	6	N	D	2	ＰＲＴＢのhead pointer
ＴＲＴＢ	7	N	D	2	ＰＲＴＢのtail pointer

表8－17　PRTBの構成

属性名	属性番号	ROLE	TYPE	LENGTH	説明
RTYPE	1	K	C	16	レコード型名
CRSTR	2	N	D	2	YES（=1）：CRSTRをもたねばならない。 NO（=0）：どちらでもよい。
RSTR	3	N	D	2	作業領域
R#	4	N	D	2	RTBへのポインタ

表8－18　PSTBの構成

属性名	属性番号	ROLE	TYPE	LENGTH	説明
STYPE	1	K	C	16	セット型名
S#	2	N	D	2	STBへのポインタ

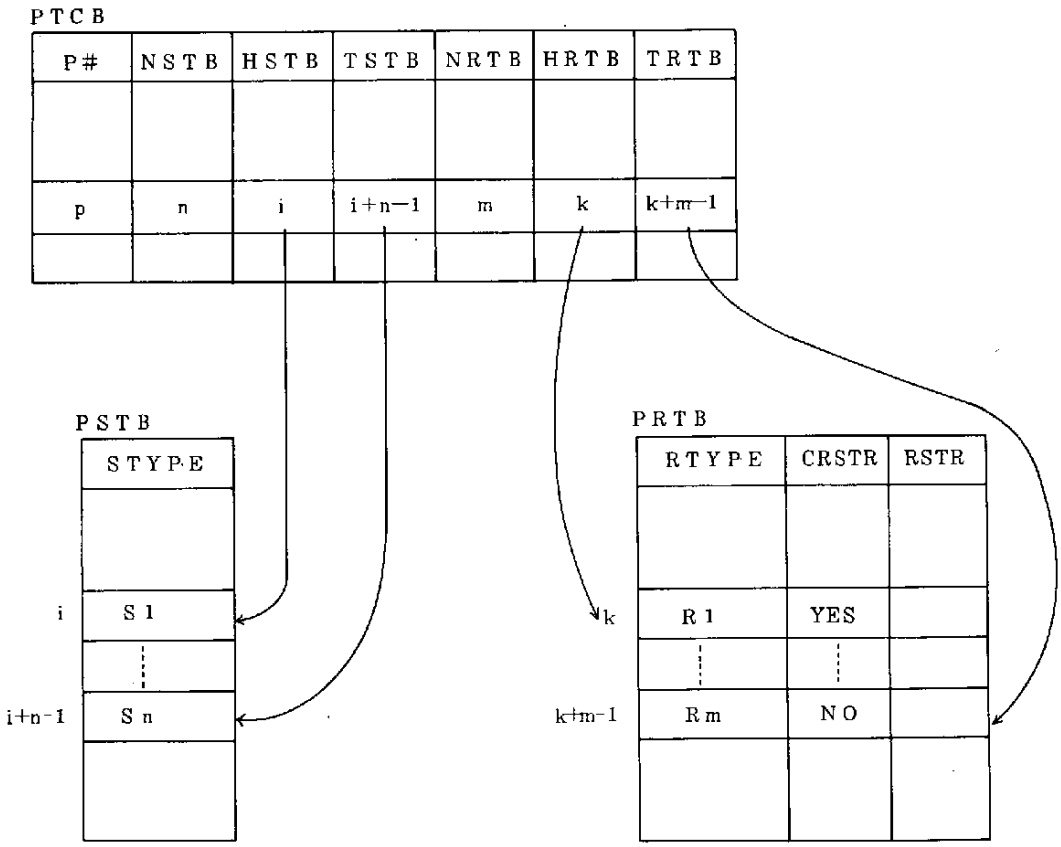


図8－70　PTCBとPSTB、PRTB

表8－19　PTCB の例

P#	NSTB	HSTB	TSTB	NRTB	HRTB	TRTB
1	2	1	2	3	1	3
2	1	3	3	2	4	5
3	1	4	4	2	6	7
4	2	5	6	3	8	10
5	4	7	10	5	11	15
6	4	11	14	5	16	20
7	4	15	18	5	21	25
8	2	19	20	3	26	28

表8－20　PRTB の例

	RTYPE	CRSTR	RSTR
1	KEYWORDS	1	
	PROJ－KEYW－LINK	0	
3	PROJECT	0	
4	PROJECT	1	
5	REPRESENTATIVE	0	
6	REPRESENTATIVE	1	
7	PROJECT	0	
8	SPONSOR	1	
	PROJ－SPON－LINK	0	
10	PROJECT	0	
11	SPONSOR	1	
	PROJ－SPON－LINK	0	
	PROJECT	0	
	PROJ－SPON－LINK	0	
15	KEYWORDS	0	
16	KEYWORDS	1	
	PROJ－KEYW－LINK	0	
	PROJECT	0	
	PROJ－SPON－LINK	0	

	R T Y P E	CRSTR	RSTR
20	SPONSOR	0	
21	KEYWORDS	1	
	PROJ−KEYW−LINK	0	
	SPONSOR	1	
	PROJ−SPON−LINK	0	
25	PROJECT	0	
26	PROJECT	1	
	PROJ−SPON−LINK	0	
28	SPONSOR	0	

表8−21 PSTB の例

	S T Y P E
1	KEYW − PROJ
2	PROJ − KEYW
3	REPR − PROJ
4	REPR − PROJ
5	SPON − PROJ
6	PROJ − SPON
7	SPON − PROJ
	PROJ − SPON
	PROJ − KEYW
10	KEYW − PROJ
11	KEYW − PROJ
	PROJ − KEYW
	PROJ − SPON
14	SPON − PROJ
15	KEYW − PROJ
	PROJ − KEYW
	SPON − PROJ
18	PROJ − SPON
19	PROJ − SPON
20	SPON − PROJ

B. PMATCH

ＰＴＣＢ（pattern control table）には、あるパターンに含まれるセット型の数とレコード型の数、及び、セット型が格納されているＰＳＴＢへのポインタ、レコード型が格納されているＰＲＴＢへのポインタがセットされている。パターンの照合は、まずＰＴＣＢをアクセスして、ＳＴＢの組（tuple）数とＮＳＴＢ（セット型の数）、及び、ＲＴＢのtuple 数とＮＲＴＢ（レコード型の数）の照合を行う。この照合が合うなら、ＰＳＴＢテーブルのＨＳＴＢ番目からＴＳＴＢ番目までをサーチし、ＳＴＢがアクセスパス内（ＰＳＴＢ）のセット型を全てもつかチェックする。次に、ＰＲＴＢテーブルのＨＲＴＢ番目からＴＲＴＢ番目までをサーチして、ＲＴＢがこの中の全てのレコード型をもつとき、このアクセス・パスが求めるものである。このとき、レコード型が必要な制限（restriction）をもつかも同時にチェックする。

求まったパターンに対応するＰＴＣＢのＰ＃がパターン番号である。また、ＰＲＴＢのレコード型のＨＲＴＢからＴＲＴＢのシーケンスは、アクセスパスを表わしている。ＰＳＴＢについても、ＨＳＴＢからＴＳＴＢまでのセット型のシーケンスはアクセスパスを表わしている。これらのレコード型とセット型を順にＮＩＴに格納し、上で求めたパターン番号とともに、次のプロセス（ＤＭＬＧ）へ渡す。

問合せを次のように出したとすると、

```
@T;
RANGE OF (K,P)(KEYWORDS,PROJECT);
RANGE OF (PKL)(PROJ-KEYW-LINK);
RANGE OF (PSL)(PROJ-SPON-LINK);
RANGE (S)(SPONSOR);
RETREIVE INTO R5 (K.KEYWORD,P.PROJNAME)
WHERE  K.KEYWORD  = PKL.KEYWORD  AND PKL.PROJ-NO  = P.PROJ-NO
  AND  P.PROJSTAT = "ACTIVE"     AND P.PROJ-NO    = PSL.PROJ-NO
  AND  PSL.SPONNAME=S.SPONNAME   AND S.SPONNAME   = "MIT1";
```

ＲＴＢとＳＴＢは、下のようになる。

SEQ	RECORD-TYPE	CARD	LINK	CRSTR	RSTR	CSLCT	RSLCT	RSLT	OCA	NPTR	USED	MARK
1	KEYWORDS	20	0	0	0	1.000	1.000	44	0	5	1	0
2	PROJECT	50	0	0	13	1.000	0.500	45	0	3	1	0
3	PROJ-KEYW-LINK	10	1	0	0	1.000	1.000	0	0	4	1	0
4	PROJ-SPON-LINK	6	1	0	0	1.000	1.000	0	0	2	1	0
5	SPONSOR	10	0	20	0	0.100	1.000	0	0	1	1	0

SEQ	SET-TYPE	CARD	OREC	MREC	MARK
1	SPON-PROJ	1	5	4	0
2	PROJ-SPON	6	2	4	0
3	PROJ-KEYW	10	2	3	0
4	KEYW-PROJ	1	1	3	0

従って、ＰＴＣＢとＰＳＴＢ、ＰＲＴＢの照合により、パターン番号は５、６、７のいづれかとなるが、上の例では制限がＳＰＯＮＳＯＲにあるので、パターン番号は５となる。また、生成されるＮＩＴは次の通りである。

```
SEQ   R#   S#  NTYPE   OCA  CPTR  SPTR  ERTRN  TRTRN  MARK
  1    5    0     0     0     0     0     0      0      0
0000000000000000000000000000000000  00000000000000000000000000000000
  2    4    1     1     0     0     0     0      0      0
0000000000000000000000000000000000  00000000000000000000000000000000
  3    2    2     0     0     0     0     0      0      0
0000000000000000000000000000000000  00000000000000000000000000000000
  4    3    3     1     0     0     0     0      0      0
0000000000000000000000000000000000  00000000000000000000000000000000
  5    1    4     0     0     0     0     0      0      0
0000000000000000000000000000000000  00000000000000000000000000000000
```

図８－７１　ＰＭＡＴＣＨの例

Ｃ．ＯＰＴＰ

ＰＭＡＴＣＨにおいて、ＱＤＢＴＧＲがマッチするアクセスパスが無かったときに、ＯＰＴＰは、非手続き的な問合せ表現ＱＤＢＴＧＲから適当なアクセスパスを生成する。ここでは、アクセスパスを決定するアルゴリズムとして、ＢＦアルゴリズム（breadth－first algorithm or BFA）〔5.4.4〕とＤＦアルゴリズム（depth－first algorithm or DFA）〔5.4.5〕と呼ぶ２つのアルゴリズムが組込まれている。この２つはコマンドにより選択可能となっている。

表８－２２　ＮＩＴの構成

属　性　名	属性番号	ROLE	TYPE	LENGTH	説　　明
Ｒ＃	1	Ｋ	Ｄ	2	ＲＴＢのtuple番号
Ｓ＃	2	Ｋ	Ｄ	2	ＳＴＢのtuple番号
ＮＴＹＰＥ	3	Ｎ	Ｄ	2	{ owner（＝０） next（＝１）
ＯＣＡ	4	Ｎ	Ｄ	4	ＲＴＢのアクセスされるオカーレンス数
ＣＰＴＲ	5	Ｎ	Ｄ	2	子ノードの先頭へのポインタ
ＳＰＴＲ	6	Ｎ	Ｄ	2	sisterノードのポインタ
ＣＢＩＴＪ〔ＢＦＡでのみ使用〕	7	Ｎ	Ｃ	4	右手のsisterの結果と、この結果とでjoinをとるべき合流ノードのbit－map
ＣＢＩＴＰ〔ＢＦＡでのみ使用〕	8	Ｎ	Ｃ	4	右手のsisterとのjoinの結果のうち、上位へ残されるべき合流ノードのbit－map
ＥＲＴＲＮ	9	Ｎ	Ｄ	2	条件を満足したオカーレンスがなかったときend－returnするノード番号

属　性　名	属性番号	ROLE	TYPE	LENGTH	説　　明
TRTRN	10	N	D	2	レコードが1件以上あったときにend－returnするノード番号
MARK	11	N	D	2	BFA{CN（2）：合流ノード / LINKM(=999)：リンク・レコード / NN（=1）：normalノード} DFA{CN（=2）：合流ノード / OLD(=5) / NEW(=6)}

表8－23　CNTBLの構成〔BFAでのみ使用〕

属　性　名	属性番号	ROLE	TYPE	LENGTH	説　　明
R#	1	K	D	2	合流ノードのtuple番号

a) BFアルゴリズム（BFA）

1) 〔開始ノード(entry node)の決定〕

entry node は、アクセスされるオカーレンス数が最小となるものを選ぶことにより決められる。

R_1、R_2、……、Rn を、RTB内の全レコード型とする。各々のRi（i=1、…、n）のアクセスされるオカーレンス数OCA(Ri)は次のようにして求められる。

$$OCA(Ri) = CARD(Ri) * CSLCT(Ri)$$

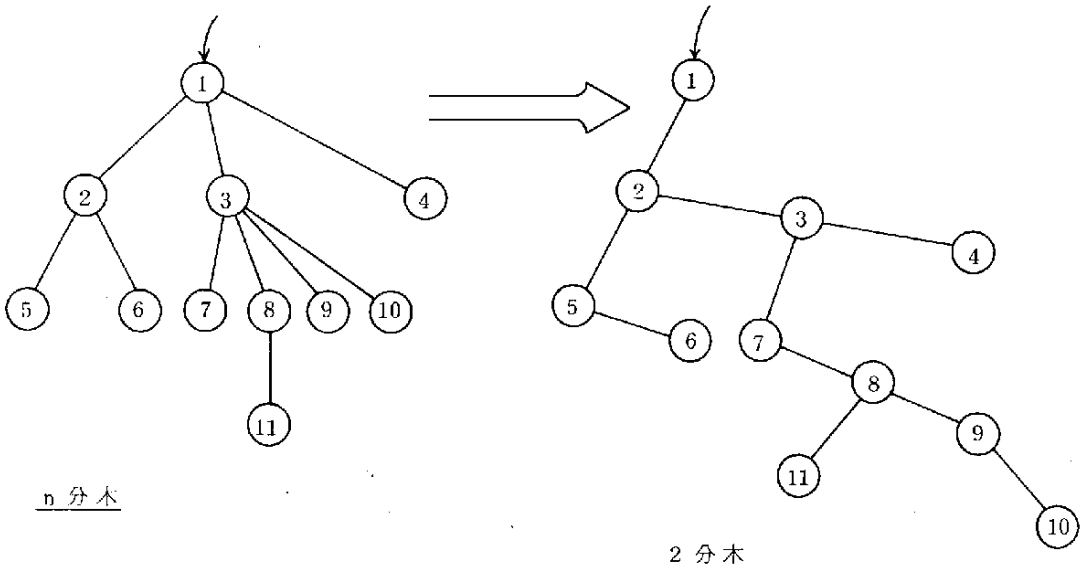
ここで、CARD(Ri) = Ri のカーディナリティ

$$CSLCT(Ri) = \begin{cases} 1 & (CRSTRをもたないとき) \\ CRSTRの選択度 & (CRSTRをもつとき) \end{cases}$$

各OCA(Ri)の値は、RTBのOCA属性値としてセットされる。

RTB内の全てのレコード型についてOCAが求まったら、OCAが最小のレコード型が開始ノード(entry node)となる。もし、同一のOCAを持つものがあれば、このノードに関する全てのセット型に対してこれがowner であるものをentry node とする。また、このentry node に関する情報をRTBからNITの先頭に格納する。

2) 以降の処理では、1) で決定されたentry node から、アクセスシーケンスがhierarchical order であるアクセス木(access tree) を生成する。アクセス木は、一般に、n分木（n≧2）であるが、これをNITの各組(tuple) をノードとする2分木として表わす。



entry node のＲＴＢでの組番号をｒとする。

3)〔ＱＤＢＴＧＲ内の隣接ノードを見つける〕

ＳＴＢをサーチして、ｒに隣接する子ノードをみつける。即ち、ＳＴＢのＯＲＥＣかＭＲＥＣがｒに等しいものをみつける。このセット型を介してｒとリンクされているレコード型ｒｒ（ＲＴＢの組番号）をみつける。

ｒｒがnormal レコード型（ＲＴＢ・ＬＩＮＫ（ｒｒ）＝０）のとき、 RTBのMARK（ｒｒ）に１を加える。その結果が１のときは処理4) へ行く。結果が２のときはＲＴＢ・ＮＰＴＲ（ｒｒ）にマークされているＮＩＴノードを合流ノード（confluent node)とマークし、ｒｒをＣＮＴＢＬに格納し、同じく処理4)へ進む。

また、ｒｒがリンクレコード型（ＲＴＢ・ＬＩＮＫ（ｒｒ）＝１）のとき、ＲＴＢのＭＡＲＫ（ｒｒ）がｌｉｎｋ－ｍａｒｋ（＝９９９）なら、処理３）へ戻り、ｒに隣接する次のＳＴＢをさがす。ＲＴＢのＭＡＲＫ（ｒｒ）がｌｉｎｋ－ｍａｒｋ以外なら、ｌｉｎｋ－ｍａｒｋをセットし処理4)へ。

4)〔ＮＩＴのｒとＲＴＢのｒｒをリンク〕

ＮＩＴにＲＴＢ（ｒｒ）の内容をセットし、ＲＴＢ・ＮＰＴＲ（ｒｒ）にそのＮＩＴノードへのポインタを格納する。ＳＴＢを使用済みとマークし、３）へ戻り、次のＳＴＢをさがす。

5)〔子ノードのソート〕

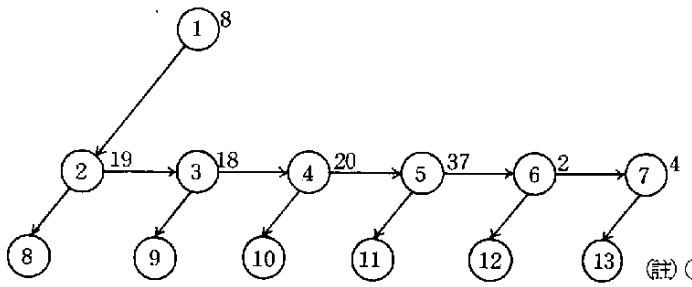
ｒに隣接する子ノードがＳＴＢにみつからなくなったら、子ノードをＮＩＴ・ＯＣＡをＫＥＹにして昇順にソートする。ソート後、子ノードを親ノードにリンクする。ＮＩＴ内の親ノードのポインタをｃｔ、子ノードはｓｔからｅｔまでに格納されているとすると、ＮＩＴ・ＣＰＴＲ（ｃｔ）にｓｔをセットし、ｓｔ＋１からｅｔまでは、ＳＰＴＲを用いてリンクする。即ち、図8－72

NIT

		R#	S#	NTYPE	OCA	CPTR	SPTR	
親ノード→	1	8	0	1	1	2	0	← ct
	2	19	2	1	4	8	3	← st
	3	18	3	1	4	9	4	
	4	20	20	1	4	10	5	
子ノード	5	37	17	1	5	11	6	
	6	2	9	1	6	12	7	
	7	4	11	1	10	13	0	← et
	8							
	⋮							

図8－72　ＮＩＴの例

では次のようにリンクされている。



(註)①はノード番号であり、円外の数字はＲ＃を表わす。

6) 次に、ct+1番目のノードを親ノード（ＮＩＴ・Ｒ＃（ct+1）＝ｒ）として、3）以降の処理を続ける。但し、そのノードが合流ノードであり、かつ、3回以上出現しているときは、次のノードを親ノードとする。また、合流ノードのときは、ＮＩＴのＣＢＩＴＰのＣＮＴＢＬに対応するビットをＯＮにする。

7) 6）でct＞etとなったとき、ＮＩＴの生成は終りとなる。このとき、各ノードのＣＢＩＴＰには、このノードが合流ノード（confluent node）である場合に対応するビットがセットされている。このＣＢＩＴＰを用いて、ＣＢＩＴＪをセットする。

あるノード（ｔ）のＣＢＩＴＪは、ｔの右手にあるノードt_1（＝ＳＰＴＲ（ｔ））を根ノードとする部分木が生成する結果（R_1）と、ｔを根ノードとする部分木が生成する結果（Ｒ）との結合（join）を、どの属性についてとるかを示している。一方、ノードｔのＣＢＩＴＰは、ｔを根ノードとする部分木の出力結果として含まれるべき合流ノードのキー属性を示している。

rを根ノード、r_1 、r_2をrの子ノードで、かつ、隣接するノードとすると、

$$CBITJ(r_1) \leftarrow CBITP(r_1) \wedge CBITP(r_2)$$

$$CBITP(r_1) \leftarrow CBITP(r_1) \vee CBITP(r_2)$$

として求められる。

b） DFアルゴリズム（DFA）

1）〔開始ノード(entry node)の決定〕

entry nodの決定（OCAの計算とOCAが最小のノードの選択）は、BCアルゴリズムの場合と同じである。entry node のRTBポインタをrとする。

2）〔RTBの全ノードをNEWとマーク〕

RTBのMARKを全てNEWとマークする。ここで、atはNITのカラント・ノード、rはRTB内のノードへのポインタ（tuple番号）、tはrに対応したNITノードの番号を、それぞれ、表わしているとする。

3）〔rに対応するNITノードの生成〕

entry node RTB（r）に対応するNITノードを生成する。

4）〔rをOLDとマーク〕

RTB（r）をOLDとマークする。即ち、RTB・MARK（r） がNEWであれば、OLDとマークし、RTB・NPTRにtをセットする。

5）〔rの隣接ノードをみつける〕

STBをサーチして、RTBノード（r）に隣接するノードを求める。1つも無ければ処理11）へ。

6）〔隣接ノードをNITに格納〕

隣接ノードを、NITのsat番目から順に格納する。 これらのノードは、NITのsat番目からtat番目に格納されたとする。

7）〔子ノードのソート〕

NITのsatからtatまでを、オカーレンス数（NIT・OCA）をKEYにして、昇順にソートする。

8）〔親ノードとリンク〕

NITのsatに格納されたノードをtにリンクする。

9）〔子ノードのMARKをセット〕

NITのsat番目のノードに対応するRTBのMARKがNEWであれば、それをOLDとマー

クする。２つ目以降のＮＥＷのものはリンクしないでスキップする。

ＯＬＤであれば、それは合流ノードとなったことを示しているので、ＲＴＢ、ＮＩＴノードとも合流ノードとマークする。ＲＴＢ・ＮＲＴＲのさすＮＩＴノードも合流ノードとマークする。また、ＮＩＴに対応するＳＴＢを使用済み（ＳＴＢ・ＭＡＲＫ＝ＯＮ）にする。ＲＴＢが、既に合流ノードなら、対応するＳＴＢ・ＭＡＲＫをＯＮにする。

10）〔 pushdown 〕

９）で求まった次の親ノードとすべきノードのＲＴＢ（ｒ）とＮＩＴ（ｔ）の対を、スタック領域にpushdown する。処理４）へ戻る。ここまでで、ＮＩＴは次のようになっている。

ＮＩＴ

	R#	S#	NTYPE	OCA	CPTR	SPTR	～	
1	8	0	1	1	2	0		
2	19	2	1	0	8			→ pushdown
3	18	3	1	0	0	0		
4	20	20	1	0	0	0		使用されない
5	37	17	1	0	0	0		
6	2	9	1	0	0	0		
7	4	11	1	0	0	0		
8	6	1	0	4	9			→ pushdown
9	14	7	1	200				→ pushdown
10	15	31	1	200	0	0		
11	42	29	1	250	0	0		使用されない

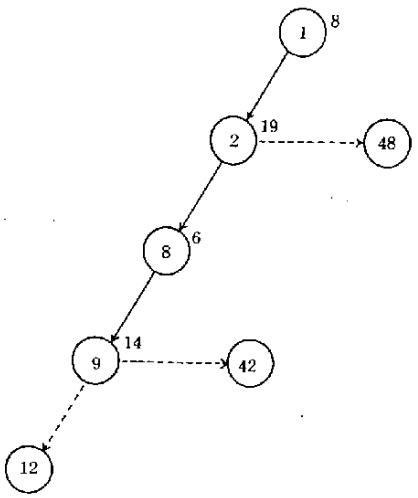


図８－７４　ＮＩＴと木

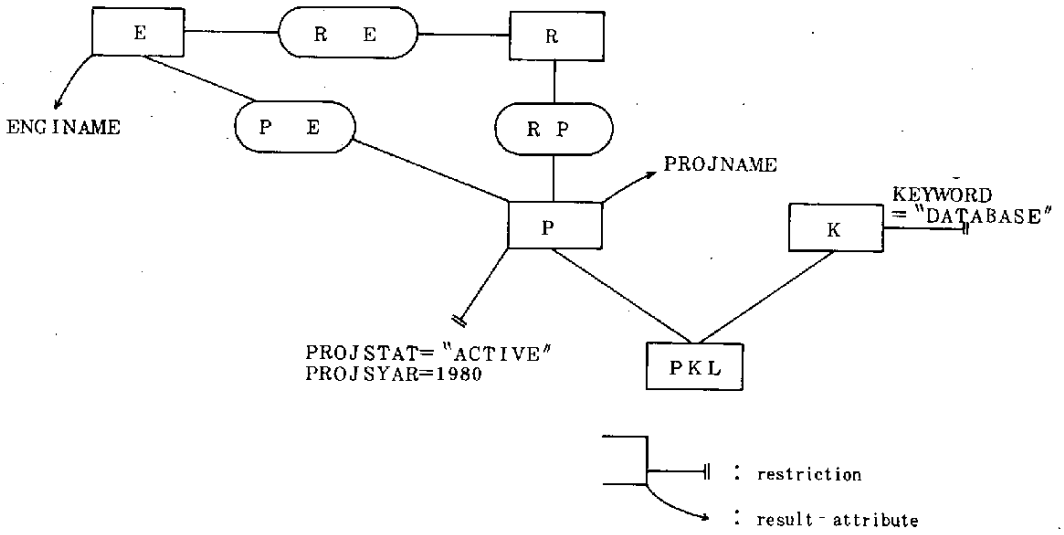
3番目から7番目のノードは使用されないまま残ることになる。

11) 〔 popup 〕

ＲＴＢ（ｒ）とＮＩＴ（ｔ）の対をpopupし、処理５）から実行する。スタック領域が空となったら処理を終る。

12) ＤＦＡでは、合流ノードについての中間結果の結合を行う必要がないので、ＣＢＩＴＰ、ＣＢＩＴＪフィールド、および、ＣＮＴＢＬ(confluent node table)は使用しない。

Ｃ） ＢＦアルゴリズムとＤＦアルゴリズムの例



問合せは次のようであったとする。

```
RANGE OF (P,R,E,K,PKL,RE,PE,RP) (PROJECT,REPRESENTATIVE,ENGINEER,
         KEYWORDS,PROJ-KEYW-LINK,REPR-ENGI,PROJ-ENGI,REPR-PROJ);
RETRIEVE INTO RESULT (P.PROJNAME, E.ENGINAME)
WHERE K.KEYWORD = "DATABASE"  AND P.PROJSTAT  = "ACTIVE"
  AND P.PROJSYAR= 1980
  AND K.KEYWORD = PKL.KEYWORD AND PKL.PROJ-NO = P.PROJ-NO
  AND P.PROJ-NO = RP.PROJ-NO  AND RP.REPR-NO  = R.REPR-NO
  AND R.REPR-NO = RE.REPR-NO  AND RE.ENGINAME = E.ENGINAME
  AND E.ENGINAME= PE.ENGINAME AND PE.PROJ-NO  = P.PROJ-NO;
```

この問合せから、ＱＤＢＴＧＲは次のように生成される。

ＲＴＢ

SEQ	RECORD-TYPE	CARD	LINK	CRSTR	RSTR	CSLCT	RSLCT	RSLT	OCA	NPTR	USED	MARK
1	PROJECT	50	0	0	74	1.000	0.050	75	50	3	1	2
2	REPRESENTATIVE	10	0	0	0	1.000	1.000	0	10	4	1	5
3	ENGINEER	20	0	0	0	1.000	1.000	76	20	6	1	5
4	KEYWORDS	20	0	8	0	0.200	1.000	0	4	1	1	5
5	PROJ-KEYW-LINK	10	1	0	0	1.000	1.000	0	10	2	1	5
6	REPR-ENGI	5	2	0	0	1.000	1.000	0	5	0	1	6
7	PROJ-ENGI	5	2	0	0	1.000	1.000	0	5	0	1	6
8	REPR-PROJ	5	2	0	0	1.000	1.000	0	5	0	1	6

STB

SEQ	SET-TYPE	CARD	OREC	MREC	MARK
1	PROJ-ENGI	5	1	3	1
2	REPR-ENGI	5	2	3	1
3	REPR-PROJ	5	2	1	1
4	PROJ-KEYW	10	1	5	1
5	KEYW-PROJ	1	4	5	1

BFアルゴリズムによると、NITは次の通りとなる。

NIT

	R#	S#	NTYPE	OCA	CPTR	SPTR	～
1	4	0	1	4	2	0	
2	5	5	1	4	3	0	
3	1	4	0	4	4	0	
4	2	3	0	0	6	5	
5	3	1	1	1	0	0	
6	3	2	1	1	0	0	

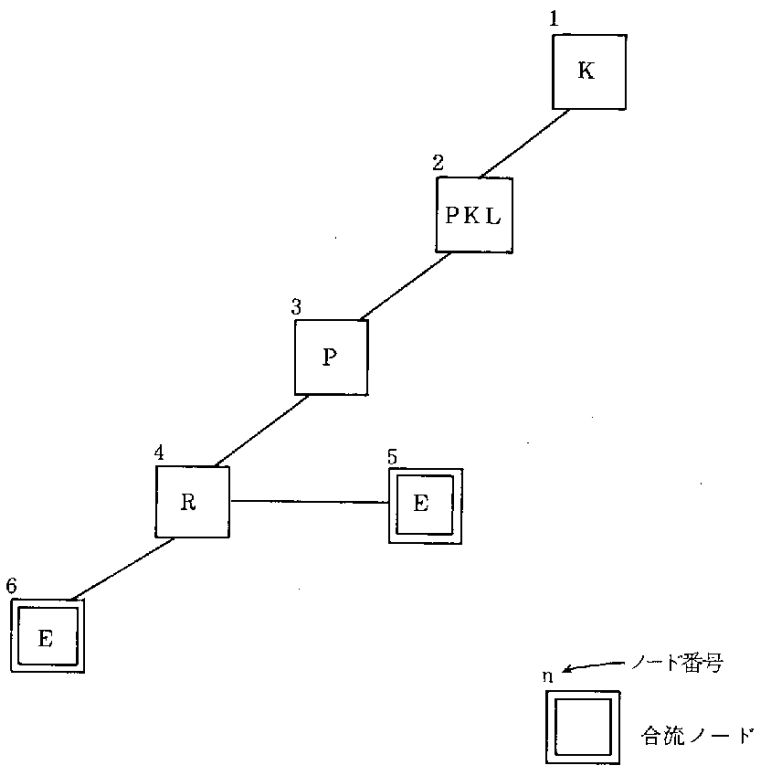


図8－75　BFAによるNITの生成と木構造表現

また、ＤＦアルゴリズムでは、ＮＩＴは次のように生成される。

ＮＩＴ

	R＃	S＃	NTYPE	ＯＣＡ	ＣＰＴＲ	ＳＰＴＲ	～
1	4	0	1	4	2	0	
2	5	5	1	1	3	0	
3	1	4	0	10	4	0	
4	2	3	0	3	6	0	
5	3	1	1	13	0	0	
6	3	2	1	50	7	0	
7	1	1	0	20	0	0	

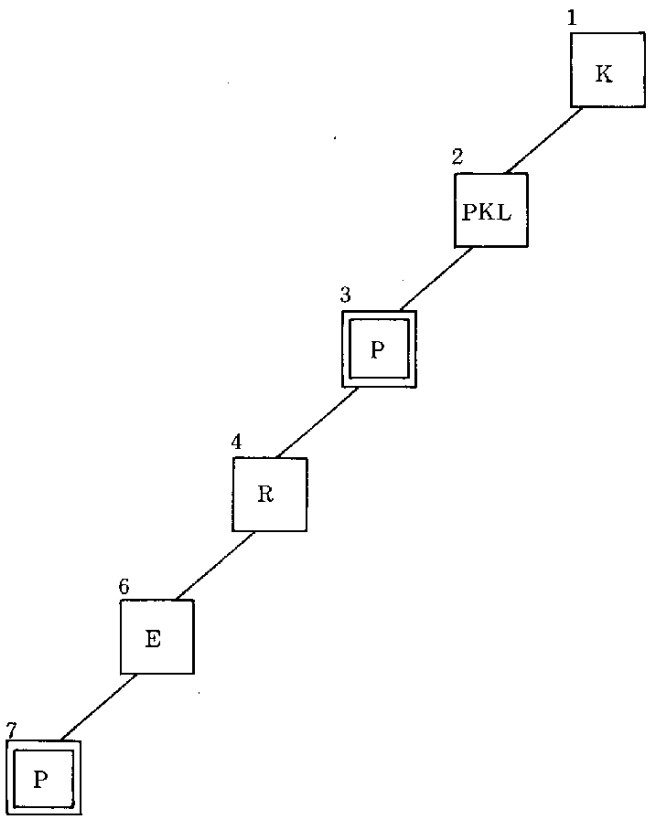


図８－７６　ＤＦＡによるＮＩＴの生成と木構造表現

8.2.6 ＤＭＬＧ

ＤＭＬＧは、ＱＯＰＴＰにより最適化されたアクセスパスに基づいて、目標ＤＢＭＳのＤＭＬシーケンスを生成するプロセスである。このとき、ＱＤＢＴＧＰにより作成されたＱＤＢＴＧＲ（ＲＴＢとＳＴＢ）、及び、アクセスパスを表現しているアクセス木（ＮＩＴ）を入力情報とする。一方、ＤＢＴＧＤＭＬライブラリには、ノードとアクセス形式の種類別の基本ＤＭＬシーケンス（Ａ～Ｊ）が格納されている〔表８－２４〕。ノード番号、レコード型名、ＣＡＬＣ項目名等の情報を指定することにより、アクセスパスに対応してＤＭＬシーケンスが生成される。

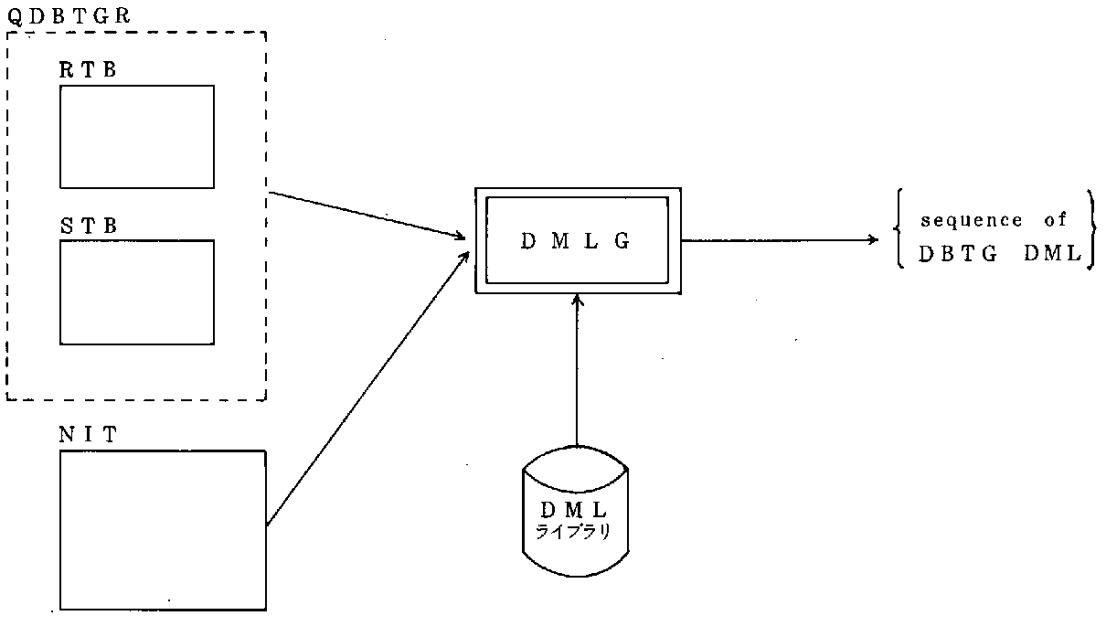


図８－７７　ＤＭＬＧ の 概 要

Ａ．ＤＭＬＧの構成

ＤＭＬＧは、ＤＭＬＧＰ、ＤＭＬＧＬ、ＤＭＬＧＴという３つのサブプロセスから構成されている。ＤＭＬＧＰは、ＱＯＰＴＰのＰＭＡＴＣＨでアクセスパターンが照合している場合に、パターン番号に従ってＤＭＬシーケンスを生成するプロセスである。また、ＤＭＬＧＬは、アクセスパスが線形の場合であり、ＤＭＬＧＴは、アクセスパスが木として表わされているときアクセス木をたどりながらＤＭＬシーケンスを生成する場合のプロセスである。

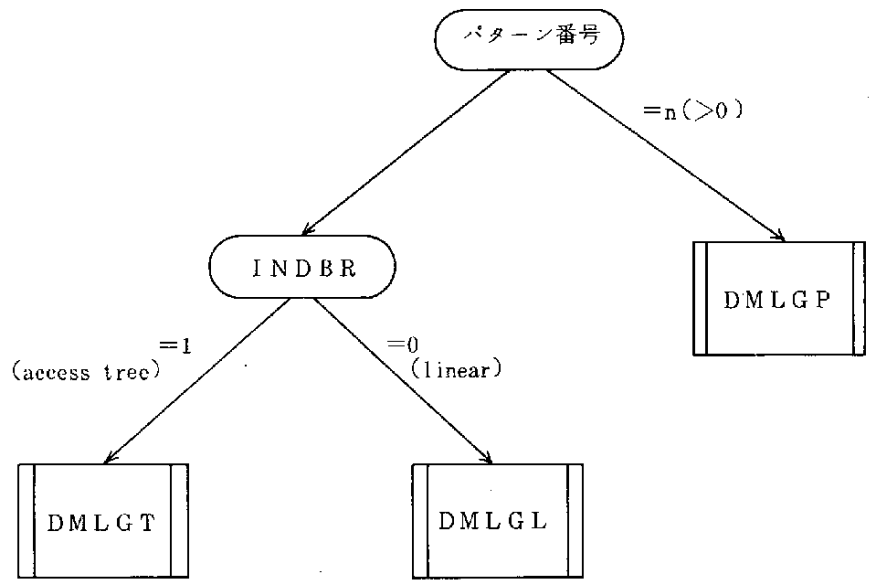


図8－78　DMLG の 構 成

B．DMLGL

DMLGLは、線形なアクセスパスからDMLシーケンスを生成するプロセスである。即ち、NITのCPTRをたどりながら、DMLシーケンスを生成してゆくプロセスである。これは、CBITJ、CBITPのビットが全てOFFであり、右手のポインタがNILであるような一種の木(tree)であると考えられる。従って、処理効率の点からスタックを用いていないことが異なるだけで、処理の内容についてはDMLGTと同じである。

C．DMLGT

DMLGTは、アクセス木(NIT)を、CPTRを左手のポインタ、SPTRを右手のポインタとして、行きがけ順(prcorder)になぞりながらDMLシーケンスを生成してゆくプロセスである。また、DMLシーケンスを生成する前の処理として、NITのERTRN(error－return pointer)とTRTRN(true－return pointer)のセットも行う。BFAの場合は、更に次のことを行う。根(root)に戻ったときに、そのNJTノードのCBITJ、又は、CBITPにONとなっているビットがあれば、論理和(join)をとるべき属性のリスト、又は、合流ノードのキー属性のリストを出力する。(もちろん、DFAではCBITP、CBITJは用いられない。)

1)〔ERTRNとTRTRNのセット〕

tをcurrent node、kをtの親のノードとする。図8－79は、tノードへのERTRNとTRTRNをセットするアルゴリズムを示している〔5.5.2を参照〕。

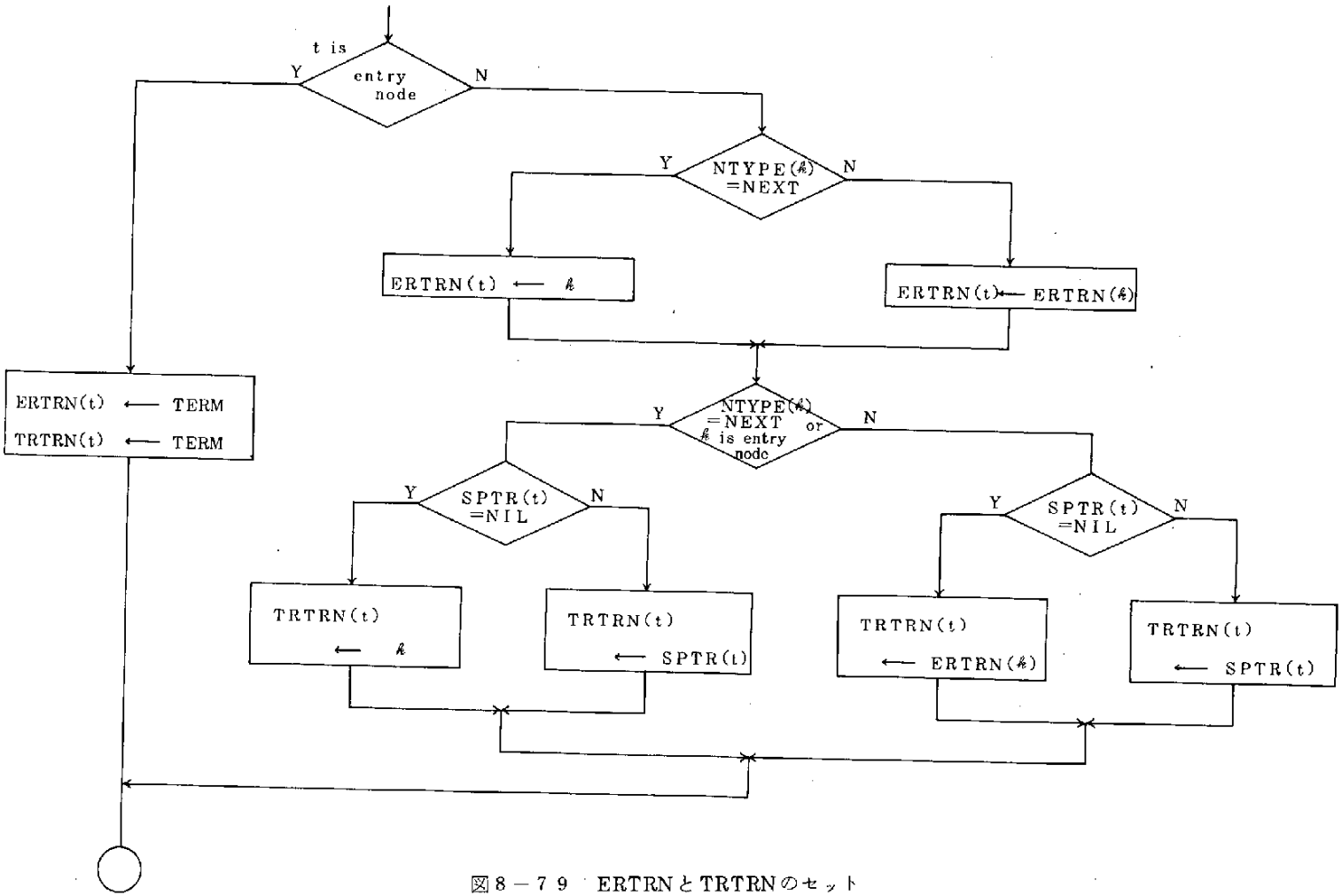


図8－79　ERTRNとTRTRNのセット

2) 〔アクセス・ユニット・パターンの判定〕

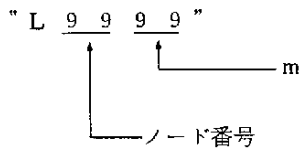
図８－８０のアルゴリズムによりユニットパターンの判定を行い、表８－２４のＤＭＬの基本型に基づいてＤＭＬシーケンスを出力する。

Ⓐ ～ Ⓙ は、表８－２４の形式に対応している。

3)〔ＤＭＬシーケンスの生成〕

アクセスユニット・パターン（Ａ～Ｊ）に対応するＤＭＬの基本型から、変換パターン・テーブル（表８－２５）に従って、￥ｎｍ、％ｍ等を置き換え、出力する。￥ｎ３、￥ｎ４、￥ｎ５については、変換対象のＣＲＳＴＲ、ＲＳＴＲ、ＲＳＬＴがＮＩＬのときは、その行を出力しない。

また、％ｍについては、そのつど、



というラベルを生成する。但し、ｍ＝１（％０１）の場合は、ＴＲＴＲＮやＥＲＴＲＮの飛び先として、他のノードから参照されるので、ノード番号と対にして管理する。

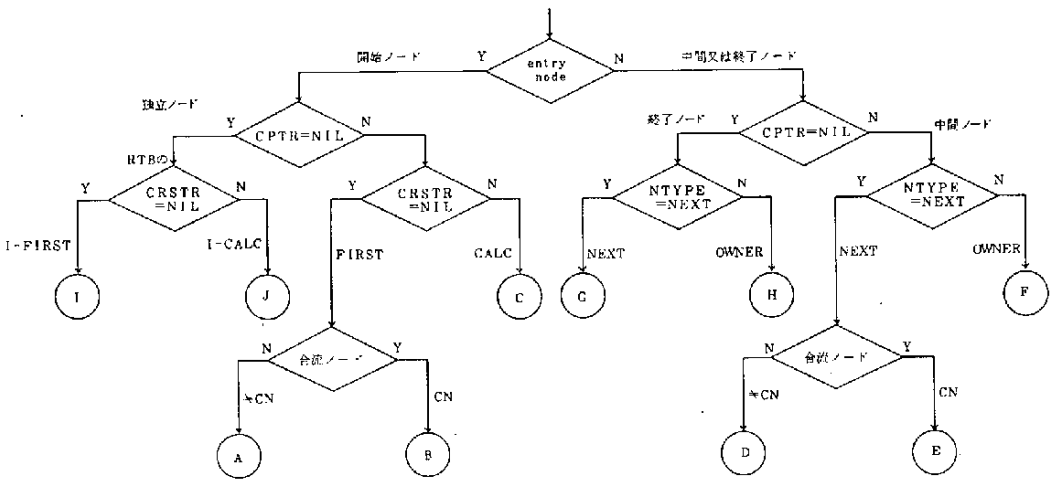


図８－８０　ユニットパターンの判定

（例）

NIT

	R#	S#	NTYPE	OCA	CPTR	SPTR	ERTRN	TRTRN
1	1	0	1	1	2	0	−1	−1
2	2	1	1	5	0	0	1	1

REPR-PROJ

RTB

RTYPE	～	CRSTR	RSTR	～	RSLT	OCA	～
REPRESENTATIVE						1	
PROJECT						1	

REPRNAME

PROJNAME

=

REPR-NO　　2000

=

PROJSTAT　　"ON"

この例では、ノード(1)は開始ノード（ＣＡＬＣ）＝C、ノード(2)は終了ノード（ＮＥＸＴ）＝G に従って変換される。

〔形式Ｃ〕

```
          MOVE FALSE TO LFOUND.
%01.  CALL  GET-NEXT-VALUE ((¥03),VAL,MODE).
          IF  MODE = 'END'  GO TO ¥06.
          MOVE VAL TO ¥08 IN ¥01.
          FIND ANY ¥01.
          IF  NOTFOUND = 'YES'  GO TO %01.
          GET ¥01.
          IF  NOT (¥04)  GO TO %01.
          CALL  RESULT (¥05).
*
```

〔形式Ｇ〕

```
%01.  IF  %10 = TRUE  GO TO %02.
          IF  ¥02 IS EMPTY  GO TO ¥06.
          MOVE FALSE TO %11.
          MOVE TRUE  TO %10.
%02.
          FIND NEXT ¥01 WITHIN ¥02.
          IF  ENDSET = 'YES'  GO TO %03.
          GET ¥01.
          IF  NOT (¥03 AND ¥04)  GO TO %01.
          CALL  RESULT (¥05).
          GO TO %02.
%03.
          MOVE %11 TO LFOUND.
          MOVE FALSE TO %10.
          IF  %11 = FALSE  GO TO ¥06.
                           GO TO ¥07.
```

最終的には、次のように出力される。

```
*---------- DBTG DML ----------

          MOVE FALSE TO LFOUND.
 L0101.   CALL  GET-NEXT-VALUE ((REPR-NO =  2000 ),VAL,MODE).
          IF  MODE = 'END'  GO TO TERM.
          MOVE VAL TO REPR-NO IN REPRESENTATIVE.
          FIND ANY REPRESENTATIVE.
          IF  NOTFOUND = 'YES'  GO TO L0101.
          GET REPRESENTATIVE.
          CALL  RESULT (REPRNAME ).
*
 L0201.   IF  L0210 = TRUE  GO TO L0202.
          IF  REPR-PROJ IS EMPTY  GO TO L0101.
          MOVE FALSE TO L0211.
          MOVE TRUE  TO L0210.
 L0202.
          FIND NEXT PROJECT WITHIN REPR-PROJ.
          IF  ENDSET = 'YES'  GO TO L0203.
          GET PROJECT.
          IF  NOT ( PROJSTAT = "ON" )  GO TO L0201.
          CALL  RESULT (PROJNAME ).
          GO TO L0202.
 L0203.
          MOVE L0211 TO LFOUND.
          MOVE FALSE TO L0210.
          IF  L0211 = FALSE  GO TO L0101.
                             GO TO L0101.
```

表8－24　DMLの基本型(1)

表8－24　DMLの基本型(1)

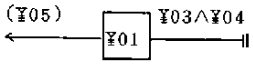
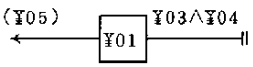
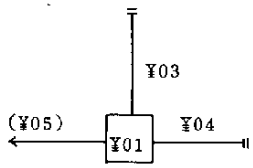
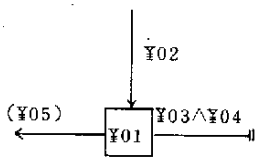
形式	アクセス木	DMLブロック
A 開始ノード (FIRST)		MOVE FALSE TO LFOUND. FIND FIRST ¥01. GO TO %02. %01. FIND NEXT ¥01. %02. IF ENDSET = 'YES' GO TO ¥06. GET ¥01. IF NOT (¥03 AND ¥04) GO TO %01. CALL RESULT (¥05). •
B 開始ノード (CN－ FIRST)		MOVE FALSE TO LFOUND. FIND FIRST ¥01. GO TO %02. %01. FIND ¥01 DB-KEY IS %20. FIND NEXT ¥01. %02. IF ENDSET = 'YES' GO TO ¥06. ACCEPT %20 FROM CURRENCY. GET ¥01. IF NOT (¥03 AND ¥04) GO TO %01. CALL RESULT (¥05). •
C 開始ノード (CALC)		MOVE FALSE TO LFOUND. %01. CALL GET-NEXT-VALUE ((¥03),VAL,MODE). IF MODE = 'END' GO TO ¥06. MOVE VAL TO ¥08 IN ¥01. FIND ANY ¥01. IF NOTFOUND = 'YES' GO TO %01. GET ¥01. IF NOT (¥04) GO TO %01. CALL RESULT (¥05). •
D 中間ノード (NEXT)		%01. IF %10 = TRUE GO TO %02. MOVE FALSE TO LFOUND. IF ¥02 IS EMPTY GO TO ¥06. MOVE TRUE TO %10. MOVE FALSE TO %11. GO TO %03. %02. IF %11 = TRUE OR LFOUND = TRUE MOVE TRUE TO %11 ELSE MOVE FALSE TO %11. %03. FIND NEXT ¥01 WITHIN ¥02. IF ENDSET NOT = 'YES' GO TO %04. MOVE %11 TO LFOUND. MOVE FALSE TO %10. IF %11 = FALSE GO TO ¥06. GO TO ¥07. %04. GET ¥01. IF NOT (¥03 AND ¥04) GO TO %03. CALL RESULT (¥05). •

表8－24　DMLの基本型(2)

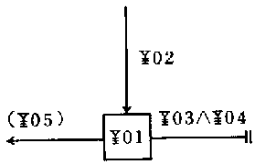
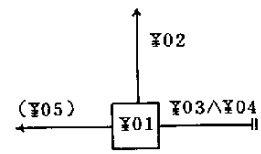
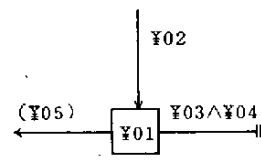
形式	アクセス木	DMLブロック
E 中間ノード (CN－ NEXT)		%01. IF %10 = TRUE GO TO %02. MOVE FALSE TO LFOUND. IF ¥02 IS EMPTY GO TO ¥06. MOVE TRUE TO %10. MOVE FALSE TO %11. GO TO %03. %02. IF %11 = TRUE OR LFOUND = TRUE MOVE TRUE TO %11 ELSE MOVE FALSE TO %11. FIND ¥01 DB-KEY IS %20. %03. FIND NEXT ¥01 WITHIN ¥02. IF ENDSET NOT = 'END' GO TO %04. MOVE %11 TO LFOUND. MOVE FALSE TO %10. IF %11 = FALSE GO TO ¥06. GO TO ¥07. %04. ACCEPT %20 FROM ¥02 CURRENCY. GET ¥01. IF NOT (¥03 AND ¥04) GO TO %03. CALL RESULT (¥05). *
F 中間ノード (OWNER)		%01. FIND OWNER WITHIN ¥02. IF NOTFOUND NOT = 'YES' GO TO %03. %02. MOVE FALSE TO LFOUND. GO TO ¥06. %03. GET ¥01. IF NOT (¥03 AND ¥04) GO TO %02. MOVE TRUE TO LFOUND. CALL RESULT (¥05). *
G 終了ノード (NEXT)		%01. IF %10 = TRUE GO TO %02. IF ¥02 IS EMPTY GO TO ¥06. MOVE FALSE TO %11. MOVE TRUE TO %10. %02. FIND NEXT ¥01 WITHIN ¥02. IF ENDSET = 'YES' GO TO %03. GET ¥01. IF NOT (¥03 AND ¥04) GO TO %01. CALL RESULT (¥05). GO TO %02. %03. MOVE %11 TO LFOUND. MOVE FALSE TO %10. IF %11 = FALSE GO TO ¥06. GO TO ¥07.

表８－２４　ＤＭＬの基本型(3)

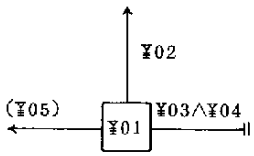
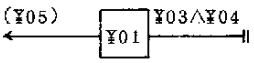
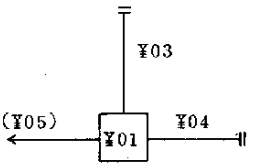
形式	アクセス木	ＤＭＬブロック
H 終了ノード (OWNER)		%01. FIND OWNER WITHIN #02. IF NOTFOUND = 'NO' GO TO %03. %02. MOVE FALSE TO LFOUND. GO TO #06. %03. GET #01. IF NOT (#03 AND #04) GO TO %02. MOVE TRUE TO LFOUND. CALL RESULT (#05). GO TO #07.
I 独立ノード (I－ FIRST)		FIND FIRST #01. GO TO %02. %01. FIND NEXT #01. %02. IF ENDSET = 'YES' GO TO #06. GET #01. IF NOT (#03 AND #04) GO TO %01. CALL RESULT (#05). GO TO %01.
J 独立ノード (I－ CALC)		%01. CALL GET-NEXT-VALUE ((#03),VAL,MODE). IF MODE = 'END' GO TO #06. MOVE VAL TO #08 IN #01. FIND ANY #01. IF NOTFOUND = 'YES' GO TO %01. GET #01. IF NOT (#04) GO TO %01. CALL RESULT (#05). GO TO %01.

表８－２５　変換パターン・テーブル

パターン	rule #	規　　則
¥ｎ１	1	これを、レコード型で置き換える。
¥ｎ２	2	これを、セット型で置き換える。
¥ｎ３	3	これを、CRSTRをinfix notationに直したものに置き換える。
¥ｎ４	4	これを、RSTRをinfix notationに直したものに置き換える。
¥ｎ５	5	これを、ＲＳＬＴ（result attribute）で置き換える。
¥ｎ６	6	これを、error－return nodeに対応するラベルで置き換える。
¥ｎ７	7	これを、true－return nodeに対応するラベルで置き換える。
¥ｎ８	8	これを、ＣＡＬＣ項目名に置き換える。
%ｍ	9	これを、ラベル名に置き換える。

ここで、０≦ｎ≦９であり、

ｎ≧１のとき、ＮＩＴ内のノード番号ｎのノードを用いる。

ｎ＝０のとき、現在のノードを用いる。

D．DMLGP

ＤＭＬＧＰは、パターン番号に基づいてＤＭＬシークンスを生成するプロセスである。変換パターン・テーブルによる変換は、ＤＭＬＧＴと同じである。アクセスのシーケンスは、ＮＩＴの先頭から順に格納されている。

このプロセスでは、アクセスするデータベース毎に異なったアクセスパスライブラリを使用することになる。次節で、今回使用したプロジェクトデータベース（ＰＲＤＢ）とアクセスパスライブラリについて述べる。

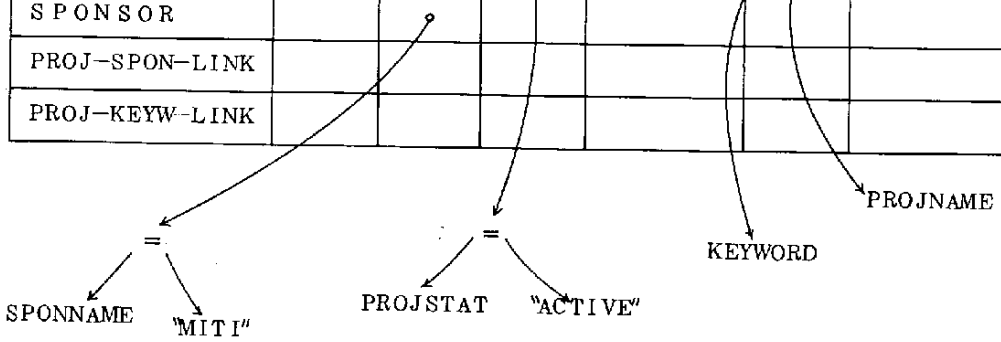
（例） パターン番号＝５〔表８－１９を参照〕の例を示す。

NIT

R#	S#	NTYPE	～
3	0	0	
4	1	1	
2	2	0	
5	3	1	
1	4	0	

RTB

RTYPE	～	CRSTR	RSTR	～	RSLT	～
KEYWORDS						
PROJECT						
SPONSOR						
PROJ-SPON-LINK						
PROJ-KEYW-LINK						



STB

STYPE	OCA	OREC	MREC
SPON-PROJ	1	3	4
PROJ-SPON	6	2	4
PROJ-KEYW	10	2	5
KEYW-PROJ	1	1	5

PMATCHプロセスにより、上のようにセットされているとする。

アクセスパス　ライブラリ

```
          MOVE ZERO TO I.
L1.       IF (I + 1) > J,  GO TO TERM.
          ADD 1 TO I.
          MOVE D (I) TO SPONNAME IN SPONSOR.
*
L12.      FIND DUPLICATE SPONSOR.
          IF NOTFOUND = 'YES',  GO TO L14.
*
L13.      IF D (I) IS ANY,  GO TO L1.
          GO TO L12.
*
L14.      GET SPONSOR.
          IF  NOT (¥14),  GO TO L13.
          MOVE  (¥15)  TO  RR.
*
L2.       FIND NEXT PROJ-SPON-LINK WITHIN SPON-PROJ.
          IF ENDSET = 'YES',  GO TO L13.
          IF OWNER WITHIN PROJ-SPON IS NULL,  GO TO L2.
          GET PROJ-SPON-LINK.
          IF  NOT (¥23 AND ¥24),  GO TO L2.
          MOVE  (¥25)  TO  RR.
*
L3.       FIND OWNER WITHIN PROJ-SPON.
          IF PROJ-KEYW IS EMPTY,  GO TO L2.
          GET PROJECT.
          IF  NOT (¥33 AND ¥34),  GO TO L2.
          MOVE  (¥35)  TO  RR.
*
L4.       FIND NEXT PROJ-KEYW-LINK WITHIN PROJ-KEYW.
          IF ENDSET = 'YES',  GO TO L2.
          IF OWNER WITHIN KEYW-PROJ IS NULL,  GO TO L4.
          GET PROJ-KEYW-LINK.
          IF  NOT (¥43 AND ¥44),  GO TO L4.
          MOVE  (¥45)  TO  RR.
*
L5.       FIND OWNER WITHIN KEYW-PROJ.
          GET KEYWORDS.
          IF  NOT (¥53 AND ¥54),  GO TO L4.
          MOVE  (¥55)  TO  RR.
          WRITE RR.
          GO TO L4.
```

DBTG　DML　シーケンス

```
*---------- DBTG DML ----------
          MOVE ZERO TO I.
L1.       IF (I + 1) > J,  GO TO TERM.
          ADD 1 TO I.
          MOVE D (I) TO SPONNAME IN SPONSOR.
*
L12.      FIND DUPLICATE SPONSOR.
          IF NOTFOUND = 'YES',  GO TO L14.
*
L13.      IF D (I) IS ANY,  GO TO L1.
          GO TO L12.
*
L14.      GET SPONSOR.
*
L2.       FIND NEXT PROJ-SPON-LINK WITHIN SPON-PROJ.
          IF ENDSET = 'YES',  GO TO L13.
          IF OWNER WITHIN PROJ-SPON IS NULL,  GO TO L2.
          GET PROJ-SPON-LINK.
*
L3.       FIND OWNER WITHIN PROJ-SPON.
          IF PROJ-KEYW IS EMPTY,  GO TO L2.
          GET PROJECT.
          IF  NOT ( PROJSTAT = 'ACTIVE' ),  GO TO L2.
          MOVE  (PROJNAME )  TO  RR.
*
L4.       FIND NEXT PROJ-KEYW-LINK WITHIN PROJ-KEYW.
          IF ENDSET = 'YES',  GO TO L2.
          IF OWNER WITHIN KEYW-PROJ IS NULL,  GO TO L4.
          GET PROJ-KEYW-LINK.
*
L5.       FIND OWNER WITHIN KEYW-PROJ.
          GET KEYWORDS.
          MOVE  (KEYWORD )  TO  RR.
          WRITE RR.
          GO TO L4.
```

図8－81　アクセス　ライブラリから
DBTG　DML　シーケンスへの変換

E．アクセスパス　ライブラリ

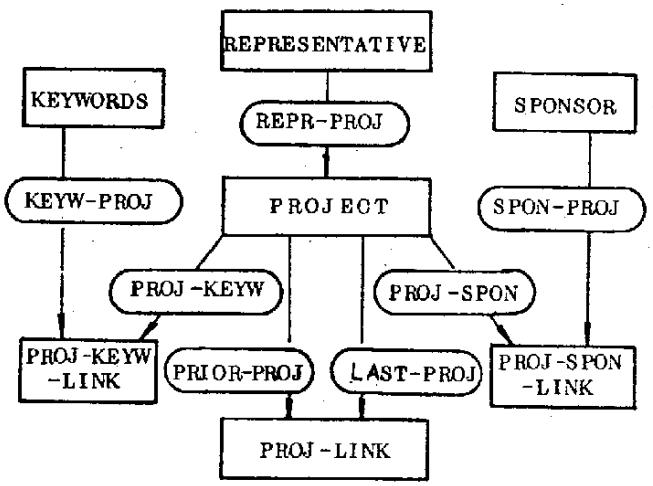


図８－８２　ＰＲＤＢの構造図

ＰＭＡＴＣＨプロセスで照合の対象となるデータベースと、そのデータベースに対する代表的な問合せの例について述べる。

このデータベースはプロジェクトデータベース（ＰＲＤＢ）と呼ばれ、ＤＢＴＧ型のＤＢＭＳにより管理される〔8.1.3例題の図８－２９を参照〕。ここでは、アクセスパスライブラリに関係のある部分（即ち、ＥＮＧＩＮＥＥＲレコードを除いた部分）のデータ構造図を、図８－８２に示す。また、ＰＲＤＢに対する代表的な問合せの、パターン番号毎のＤＭＬシーケンスを表８－２６に示す。

ＰＲＤＢに対する代表的な問合せの例を示す。

① ＫＥＹＷＯＲＤＳ → ＰＲＯＪＥＣＴ
② ＰＲＯＪＥＣＴ → ＲＥＰＲＥＳＥＮＴＡＴＩＶＥ
③ ＲＥＰＲＥＳＥＮＴＡＴＩＶＥ → ＰＲＯＪＥＣＴ
④ ＳＰＯＮＳＯＲ → ＰＲＯＪＥＣＴ
⑤ ＳＰＯＮＳＯＲ → ＰＲＯＪＥＣＴ → ＫＥＹＷＯＲＤＳ
⑥ ＫＥＹＷＯＲＤＳ → ＰＲＯＪＥＣＴ → ＳＰＯＮＳＯＲ
⑦ ＳＰＯＮＳＯＲ → ＰＲＯＪＥＣＴ
　 ＫＥＹＷＯＲＤＳ ↗
⑧ ＰＲＯＪＥＣＴ → ＳＰＯＮＳＯＲ

これを、ＰＲＢＬのＬＣＳレベルの図で示すと次のようになる。

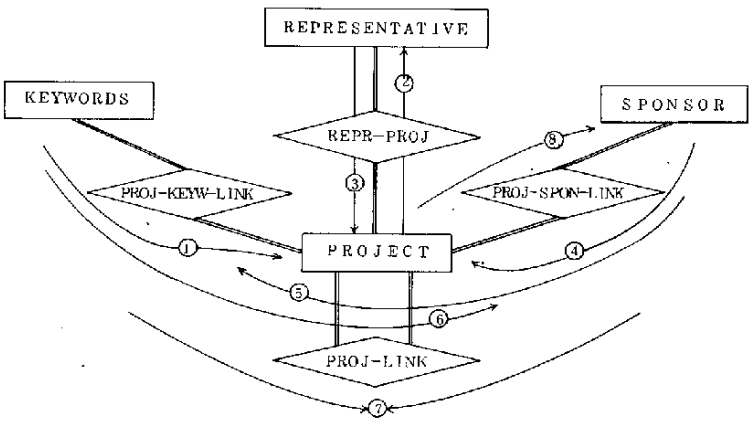


図８－８３　ＰＲＢＬの代表的な問合せ

表8－26　パターン番号によるアクセスパス(1)

パターン番号	DML シーケンス
1	(KEYWORDS → PROJECT) MOVE ZERO TO I. L1. IF (I + 1) > J, GO TO TERM. ADD 1 TO I. MOVE D (I) TO KEYWORD IN KEYWORDS. * L12. FIND DUPLICATE KEYWORDS. IF NOTFOUND = 'YES', GO TO L1. IF KEYW-PROJ NOT EMPTY, GO TO L14. * L13. IF D (I) IS ANY, GO TO L1. GO TO L12. * L14. GET KEYWORDS. IF NOT (¥14), GO TO L13. MOVE (¥15) TO RR. * L2. FIND NEXT PROJ-KEYW-LINK WITHIN KEYW-PROJ. IF ENDSET = 'YES', GO TO L13. IF OWNER WITHIN PROJ-KEYW IS NULL, GO TO L2. GET PROJ-KEYW-LINK. IF NOT (¥23 AND ¥24), GO TO L2. MOVE (¥25) TO RR. * L3. FIND OWNER WITHIN PROJ-KEYW. GET PROJECT. IF NOT (¥33 AND ¥34), GO TO L2. MOVE (¥35) TO RR. WRITE RR. GO TO L13.
2	(PROJECT → REPRESENTATIVE) MOVE ZERO TO I. L1. IF (I + 1) > J, GO TO TERM. ADD 1 TO I. MOVE D (I) TO PROJ-NO IN PROJECT. * L12. FIND DUPLICATE PROJECT. IF NOTFOUND = 'YES', GO TO L1. IF OWNER WITHIN REPR-PROJ IS NOT NULL, GO TO L14. * L13. IF D (I) IS ANY, GO TO L1. GO TO L12. * L14. GET PROJECT. IF NOT (¥14), GO TO L13. MOVE (¥15) TO RR. * FIND OWNER WITHIN REPR-PROJ. GET REPRESENTATIVE. IF NOT (¥23 AND ¥24), GO TO L13. MOVE (¥25) TO RR. WRITE RR. GO TO L13.

表８－２６　パターン番号によるアクセスパス(2)

パターン番号	ＤＭＬ　シ　ー　ケ　ン　ス
3	（REPRESENTATIVE → PROJECT） MOVE ZERO TO I. L1. IF (I + 1) > J, GO TO TERM. ADD 1 TO I. MOVE D (I) TO REPR-NO IN REPRESENTATIVE. * L12. FIND DUPLICATE REPRESENTATIVE. IF NOTFOUND = 'YES', GO TO L1. IF REPR-PROJ NOT EMPTY, GO TO L14. * L13. IF D (I) IS ANY, GO TO L1. GO TO L12. * L14. GET REPRESENTATIVE. IF NOT (¥14), GO TO L13. MOVE (¥15) TO RR. * L2. FIND NEXT PROJECT WITHIN REPR-PROJ. IF ENDSET = 'YES', GO TO L13. GET PROJECT. IF NOT (¥23 AND ¥24), GO TO L2. MOVE (¥25) TO RR. WRITE RR. GO TO L2.
4	（SPONSOR → PROJECT） MOVE ZERO TO I. L1. IF (I + 1) > J, GO TO TERM. ADD 1 TO I. MOVE D (I) TO SPONNAME IN SPONSOR. * L12. FIND DUPLICATE SPONSOR. IF NOTFOUND = 'YES', GO TO L1. IF SPON-PROJ NOT EMPTY, GO TO L14. * L13. IF D (I) IS ANY, GO TO L1. GO TO L12. * L14. GET SPONSOR. IF NOT (¥14), GO TO L13. MOVE (¥15) TO RR. * L2. FIND NEXT PROJ-SPON-LINK WITHIN SPON-PROJ. IF ENDSET = 'YES', GO TO L13. IF OWNER WITHIN PROJ-SPON IS NULL, GO TO L2. GET PROJ-SPON-LINK. IF NOT (¥23 AND ¥24), GO TO L2. MOVE (¥25) TO RR. * L3. FIND OWNER WITHIN PROJ-SPON. GET PROJECT. IF NOT (¥33 AND ¥34), GO TO L2. MOVE (¥35) TO RR. WRITE RR. GO TO L13.

表8－26　パターン番号によるアクセスパス(3)

パターン番号	DML シーケンス
5	(SPONSOR → PROJECT → KEYWORDS) MOVE ZERO TO I. L1. IF (I + 1) > J, GO TO TERM. ADD 1 TO I. MOVE D (I) TO SPONNAME IN SPONSOR. * L12. FIND DUPLICATE SPONSOR. IF NOTFOUND = 'YES', GO TO L14. * L13. IF D (I) IS ANY, GO TO L1. GO TO L12. * L14. GET SPONSOR. IF NOT (¥14), GO TO L13. MOVE (¥15) TO RR. * L2. FIND NEXT PROJ-SPON-LINK WITHIN SPON-PROJ. IF ENDSET = 'YES', GO TO L13. IF OWNER WITHIN PROJ-SPON IS NULL, GO TO L2. GET PROJ-SPON-LINK. IF NOT (¥23 AND ¥24), GO TO L2. MOVE (¥25) TO RR. * L3. FIND OWNER WITHIN PROJ-SPON. IF PROJ-KEYW IS EMPTY, GO TO L2. GET PROJECT. IF NOT (¥33 AND ¥34), GO TO L2. MOVE (¥35) TO RR. * L4. FIND NEXT PROJ-KEYW-LINK WITHIN PROJ-KEYW. IF ENDSET = 'YES', GO TO L2. IF OWNER WITHIN KEYW-PROJ IS NULL, GO TO L4. GET PROJ-KEYW-LINK. IF NOT (¥43 AND ¥44), GO TO L4. MOVE (¥45) TO RR. * L5. FIND OWNER WITHIN KEYW-PROJ. GET KEYWORDS. IF NOT (¥53 AND ¥54), GO TO L4. MOVE (¥55) TO RR. WRITE RR. GO TO L4.

表８－２６　パターン番号によるアクセスパス(4)

パターン番号	ＤＭＬ　シ　ー　ケ　ン　ス
6	（KEYWORDS → PROJECT → SPONSOR）

```
          MOVE ZERO TO I.
L1.       IF (I + 1) > J,  GO TO TERM.
          ADD 1 TO I.
          MOVE D (I) TO KEYWORD IN KEYWORDS.
*
L12.      FIND DUPLICATE KEYWORDS.
          IF NOTFOUND = 'YES',  GO TO L1.
          IF KEYW-PROJ NOT EMPTY,  GO TO L14.
*
L13.      IF D (I) IS ANY,  GO TO L1.
          GO TO L12.
*
L14.      GET KEYWORDS.
          IF  NOT (¥14),  GO TO L13.
          MOVE  (¥15)  TO  RR.
*
L2.       FIND NEXT PROJ-KEYW-LINK WITHIN KEYW-PROJ.
          IF ENDSET = 'YES',  GO TO L13.
          IF OWNER WITHIN PROJ-KEYW IS NULL,  GO TO L2.
          GET PROJ-KEYW-LINK.
          IF  NOT (¥23 AND ¥24),  GO TO L2.
          MOVE  (¥25)  TO  RR.
*
L3.       FIND OWNER WITHIN PROJ-KEYW.
          IF PROJ-SPON EMPTY,  GO TO L2.
          GET PROJECT.
          IF  NOT (¥33 AND ¥34),  GO TO L2.
          MOVE  (¥35)  TO  RR.
*
L4.       FIND NEXT PROJ-SPON-LINK WITHIN PROJ-SPON.
          IF ENDSET = 'YES',  GO TO L2.
          IF OWNER WITHIN SPON-PROJ IS NULL,  GO TO L4.
          GET PROJ-SPON-LINK.
          IF  NOT (¥43 AND ¥44),  GO TO L4.
          MOVE  (¥45)  TO  RR.
*
L5.       FIND OWNER WITHIN SPON-PROJ.
          GET SPONSOR.
          IF  NOT (¥53 AND ¥54),  GO TO L4.
          MOVE  (¥55)  TO  RR.
          WRITE RR.
          GO TO L4.
```

表8－26　パターン番号によるアクセスパス(5)

パターン番号	DML シーケンス
7	(SPONSOR, KEYWORDS → PROJECT) MOVE ZERO TO I. L1. IF (I + 1) > J, GO TO L3. ADD 1 TO I. MOVE D (I) TO KEYWORD IN KEYWORDS. * L12. FIND DUPLICATE KEYWORDS. IF NOTFOUND = 'YES', GO TO L1. IF KEYW-PROJ NOT EMPTY, GO TO L14. * L13. IF D (I) IS ANY, GO TO L1. GO TO L12. * L14. GET KEYWORDS. IF NOT (¥14), GO TO L13. MOVE (¥15) TO RR-1. * L2. FIND NEXT PROJ-KEYW-LINK WITHIN KEYW-PROJ. IF ENDSET = 'YES', GO TO L13. GET PROJ-KEYW-LINK. IF NOT (¥23 AND ¥24), GO TO L2. MOVE PROJ-NO OF PROJ-KEYW-LINK TO RR-1. WRITE RR-1. GO TO L2. * L3. MOVE ZERO TO I. L32. IF (I + 1) > K, GO TO JOIN. ADD 1 TO I. MOVE D (I) TO SPONNAME TO SPONSOR. * L33. FIND DUPLICATE SPONSOR. IF NOTFOUND = 'YES', GO TO L32. IF SPON-PROJ NOT EMPTY, GO TO L35. * L34. IF D (I) IS ANY, GO TO L32. GO TO L33. * L35. GET SPONSOR. IF NOT (¥33 AND ¥34), GO TO L34. MOVE (¥35) TO RR-2. * L4. FIND NEXT PROJ-SPON-LINK WITHIN SPON-PROJ. IF ENDSET = 'YES', GO TO L34. GET PROJ-SPON-LINK. IF NOT (¥43 AND ¥44), GO TO L4. MOVE PROJ-NO OF PROJ-SPON-LINK TO RR-2. WRITE RR-2. GO TO L4. * JOIN. GET UNIQUE D FROM RR-1 AND RR-2. * MOVE ZERO TO I. L5. IF (I + 1) > L, GO TO TERM. ADD 1 TO I. MOVE D (I) TO PROJ-NO. * L52. FIND DUPLICATE PROJECT. IF NOTFOUND = 'YES', GO TO L5. GO TO L54.

表８－２６　パターン番号によるアクセスパス(6)

パターン番号	ＤＭＬ　シ　ー　ケ　ン　ス
7	L53.　IF D (I) IS ANY, GO TO L5. 　　GO TO L52. * L54.　GET PROJECT. 　　WRITE PROJNAME. 　　GO TO L53.
8	(PROJECT → SPONSOR) 　　MOVE ZERO TO I. L1.　IF (I + 1) > J, GO TO TERM. 　　ADD 1 TO I. 　　MOVE D (I) TO PROJ-NO IN PROJECT. * L12.　FIND DUPLICATE PROJECT. 　　IF NOTFOUND = 'YES', GO TO L1. 　　IF PROJ-SPON NOT EMPTY, GO TO L14. * L13.　IF D (I) IS ANY, GO TO L1. 　　GO TO L12. * L14.　GET PROJECT. 　　IF NOT (¥14), GO TO L13. 　　MOVE (¥15) TO RR. * L2.　FIND NEXT PROJ-SPON-LINK WITHIN PROJ-SPON. 　　IF ENDSET = 'YES', GO TO L13. 　　IF OWNER WITHIN SPON-PROJ IS NULL, GO TO L2. 　　GET PROJ-SPON-LINK. 　　IF NOT (¥23 AND ¥24), GO TO L2. 　　MOVE (¥25) TO RR. * L3.　FIND OWNER WITHIN SPON-PROJ. 　　GET SPONSOR. 　　IF NOT (¥33 AND ¥34), GO TO L2. 　　MOVE (¥35) TO RR. 　　WRITE RR. 　　GO TO L13.

8.2.7 共通ルーチン

A. GBC (garbage collector)

GBCは、FNL(free node list) 上に自由ノードがなくなった場合に、FNS(free node space) 上の未使用ノードをFNLに回収するprocedureである。新しくノードを生成しようとしたときに、未使用のノードが無かったとき、garbage collectionが行われるが、一方、コマンドの1つとしてもGBCは用意されているので、任意の時点でgarbage collectionを行うこともできる。

GBCは、次の2つのステップからなる。

1) 使用中のノードのマーク

2) 未使用ノードのFNLへの回収

現在、使用中の木(tree)のhead pointerは、次のようなところに格納されている。

1) pointer register 内に保存されているもの

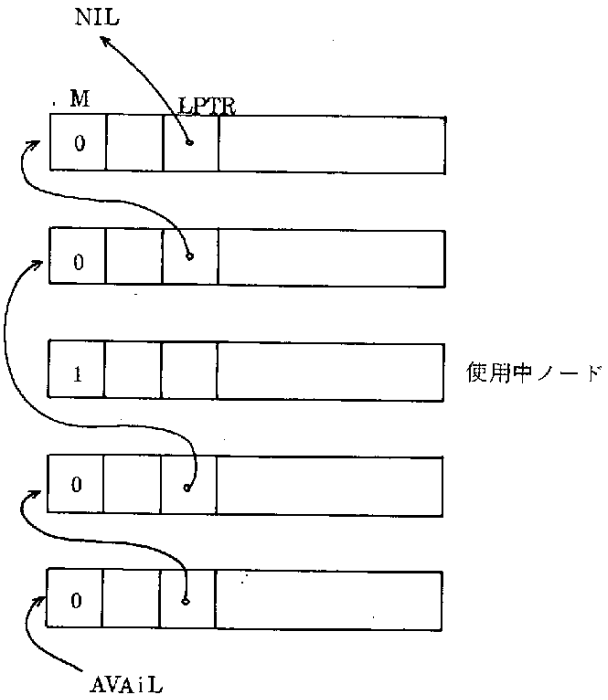
2) スタック領域内にpushdown されているもの(スタックの使用方法については後述)

3) テーブル(VBM、RTB)の内に格納されているもの

などがある。これらのさすノードをUSED(=2)とマークする。これらの全てのtreeをなぞりながら、使用中ノードをマークしてゆく。使用中ノードの表示は、各ノードのM(先頭1バイト)を、MARKED(=1)にすることにより行う。

このようにして、全ての使用中ノードのマークが終了したのち、マークされていないノード(即ち、M(P)=0 のノード)をFNLへ回収するとともに、マークされているノードのマークをはずす。未使用ノード回収の手順は、次のようになる。

```
AVAIL=NIL;  (初期値セット)
do i =MINF to MAXF;  (FNSの全ノードについて)
   if M(i) = 0
       then  do ;
                 LPTR(i) = AVAIL;
                 AVAIL   = i;
             end ;
       else do  ;
                 M(i)=0;  (マークをはずす)
             end ;
end ;
```



treeのhead pointerをpとしたとき、このtree内のノードを全てマークするアルゴリズムを示す。

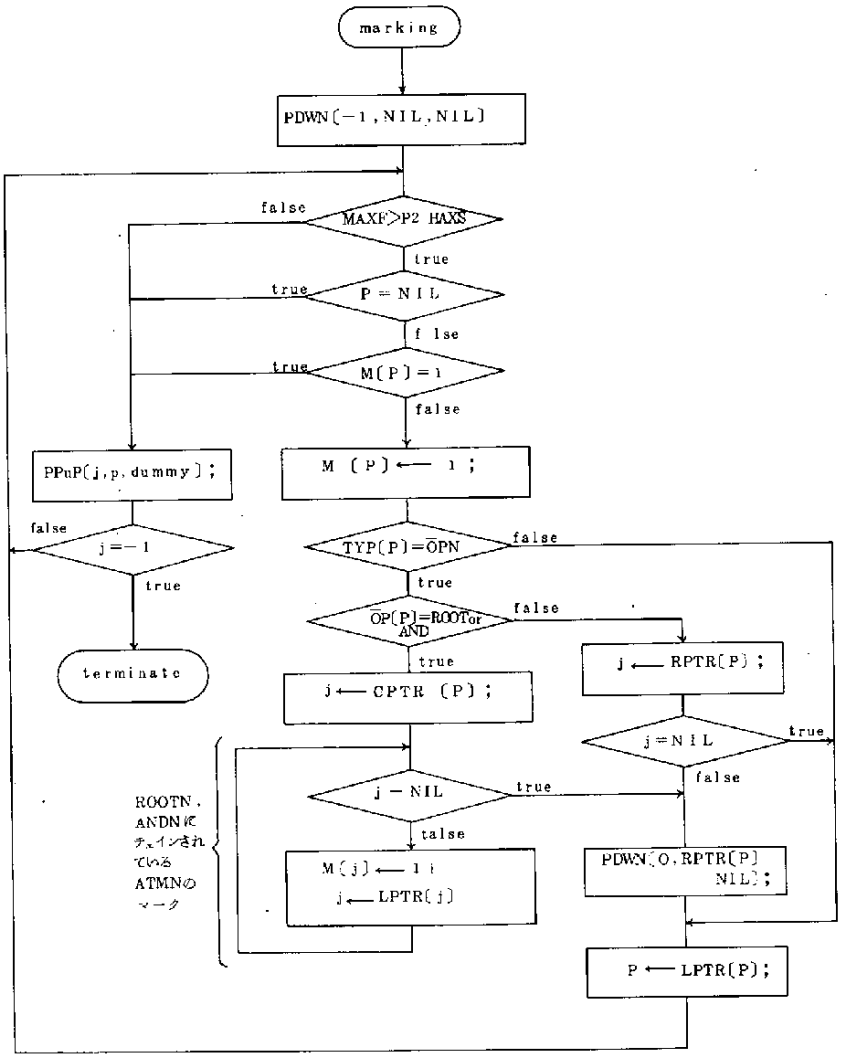


図8－84　markingのアルゴリズム

B．スタックの利用方法について

スタック領域は、主記憶内の連続領域である。スタックの１つの要素は、連続した３ワード(12バイト)から成る。

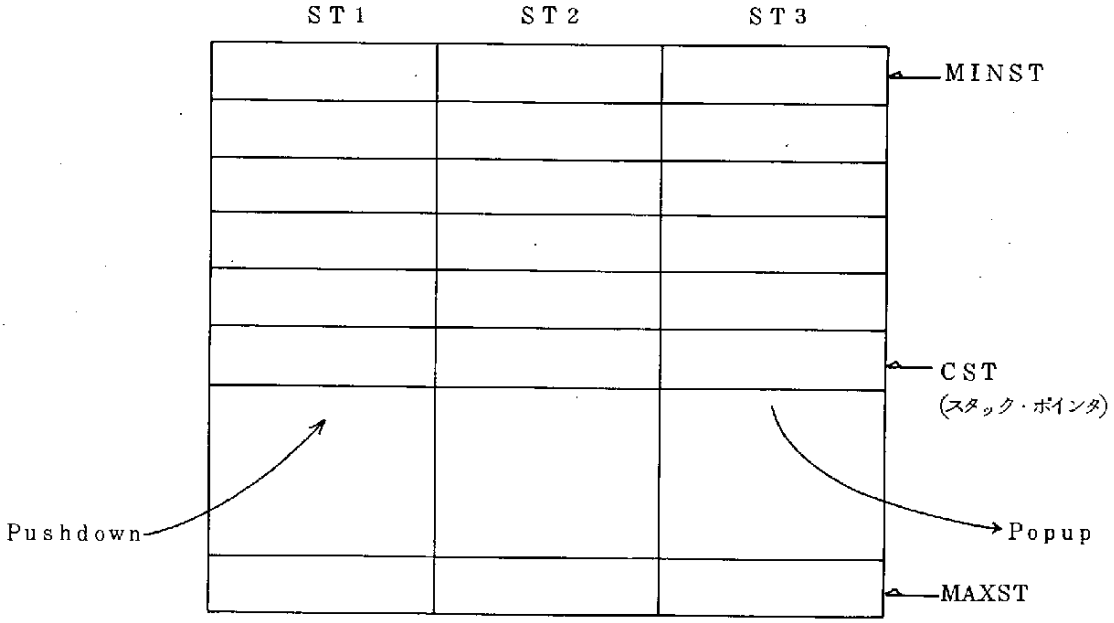


図８－８５　スタック領域の構成

スタックの利用については、次のような規則を守り、garbage collection 時にスタックの内容がポインタであるかどうかを明確にする。

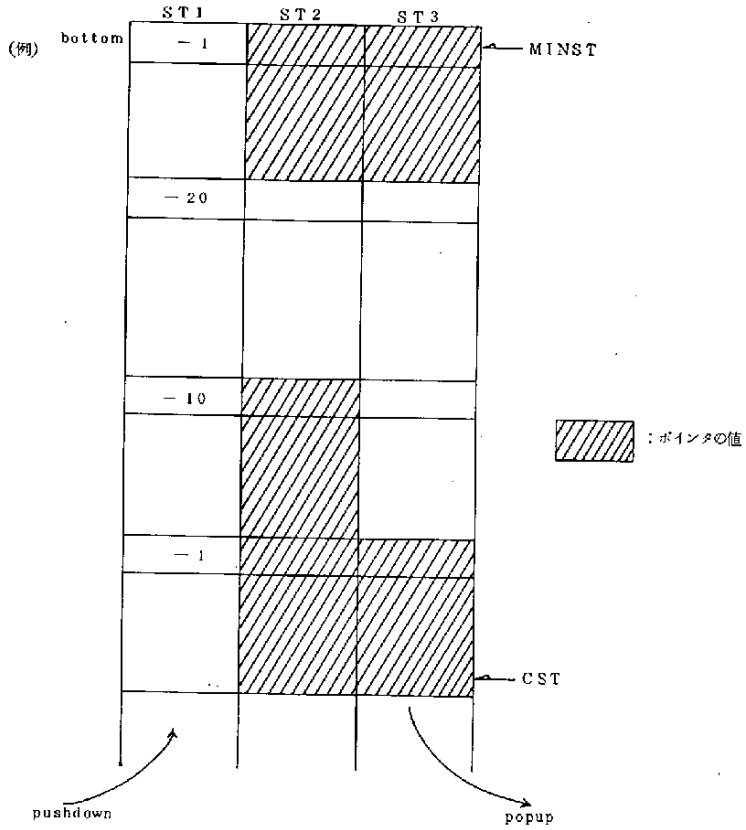
1) ＳＴ１には、決してポインタをセットしない。

2) ＳＴ２、ＳＴ３の両方にポインタを格納する時は、fenceとして（ｉ、ｂ、ｃ）をpush－downする。ここで、－９≦ｉ≦－１である。

3) ＳＴ２はポインタで、ＳＴ３はポインタではないときは、fenceとして(ｊ、ｂ、ｃ)をpushdownする。ここで、－１９≦ｊ≦－１０である。

4) ＳＴ２、ＳＴ３ともにポインタではないときは、fenceとして(ｋ、ｂ、ｃ)をpushdownする。ここで、－２９≦ｋ≦－２０である。

（例）



C．OUTP

OUTPは、ある木（tree）のポインタが与えられたとき、このtreeからinfix notationに変換するprocedureである。但し、<target－list> treeは変換の対象とせず、<qualification>（<a－exp>も含む）を変換するものである。

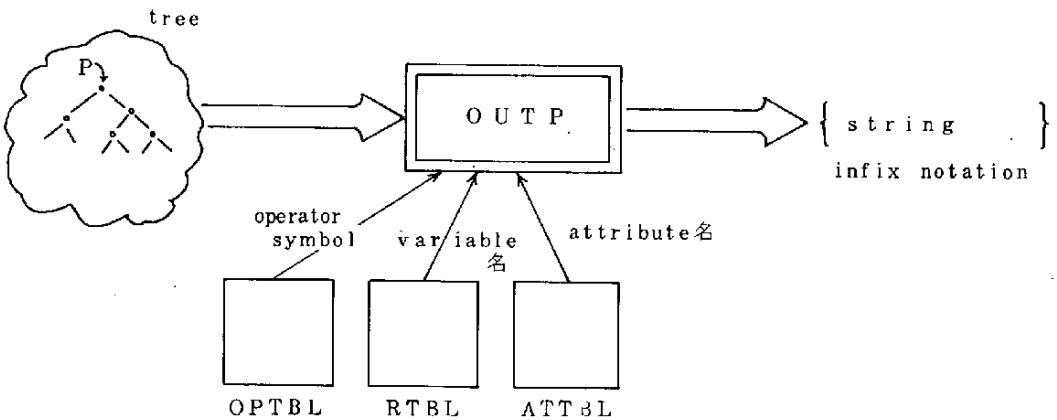
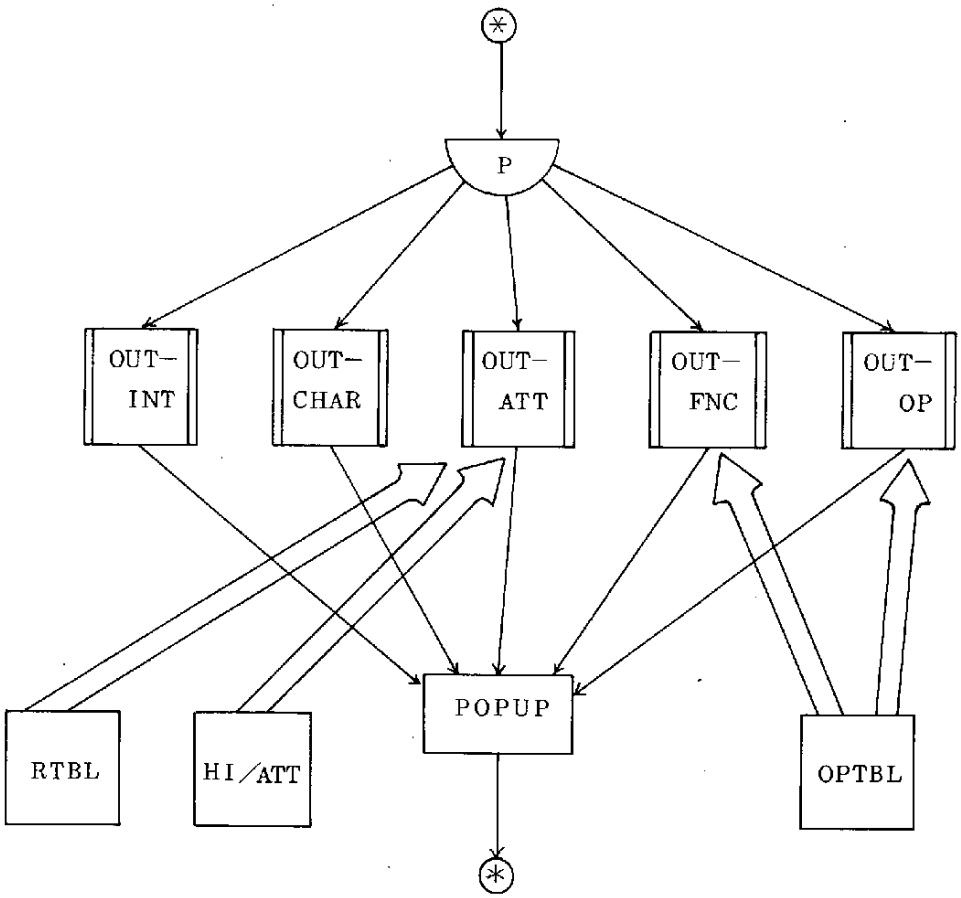


図8－86　OUTPの概要



ＯＵＴＩＮＴ ： 整数値を出力

ＯＵＴＣＨＡＲ: 文字列を出力

ＯＵＴＡＴＴ ： attributeを出力（ＨＩ／ＡＴＴとＲＴＢＬを用いる）

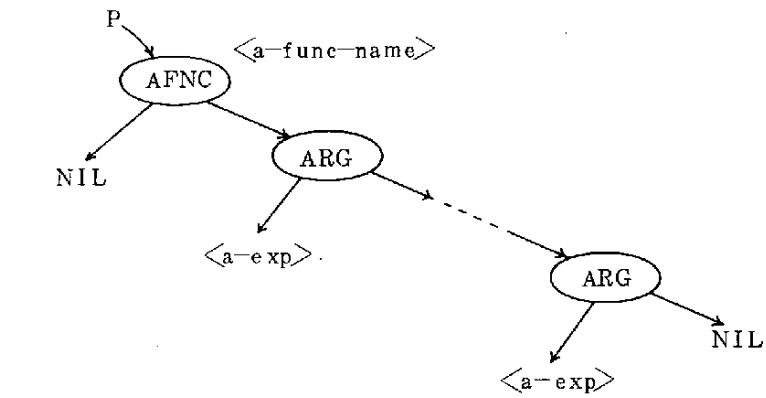
ＯＵＴＦＮＣ ： 関数の出力（ＯＰＴＢＬを用いる）

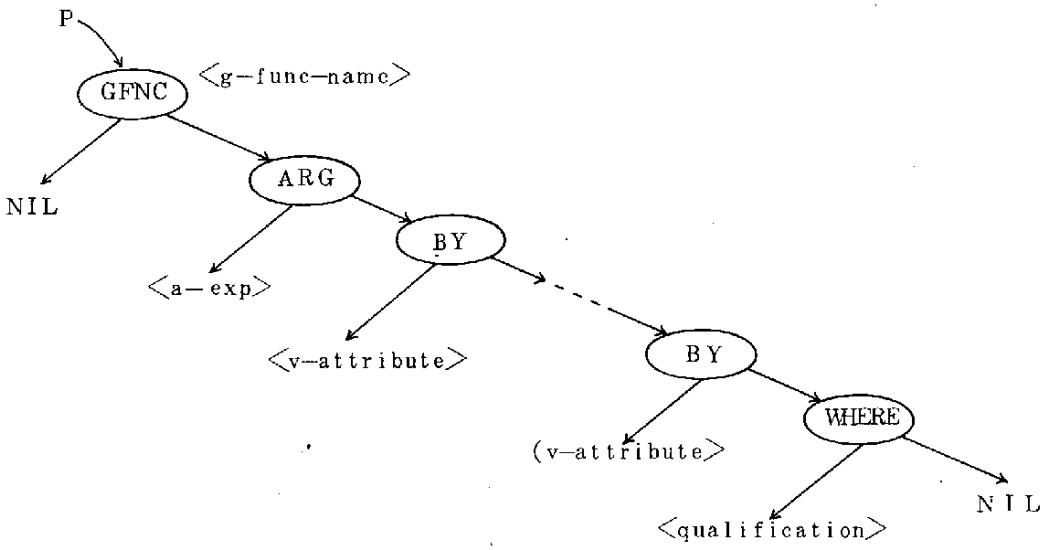
関数の引数は、＜e−exp＞又は＜qualification＞なのでＯＵＴＰを recursive callとする。

ＯＵＴＯＰ ： 通常のoperatorの出力（ＯＰＴＢＬを用いる）

図８－８７ ＯＵＴＰの構成

ＯＵＴＦＮＣ　procedureは、算術（arithmetic）及び、aggregate関数の出力を行う。



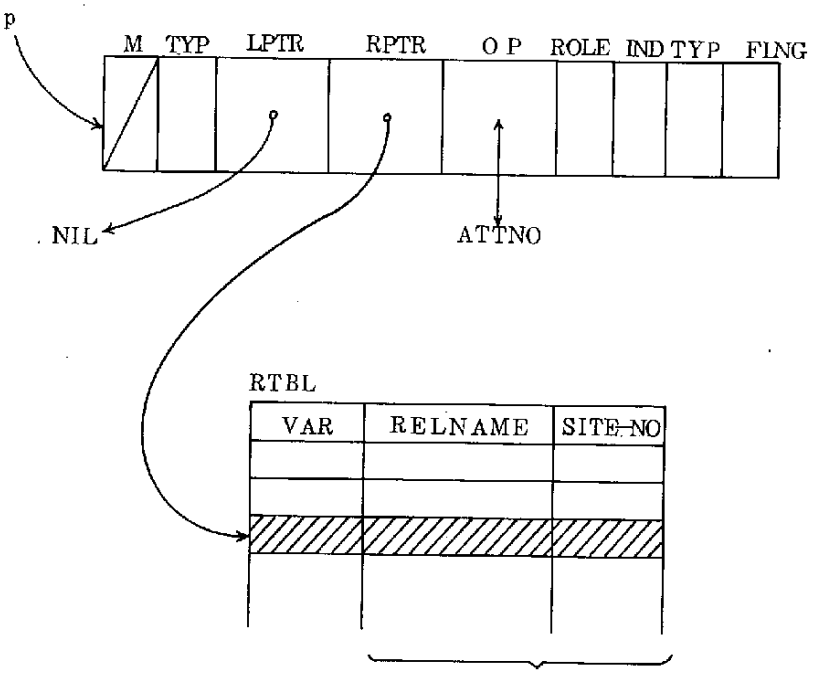


関数の引数は、一般に算術式、又は、論理式なので、引数の出力には、ＯＵＴＰをrecursiveにcallすることにより行う。このため、ＯＵＴＦＮＣからＯＵＴＰをcallするときは、（－２、ａ、ｂ）をpushdownしておく。　ＯＵＴＰでは、スタックをpopupして、indicatorが－２であればＯＵＴＦＮＣに復帰する。

ＯＵＴＡＴＴは、ＲＴＢＬとＨＩ／ＡＴＴを用いて、

"＜variable＞・＜attribute＞"

を出力するprocedureである。ｐはＡＴＴＮ（属性ノード）へのポインタである。



HI／ATTをRELNAMEとATTNO、及びTYPE（"ES"でアンマッチなら'RS'で）をKEYにしてATTRIBUTE NAMEを得る。

従って、このＯＵＴＡＴＴだけは、ＡＴＴＮの意味づけがプロセス毎に異なるので、共通には使用できないことがある。

8.3 ＤＩＰＳ（Distribution Information Processing System）

ＤＩＰＳは、分散記述（ＤＤ）を解読し、サイト情報（ＬＩ）を用いてＧＣＳレベルでのＮＤＤを生成するものである。生成される情報を分散情報（ＤＩ）と呼ぶ。ＤＩには、スキーマ情報（ＳＩ）と対応情報（ＣＩ）とがある。

スキーマ情報は、ＤＤで定義されるグローバルレベルのＥＳＲ、ＲＳＲを一元的に管理するためのものである。一方、対応情報は、ＧＣＳリレーションとＬＣＳリレーションとの対応を表わしている。

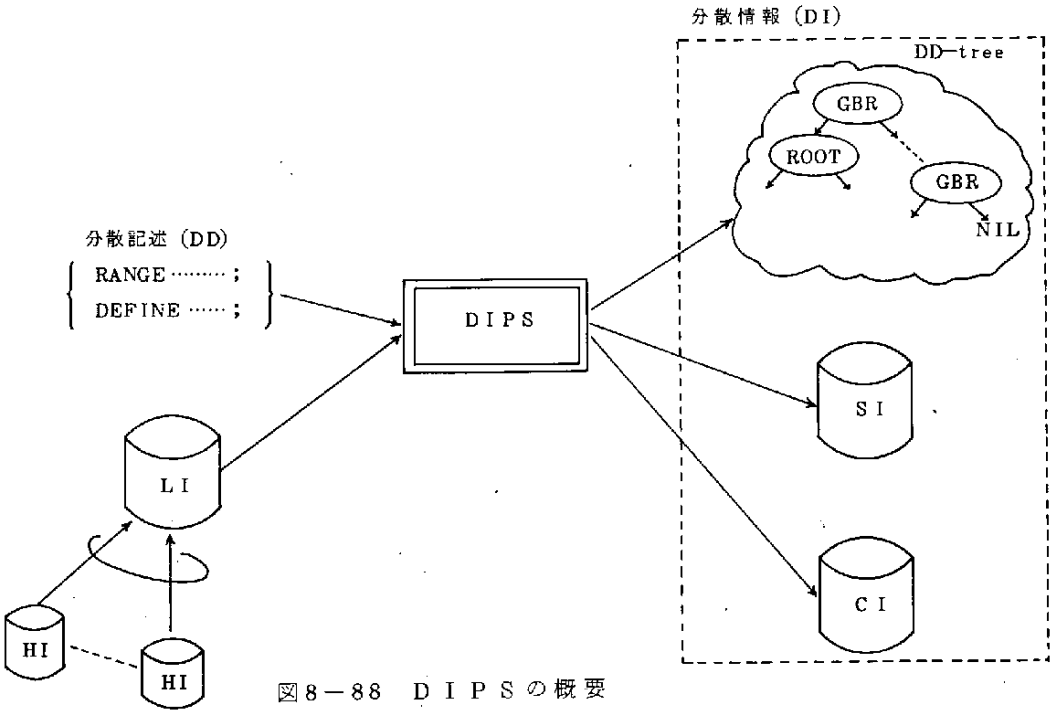


図８－８８　ＤＩＰＳの概要

8.3.1　分散記述（Distribution Description　：ＤＤ）

分散記述は、ＬＣＳ（局所概念スキーマ）のＲＤ（relational description）内のＬＣＳリレーションから、新たに、ＧＣＳ（全体概念スキーマ）におけるＧＣＳリレーション（ＥＳＲとＲＳＲ）を定義するためのものである。

ＤＤは、ＱＵＥＬ言語と同様な記述言語（ＧＳＤＬ）〔6.2を参照〕を用いて記述される。また、ＤＤにおいて定義されるべき事項としては、次のものがある。

1）ＧＣＳリレーション名

2）ＧＣＳリレーションを構成するＧＣＳ属性名と、その領域（domain）名、データフォーマット

3）ＧＣＳリレーション名とＬＣＳリレーション名との対応

4）ＧＣＳ属性名とＬＣＳ属性名との対応

5）ＧＣＳリレーションを構成する組（ tuple ）とＬＣＳリレーションを構成する組との対応（オカーレンスの対応）

6）ＬＣＳリレーション名とその所在サイト番号

ＲＡＮＧＥ文は、3）と6）を定義し、ＤＥＦＩＮＥ文は、1）、2）、4）、5）を定義するものである。ＧＳＤＬの形式的定義は、付記１に示す。

８．３．２　分散記述の内部表現

分散記述（ＤＤ）の木表現をＤＤ－ｔｒｅｅと呼ぶ。ＤＤ－ｔｒｅｅは、ＱＴＰで示したＱ－ｔｒｅｅと同様の２分木（ binary tree ）構造を持っている。即ち、ＤＤ－ｔｒｅｅにおける＜target－list＞と＜qualification＞の木表現はＱ－ｔｒｅｅと同じである。ＤＤには、一般に＜sub-def＞が複数あるので、これを表現するためにＱ－ｔｒｅｅには無かったＧＢＲノードを生成する。

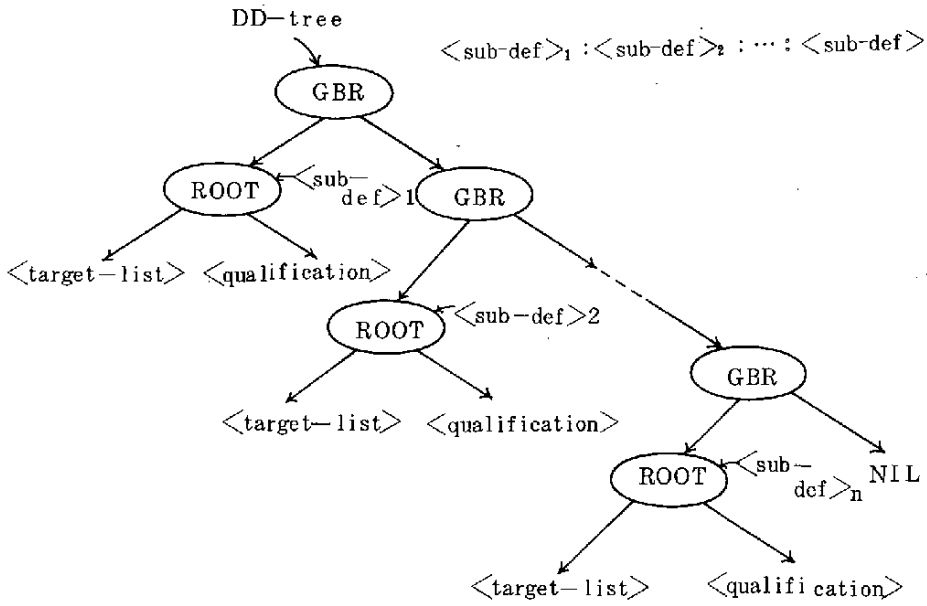


図８－８９　ＤＤ－ｔｒｅｅ

また、ＤＤの内部表現としては、ＤＤ－ｔｒｅｅの他に、ＲＴＢＬ、ＧＲＴＢＬ、ＧＡＴＢＬ、ＳＮＴＢＬと呼ばれる、４つのテーブルがある。

ＲＴＢＬ〔表８－２７〕は、組変数名とリレーション名、及び、その所在サイト番号との対応表である。ＲＡＮＧＥ文で宣言された順に、テーブルの先頭から格納される。

ＧＲＴＢＬ〔表８－２９〕には、ＤＥＦＩＮＥ文で定義されたＧＣＳリレーションと、その次数（ degree ）とが格納される。

ＧＡＴＢＬ〔表８－３０〕は、ＧＣＳリレーションを構成する属性の名前と、その属性番号、領域（ domain ）名、フォーマットとの対応表である。

ＳＮＴＢＬ〔表８－２８〕は、ＤＤ内で参照されているサイト番号が格納される。ＳＮＴＢＬ内のサイト番号の格納されている組番号は、ＡＮＤノードとＲＯＯＴノードにおけるbit-map（ＢＳＬとＢＳＲ）のビット位置に対応している。

分散記述：

D：RANGE OF （l1、l2、…、ln） （L1、L2、…、Ln）；

DEFINE ＜rel-type＞ ＜grel-name＞ （＜a-list＞）

＜sub-def＞ ｛：＜sub-def＞｝；

表８－２７　ＲＴＢＬの構成

属性名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　明
ＶＡＲ	1	Ｋ	Ｃ	4	組変数（tuple variable）名
ＲＥＬＮＡＭＥ	2	Ｎ	Ｃ	１６	リレーション名
ＳＩＴＥ－ＮＯ	3	Ｎ	Ｄ	2	サイト番号

表８－２８　ＳＮＴＢＬの構成

属性名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　明
ＳＩＴＥ－ＮＯ	1	Ｋ	Ｄ	2	サイト番号

表８－２９　ＧＲＴＢＬの構成

属性名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　明
ＧＲＥＬＮＡＭＥ	1	Ｋ	Ｃ	１６	ＧＣＳリレーション名
ＲＴＹＰＥ	2	Ｎ	Ｃ	2	ＥＳＲ　ｏｒ　ＲＳＲ
ＤＥＧＲＥＥ	3	Ｎ	Ｄ	2	このリレーションを構成する属性の数

表８－３０　ＧＡＴＢＬの構成

属性名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　明
ＡＴＴＮＯ	1	Ｋ	Ｄ	2	属性番号
ＧＡＴＴＮＡＭＥ	2	Ｎ	Ｃ	１６	ＧＣＳ属性名
ＤＮＡＭＥ	3	Ｎ	Ｃ	１６	ＧＣＳ属性の属する領域（domain）名
ＦＴＹＰＥ	4	Ｎ	Ｄ	1	タイプ（Ｃ＝１、Ｄ＝２）
ＦＬＥＮＧＴＨ	5	Ｎ	Ｄ	2	バイト長

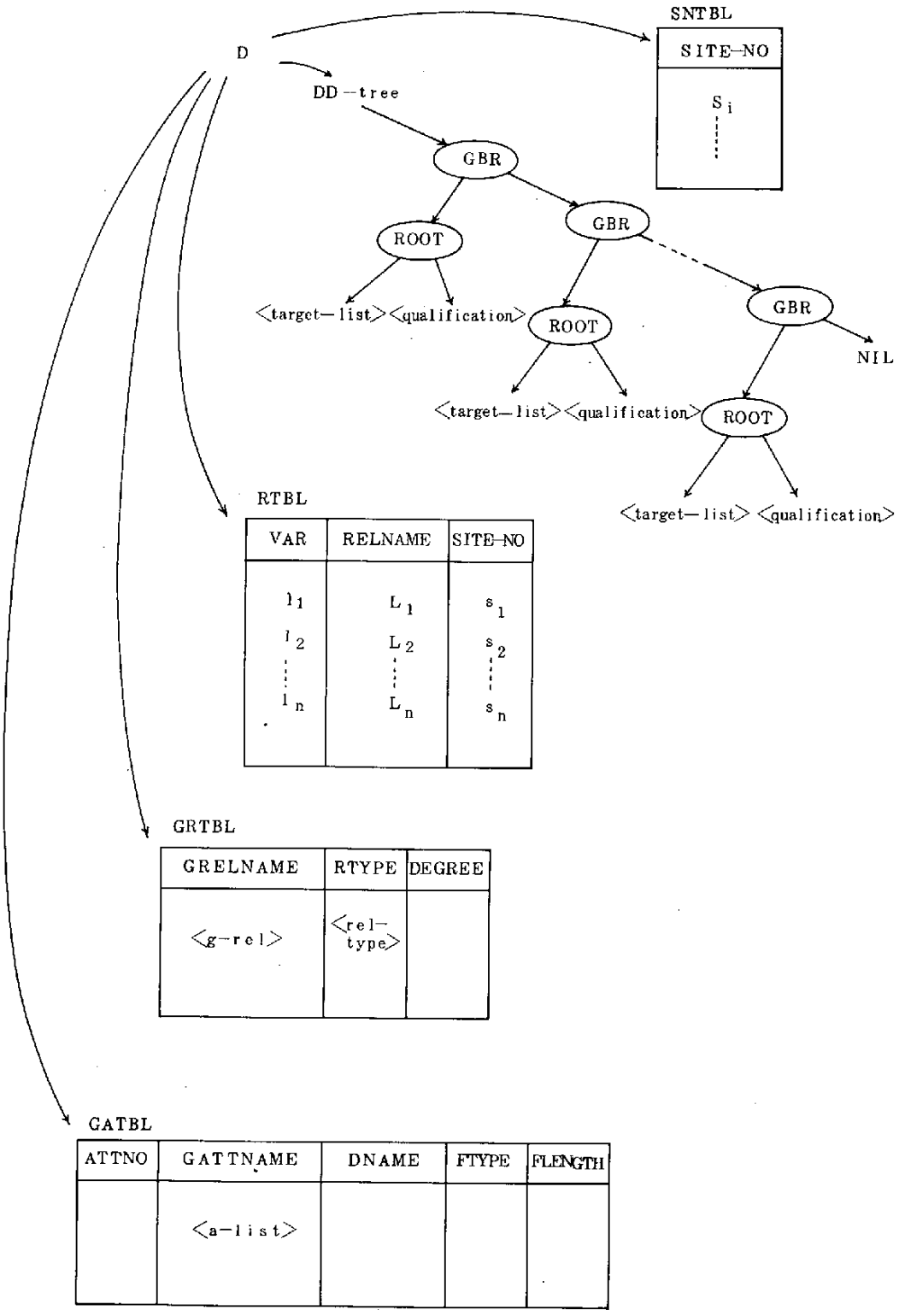


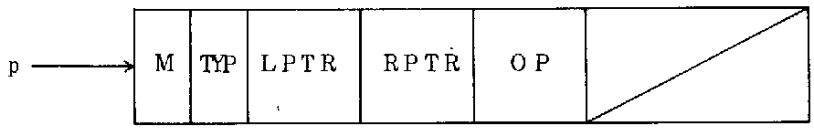
図８－９０　分散記述と内部表現

8.3.3　DD－tree におけるノード構造

DD－tree に使用されるノードの構造は、ＱＴＰのＱ－tree と全く同じである。詳細は、8.2.2.A「Ｑ－tree におけるノード構造」に示してある。従って、ここでは、分散記述に固有な＜sub-def ＞を表わすノードとしてのＧＢＲノードについてのみ説明する。

A．　ＧＢＲノード

ＧＢＲノードは、演算子ノードの１つとして表わされる。



フィールド名	説　　明
TYP	TYP（ｐ）＝５　（=OPN）
LPTR	＜sub-def ＞ を表わす木（ tree ）へのポインタ
RPTR	次のＧＢＲノードへのポインタ、無ければＮＩＬ
OP	OP（ｐ）=ＧＢＲ　（３１）

8.3.4　TGNR（Tree Generator ）

ＴＧＮＲは、ＧＳＤＬ仕様に基づいた分散記述（ＤＤ）の文字列を入力として、対応する２分木（ binary tree ）及び、テーブル表現を生成するプロセスである。

ＴＧＮＲは、ＲＡＮＧＥ文を処理するＲＡＮＧＥＰと、ＤＥＦＩＮＥ文を処理するＤＥＦＩＮＥＰとに分けることができる。ＤＥＦＩＮＥＰは、＜a-list ＞までを処理するＧＲＥＬＰ、＜target-list ＞を処理するＴＡＲＧＰ、＜qualification ＞を処理するＱＵＡＬＰというサブプロセスから成り立っている。以下、順を追って各プロセスについて説明するが、その中の状態遷移図で使用している記号は、ＱＴＰの場合〔図８－８〕と同じである。

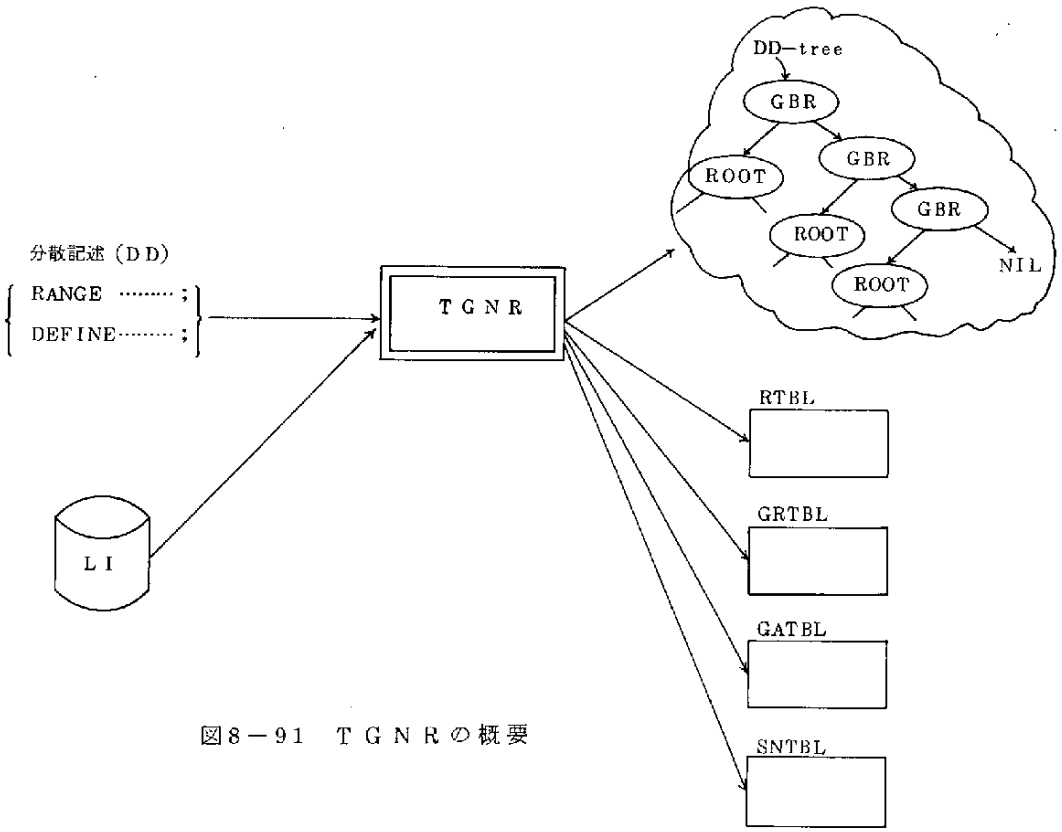


図８－９１　ＴＧＮＲの概要

８．３．５　ＲＡＮＧＥＰ

ＲＡＮＧＥＰは、文字列としてのＲＡＮＧＥ文を入力として、ＮＤＤ／ＬＩを用いて、ＲＴＢＬ（ range table ）、ＳＮＴＢＬ（ site−no table ）を生成するプロセスである。

ＲＡＮＧＥＰの状態遷移図を、図８－９２に示す。＜ｓｉｔｅ － ｎｏ＞が指示されている場合は、ＬＩをサーチして対応するＬＣＳリレーションがあるかどうかをチェックする。もし＜ｓｉｔｅ－ｎｏ＞の指定がなければ、ＬＩから対応するサイト番号をもってくる。ＬＩはＬＣＳ要素と所在サイトとの対応情報であり、表８－３２のようなスキーマをもつ。

Ａ　所在情報（ＬＩ）

所在情報は、ＤＤのＲＡＮＧＥ文の解読に必要なサイト情報をもっている。このサイト情報は、ＨＩＰＳ〔８.１〕で生成したＨＩ情報のうち、ＨＩ／ＥＳＲ、ＨＩ／ＲＳＲ、ＨＩ／ＡＴＴという３つのリレーションを用いて生成する。但し、これらには、所在サイトに関する情報は含まれていないので、この他にＨＩ／ＬＡＢＥＬ〔表８−３１〕という新しいリレーションを必要とする。

今回のインプリメンテーションでは、ＨＩ／ＬＡＢＥＬリレーションの生成は行なっていないので、直接、カードからＬＩ／ＡＴＴを作成している。ＬＩ／ＡＴＴリレーションのスキームを表８−３２に示す。

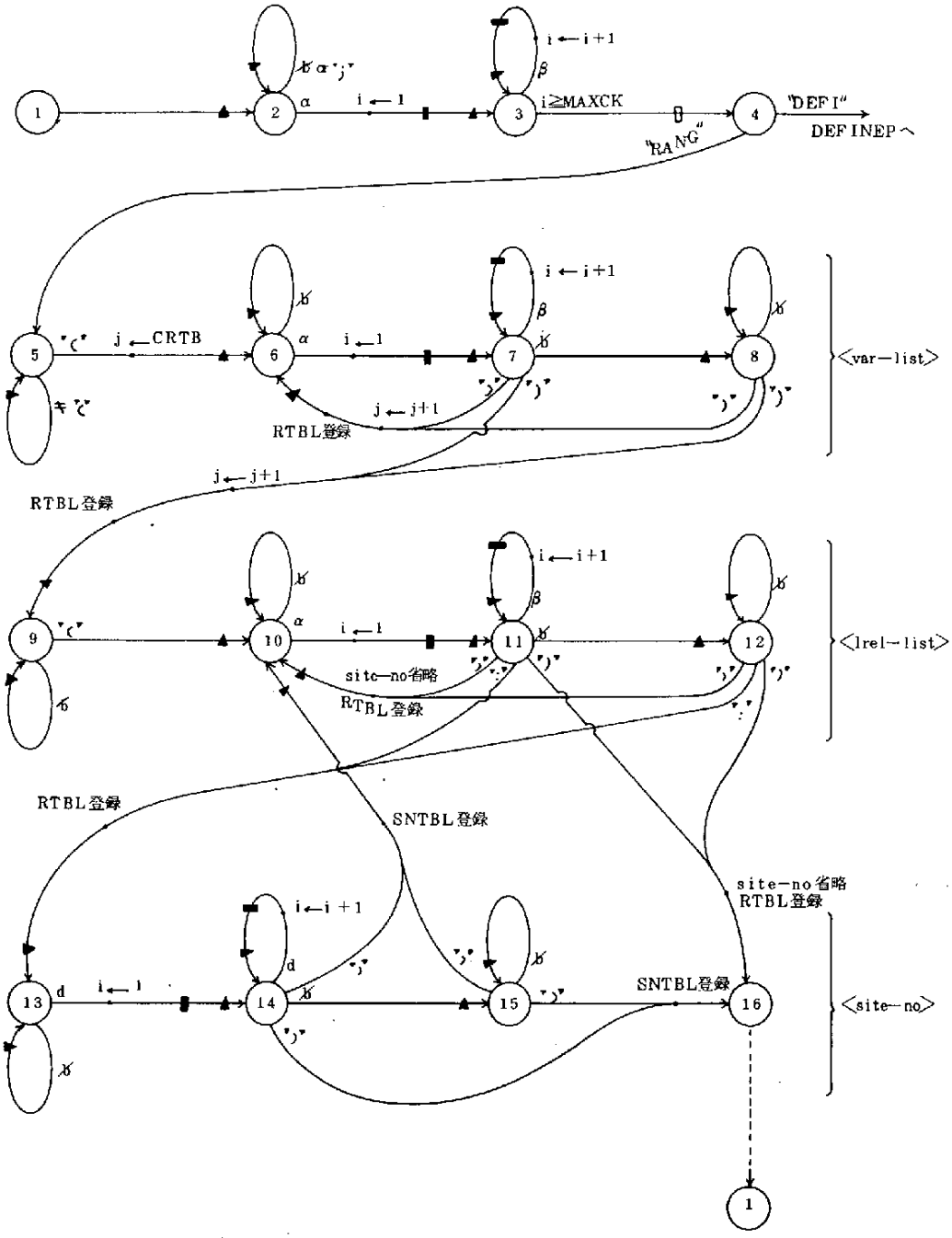


図8－92　RANGEPの状態遷移図

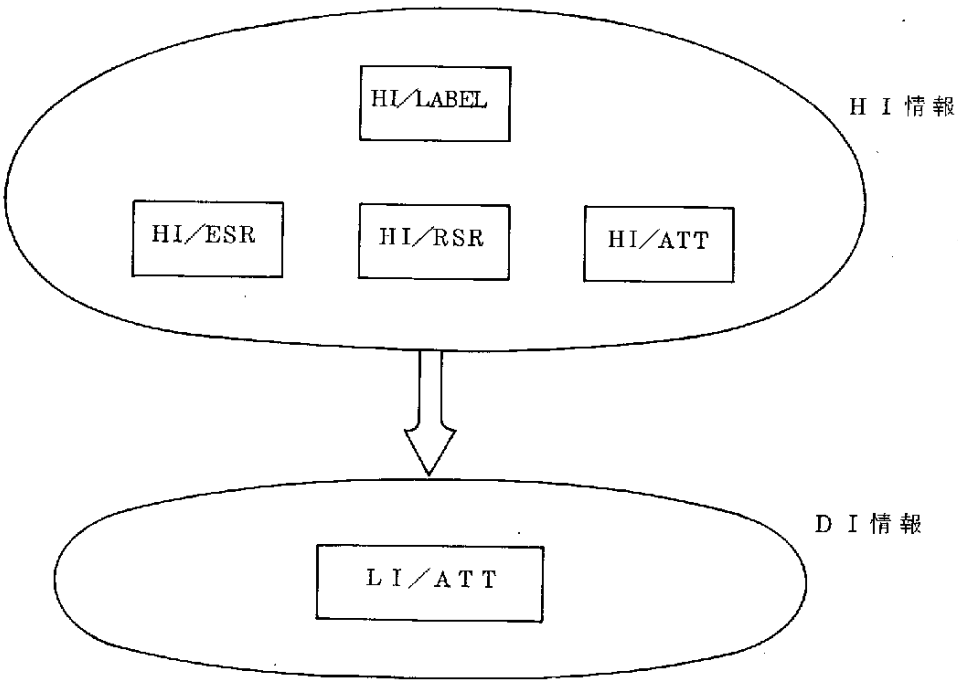


表8－31　　HI／LABELリレーション

属　性　名	属性番号	ROLE	TYPE	LENGTH	説　　明
HOST-ID(SITE)	1	C1	D	4	サイト番号
DATABASE-NAME	2	C2	C	16	データベース名
SCHEMA-NAME	3	C3	C	16	スキーマ名
RELATION-TYPE	4	C4	C	20	リレーションのタイプ（ESR or RSR）
RELATION-NAME	5	C5	C	16	リレーション名
REAL-RELNAME	6	N	C	16	実際のリレーション名
OWNER	7	N	C	8	このリレーションの所有者
CREATE-DATE	8	N	C	6	生成日
UPDATE-DATE	9	N	C	6	更新日
PURGE-DATE	10	N	C	6	消去される予定日
DEGREE	11	N	D	2	次数（属性名）
WIDTH	12	N	D	4	組の幅（size）
CARDINALITY	13	N	D	4	カーディナリティ(組数)
SIZE	14	N	D	8	リレーションのサイズ（=WIDTH *CARDINALITY）
ACCESS-METHOD	15	N	C	6	アクセス方法

表8－32　LI／ATT リレーション

属　性　名	属性番号	ROLE	TYPE	LENGTH	説　明
LOCAL-RELNAME	1	K	C	16	LCSリレーション名
ES-RS-TYPE	2	K	C	2	ES or RS
LOCATION-ID	3	N	D	4	所在サイト番号
LOCAL-ATTNAME	4	K	C	16	LCS属性名
ROLE	5	N	C	5	この属性の役割
TYPE	6	N	C	1	データの型
LENGTH	7	N	D	3	データのバイト長
DOMAIN	8	N	C	16	領域名
ATTRIBUTE-NO	9	N	D	3	属性番号
TID	10	N	C	8	組識別子
FILLER	11	N	C	6	

8.3.6　DEFINEP

DEFINEPは、文字列としてのDEFINE文を入力として、DD－tree およびGRTBL、GATBLを生成するプロセスである。このとき、RANGEPによって生成されたRTBL、SNTBLも利用する。DEFINEPは、GRELP、TARGP、QUALPという、3つのサブプロセスから成る〔図8－93〕。また、サブルーチンQUALについては、QTPで使用しているものと同じである〔 8.2.3.Eを参照〕。

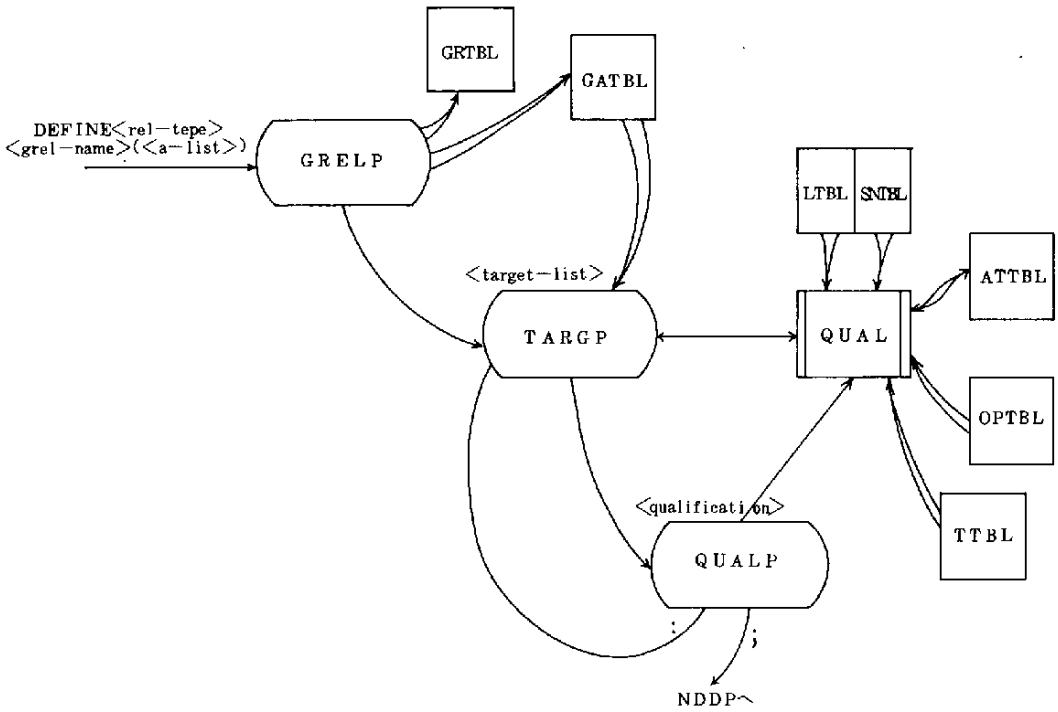


図８－９３　ＤＥＦＪＮＥＰの構成

Ａ　ＧＲＥＬＰ

ＧＲＥＬＰは、文字列

〝ＤＥＦＩＮＥ　＜rel−type＞　＜grel−name＞（＜a−list＞）〟

を入力として、ＧＲＴＢＬとＧＡＴＢＬを生成する〔表８－２９と表８－３０〕。ここで生成されたＧＲＴＢＬとＧＡＴＢＬは、ＮＤＤＰにおいて、ＮＤＤ／ＳＩ、ＮＤＤ／ＣＩの生成に使用される。ＧＲＥＬＰの状態遷移図を、図８－９４に示す。

Ｂ　ＴＡＲＧＰ

ＴＡＲＧＰは、文字列〝＜target−list＞〟を入力として、＜target−list＞treeを生成するプロセスである。＜target−list＞内の＜a−exp＞の処理には、サブルーチンＱＵＡＬを用いる。＜target−list＞treeにおけるＲＡＴノードの属性番号と、ＧＡＴＢＬ内の属性番号は、対応していなければならない。また、ここでは、＜t−clause＞::=＜attribute＞=＜a−exp＞と仮定している。即ち、＜t−clause＞::=＜v−attribute＞という形式は許されない。

ＴＡＲＧＰの状態遷移図を、図８－９６に示す。

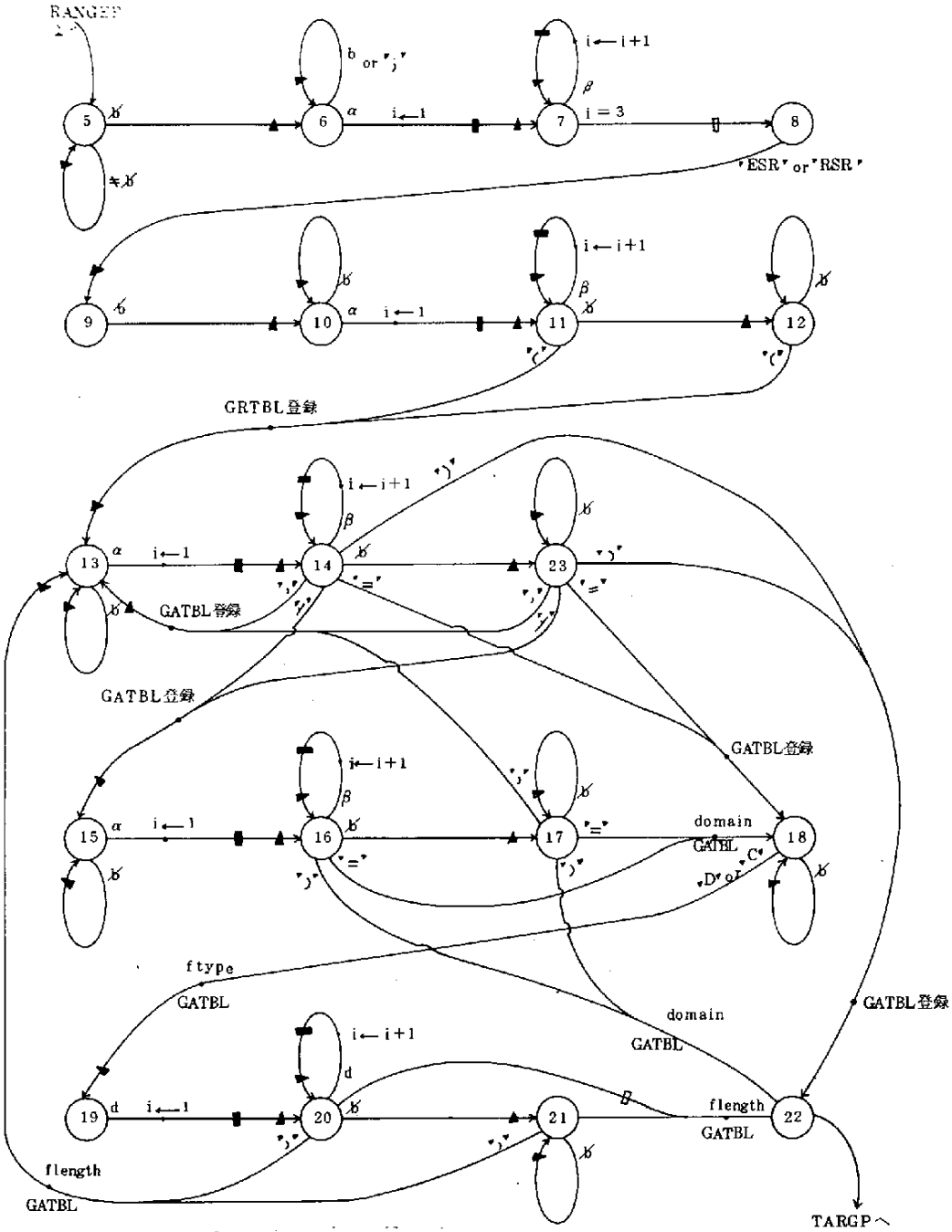


図８－９４　ＧＲＥＬＰの状態遷移図

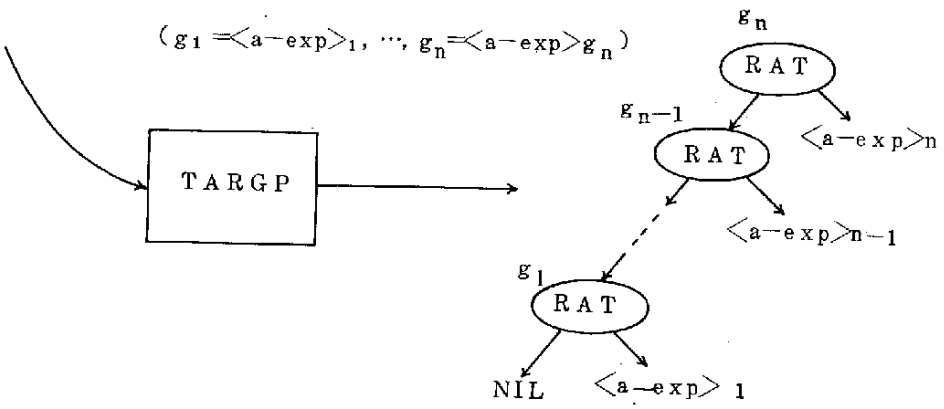


図8－95　RANGPの概要

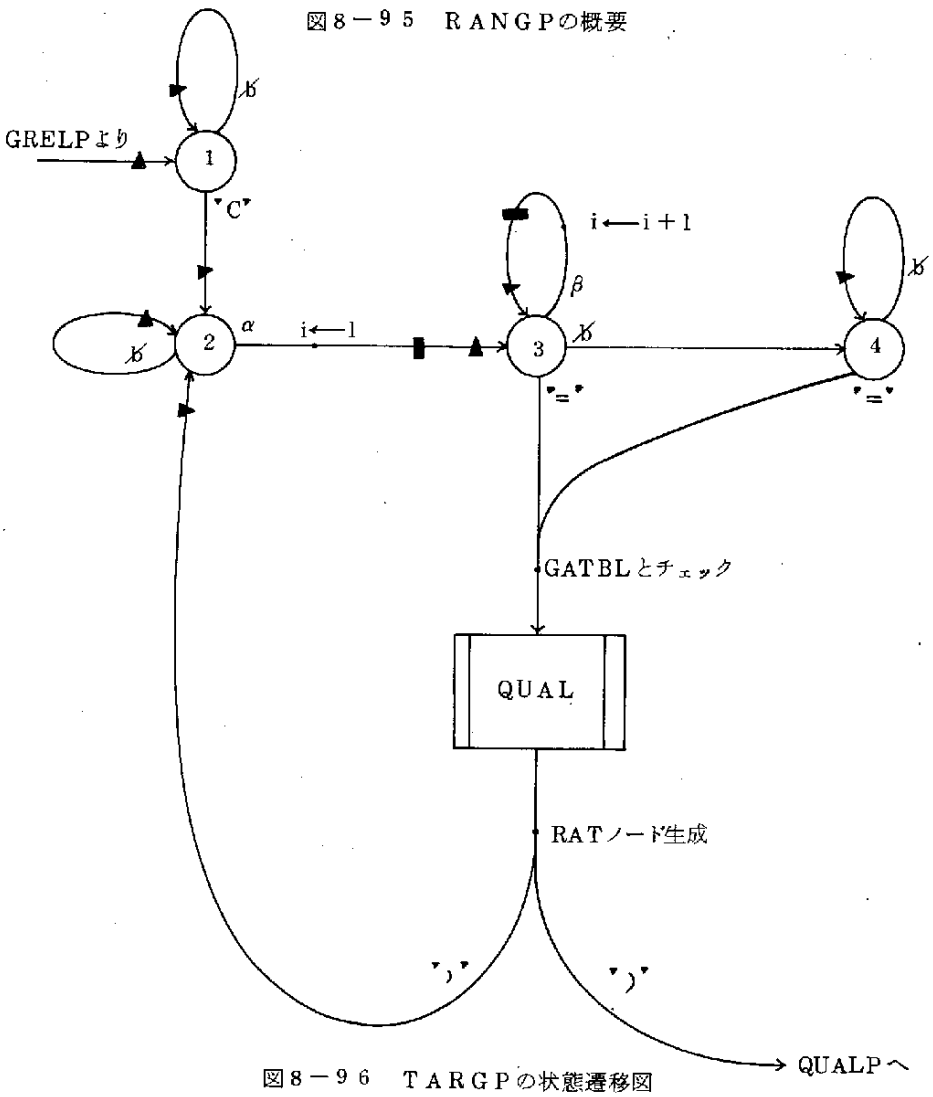


図8－96　TARGPの状態遷移図

C　QUALP

QUALPは、文字列

〝WHERE　＜qualification＞〟

を入力として、＜qualification＞treeを生成するプロセスである。＜qualification＞部の処理には、TARGPと同じく、procedure QUALを用いる。QUALPの状態遷移図を、図8－97に示す。

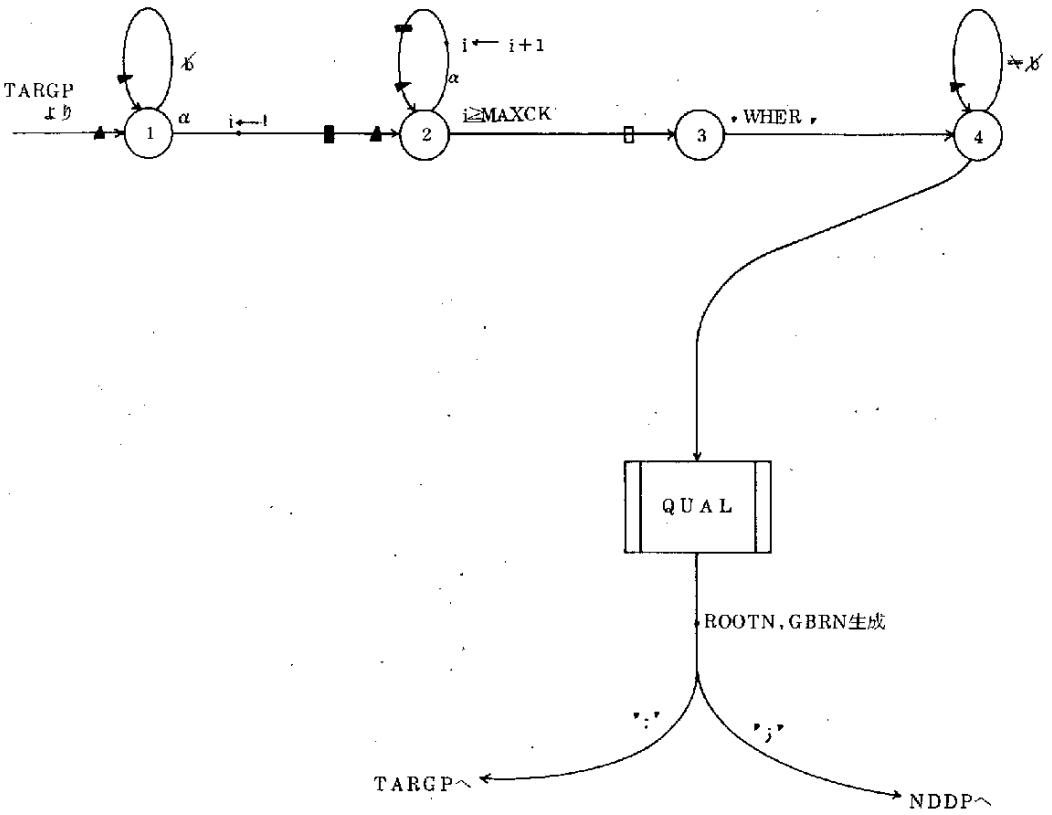


図8－97　QUALPの状態遷移図

8.3.7　NDDP

NDDPは、RANGEP、DEFINEPにおいて生成されたテーブルおよびDD－treeを用いて、NDD／SIとNDD／CIを生成するプロセスである。

NDD／SIの、DI－SI－RELATION及びDI－SI－ATTリレーションへの組（tuple）の追加は、1つのDEFINE文の処理が終る毎に行なわれる。即ち、1つのGCSリレーションには、GCSリレーションについて、DI－SI－RELATIONに1つの組を追加する。また、DI－SI－ATTリレーションには、GCSリレーションを構成する属性の数だけ組を追加する。

NDD／CIのDI－CI－GRELリレーションには、1つのDEFINE文の処理終了ごとに、DD－treeのノード数だけ組を追加する。一方、DI－CI－RANGEとDI－CI－SITEリレーションは、全てのDDの処理が終った後に、各々、RTBLとSNTBLとから生成する。

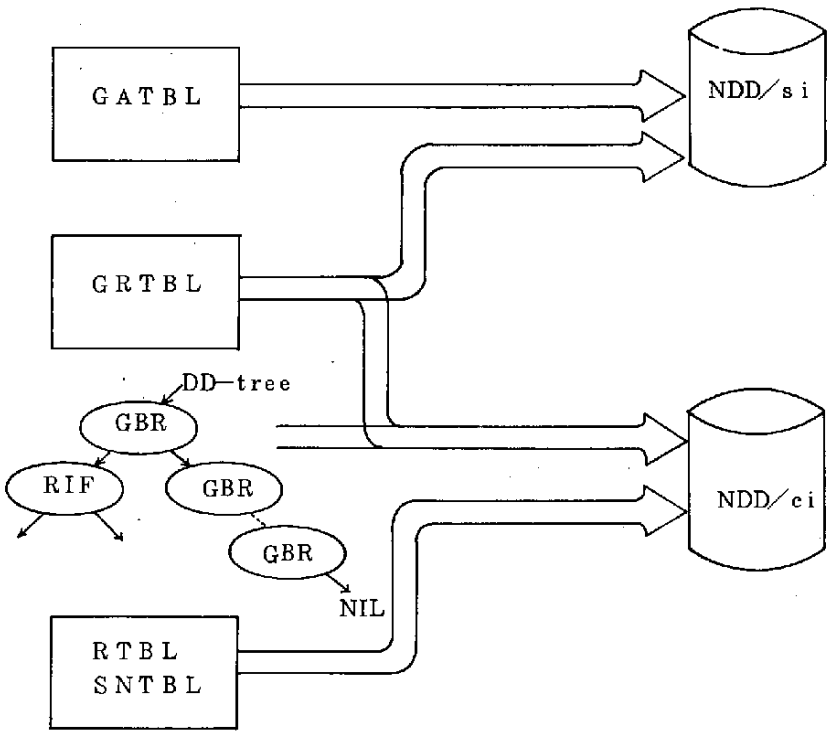


図8－98　NDDPの概要

A　スキーマ情報（NDD／SI）

スキーマ情報は、全体リレーション（ESR、RSR）の名前と、そのリレーションを構成する属性に関する情報を表わしている。DI－SI－RELATION〔表8－35〕はGRTBLから、DI－SI－ATT〔表8－36〕はGRTBLとGATBLから生成される。これらの構成を、表

8－３３と表８－３４に示す。

表８－３３　　ＤＩ－ＳＩ－ＲＥＬＡＴＩＯＮの生成

ＤＩ－ＳＩ－ＲＥＬＡＴＩＯＮの項目	転　送　内　容
*) ＧＬＯＢＡＬ－ＲＥＬＮＡＭＥ	ＧＲＴＢＬのＧＲＥＬＮＡＭＥ
*) ＥＳ－ＲＳ－ＴＹＰＥ	ＧＲＴＢＬのＲＴＹＰＥ
ＤＥＧＲＥＥ	ＧＲＴＢＬのＤＥＧＲＥＥ
ＴＩＤ	ＢＬＡＮＫ
ＦＩＬＬＥＲ	ＢＬＡＮＫ

表８－３４　　ＤＩ－ＳＩ－ＡＴＴの生成

ＤＩ－ＳＩ－ＡＴＴの項目	転　送　内　容
*) ＧＬＯＢＡＬ－ＲＥＬＮＡＭＥ	ＧＲＴＢＬのＧＲＥＬＮＡＭＥ
*) ＥＳ－ＲＳ－ＴＹＰＥ	ＧＲＴＢＬのＲＴＹＰＥ
*) ＧＬＯＢＡＬ－ＡＴＴＮＡＭＥ	ＧＡＴＢＬのＧＡＴＴＮＡＭＥ
ＴＹＰＥ	ＧＡＴＢＬのＦＴＹＰＥ
ＬＥＮＧＴＨ	ＧＡＴＢＬのＦＬＥＮＧＴＨ
ＤＯＭＡＩＮ	ＧＡＴＢＬのＤＮＡＭＥ
ＡＴＴＲＩＢＵＴＥ－ＮＯ	ＧＡＴＢＬのＡＴＴＮＯ
ＴＩＤ	ＢＬＡＮＫ
ＦＩＬＬＥＲ	ＢＬＡＮＫ

表８－３５　　ＤＩ－ＳＩ－ＲＥＬＡＴＩＯＮリレーション

属　性　名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　明
ＧＬＯＢＡＬ－ＲＥＬＮＡＭＥ	1	Ｋ	Ｃ	１６	ＧＣＳリレーション名
ＥＳ－ＲＳ－ＴＹＰＥ	2	Ｎ	Ｃ	２	ＥＳ　ｏｒ　ＲＳ
ＤＥＧＲＥＥ	3	Ｎ	Ｄ	２	属性の数
ＴＩＤ	4	Ｎ	Ｃ	８	
ＦＩＬＬＥＲ	5	Ｎ	Ｃ	８	

（ length＝３６ ）

表８－３６　ＤＩ－ＳＩ－ＡＴＴリレーション

属　性　名	属性番号	ROLE	TYPE	LENGTH	説　明
GLOBAL-RELNAME	1	C1	C	16	ＧＣＳリレーション名
ES-RS-TYPE	2	C2	C	2	ES or RS
GLOBAL-ATTNAME	3	N	C	16	ＧＣＳ属性名
TYPE	4	N	D	1	C＝1、D＝2
LENGTH	5	N	D	3	データ・バイト長
DOMAIN	6	N	C	16	領域(domain)名
ATTRIBUTE-NO	7	C3	D	3	属性番号
TID	8	N	C	8	組識別子
FILLER	9	N	C	7	

（Length＝７２）

B　対応情報（ＮＤＤ／ＣＩ）

対応情報は、ＧＣＳリレーションとＬＣＳリレーションとの対応を表わしている。この対応は、分散記述（ＤＤ）内のＤＥＦＩＮＥ文の＜sub-def＞として表わされている。この＜sub-def＞は、relational calculus like な式表現である。従って、対応情報は、ＧＣＳリレーション名とこの定義式は、内部表現としてのＤＤ－treeとして表わされている。

ＤＩ－ＣＩ－ＧＲＥＬ〔表８－４０〕は、ＧＣＳリレーション名と、対応するＤＤ－treeとの対応を示している。即ち、ＤＩ－ＣＩ－ＧＲＥＬリレーションの１つの組（tuple）は、ＤＤ－treeの１つのノードに対応している。また、全てのノードは、postorder（帰りがけ順）〔ＫＮＵＴ７３〕に番号づけられる。但し、ＡＮＤノード、ＲＯＯＴノードには、ＡＴＭノードがチェインされているので、これを考慮して次のように番号をふる。ＡＮＤ（ＲＯＯＴ）ノードのＣＰＴＲフィールドを用いてチェインされているＡＴＭノードをｉ＋１、次のＡＴＭノードをｉ＋２とする。ＡＮＤ（ＲＯＯＴ）ノードに続く、次のノードの番号はｉ＋３から始められる〔図８－９９〕。

ＤＩ－ＣＩ－ＲＡＮＧＥ〔表８－４１〕は、ＤＤにおいて用いられたＲＴＢＬ（range table）である。

ＤＩ－ＣＩ－ＳＩＴＥ〔表８－４２〕は、ＳＮＴＢＬ（site-no table）である。POSITIONは、ＳＮＴＢＬ内の先頭からの位置を表わしている。

これらの３つのリレーションの生成については、表８－３７～表８－３９に示されている。

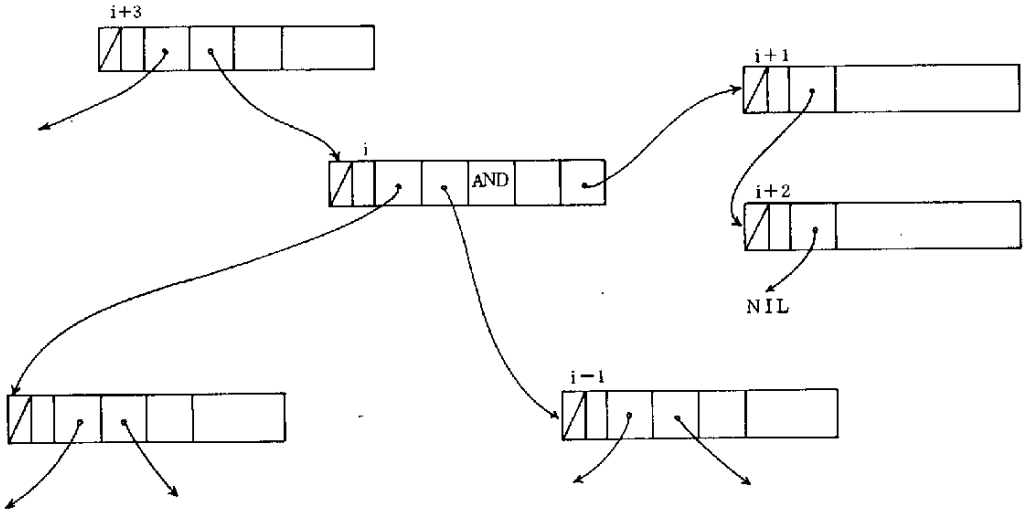


図8－99　AND／ROOTノードの番号づけ

表8－37　　DI－CI－GRELの生成

DI－CI－GRELの項目	転　送　内　容
*) GLOBAL－RELNAME	GRTBLのGRELNAME
*) ES－RS－TYPE	GRTBLのRTYPE
*) NO－OF－NODES	このtreeを構成する全ノードの数
NODE－NO	postorderに基づいて附けられたノードの番号
NODE	ノードの内容（3W＝12バイト）
TID	BLANK
FILLER	BLANK

表8－38　　DI－CI－RANGEの生成

DI－CI－RANGEの項目	転　送　内　容
*) VAR	RTBLのVAR
LOCAL－RELNAME	RTBLのRELNAME
LOCATION－ID	RTBLのSITE－NO
TID	BLANK
FILLER	BLANK

表８－３９　ＤＩ－ＣＩ－ＳＩＴＥの生成

ＤＩ－ＣＩ－ＳＩＴＥの項目	転　送　内　容
*) ＰＯＳＩＴＩＯＮ	ＳＮＴＢＬ内の位置（１、２、３、…）
ＬＯＣＡＴＩＯＮ－ＩＤ	ＳＮＴＢＬのＳＩＴＥ－ＮＯ

表８－４０　ＤＩ－ＣＩ－ＧＲＥＬリレーション

属　性　名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　明
ＧＬＯＢＡＬ－ＲＥＬＮＡＭＥ	１	Ｃ１	Ｃ	２０	ＧＣＳリレーション名
ＥＳ－ＲＳ－ＴＹＰＥ	２	Ｃ２	Ｃ	３	ＥＳＲ　ｏｒ　ＲＳＲ
ＮＯ－ＯＦ－ＮＯＤＥＳ	３	Ｃ３	Ｄ	４	このＤＤ－ｔｒｅｅの全ノード数
ＮＯＤＥ－ＮＯ	４	Ｎ	Ｄ	４	ノード番号
ＮＯＤＥ	５	Ｎ	Ｃ	１２	ノードの内容
ＴＩＤ	６	Ｎ	Ｃ	８	組識別子
ＦＩＬＬＥＲ	７	Ｎ	Ｃ	５	

（ length＝５６ ）

表８－４１　ＤＩ－ＣＩ－ＲＡＮＧＥリレーション

属　性　名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　明
ＶＡＲ	１	Ｋ	Ｃ	４	変数名
ＬＯＣＡＬ－ＲＥＬＮＡＭＥ	２	Ｎ	Ｃ	２０	ＬＣＳリレーション名
ＬＯＣＡＴＩＯＮ－ＩＤ	３	Ｎ	Ｄ	４	サイト番号
ＴＩＤ	４	Ｎ	Ｃ	８	
ＦＩＬＬＥＲ	５	Ｎ	Ｃ	８	

（ length＝４４ ）

表８－４２　ＤＩ－ＣＩ－ＳＩＴＥリレーション

属　性　名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　明
ＰＯＳＩＴＩＯＮ	１	Ｋ	Ｄ	２	bit-map（ＲＳＬ、ＲＳＲ）に対応するＳＮＴＢＬ内の位置
ＬＯＣＡＴＩＯＮ－ＩＤ	２	Ｎ	Ｄ	４	サイト番号
ＴＩＤ	３	Ｎ	Ｃ	８	
ＦＩＬＬＥＲ	４	Ｎ	Ｃ	８	

（ length＝２２ ）

8.3.8　DIPSの例

DIPSの例として次のような分散記述（DD）を考えてみよう。

```
RANGE OF   (A, B, D)
           (AUTHOR:3, BOOK:1, DOCUMENT:2) ;
DEFINE RSR   AUTHOR_REP
           (A#/NUMB=D4, ANAME/NAMES=C15, REP#=D4, REP_NAME=C30)
         (A#=A.AUTHOR#, ANAME=A.NAME, REP#=A.REPORT#*100 + 5,
          REP_NAME=B.NAME)
           WHERE   A.REPORT# / 50 = ABS (B.BOOK#-10) * 10
             AND   B.BOOK#   GT 5001 :
         (A#=A.AUTHOR#, ANAME=A.NAME, REP#=D.DOCUM#,
          REP_NAME=D.DOCUMNAME)
           WHERE   A.REPORT#   =   D.DOCUM# ;
```

RANGEPにより、RTBL（range table）とSNTBL（site-no table）が生成される。

*** RANGE TABLE (RTBL) DUMP ***

NO	VAR	RELNAME	SITE#
1	A	AUTHOR	3
2	B	BOOK	1
3	D	DOCUMENT	2

*** SITE_NO TABLE (SNTBL) DUMP ***

NO	SITE_#
1	3
2	1
3	2

DEFINEPにより、GRTBLとGATBLおよびDD－tree（図８－１００）が生成される。

*** GLOBAL RELATION TABLE (GRTBL) DUMP ***

NO	GRELNAME	RTYPE	DEGREE
1	AUTHOR_REP	RSR	4

*** GLOBAL ATTRIBUTE TABLE (GATBL) DUMP ***

ATTNO	GATTNAME	DNAME	FTYPE	FLENGTH
1	A#	NUMB	2	4
2	ANAME	NAMES	1	15
3	REP#	REP#	2	4
4	REP_NAME	REP_NAME	1	30

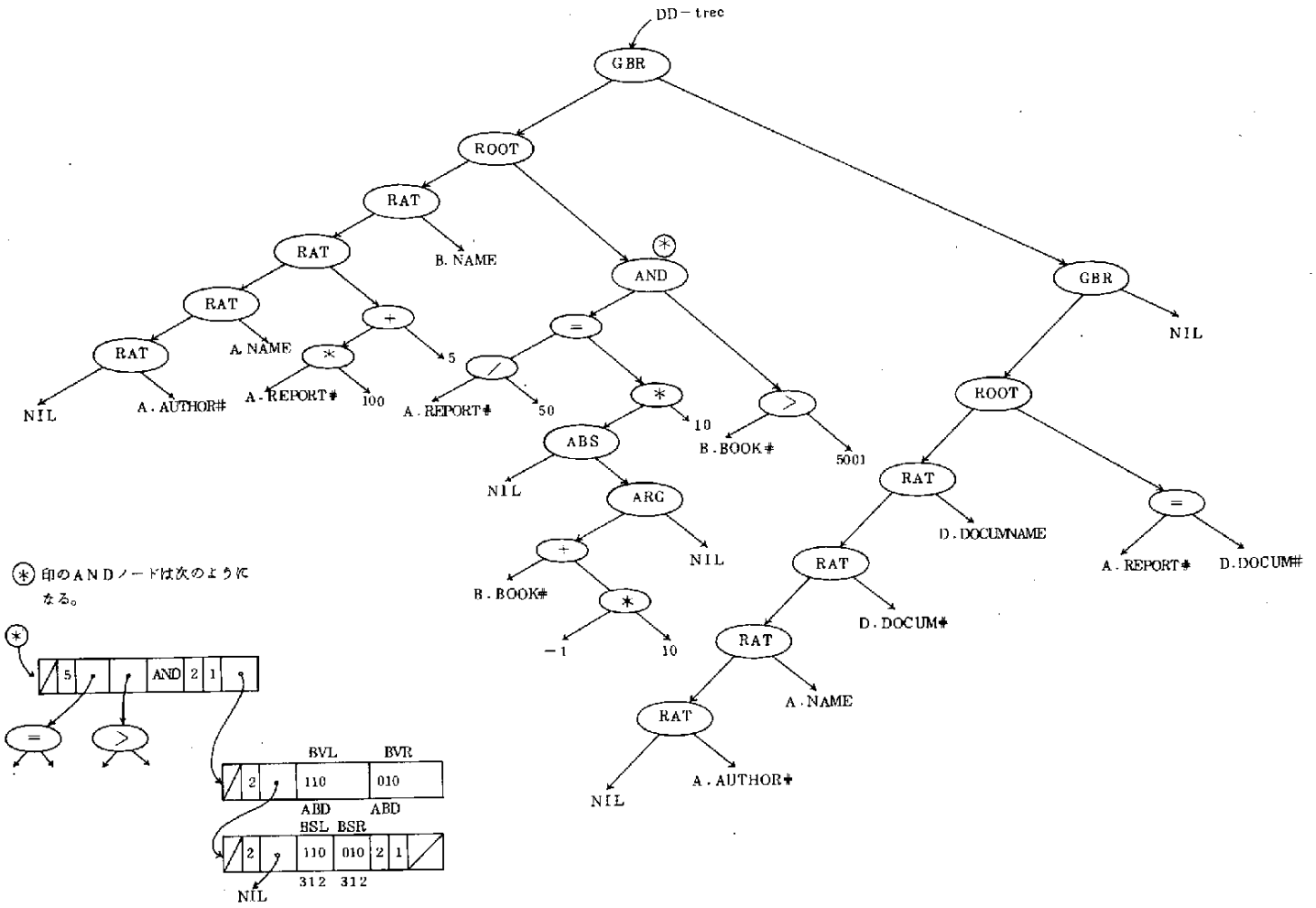


図８－１００　ＤＤ－tree

8.4 ＱＤＰ（Query Decomoosition Processor）

ＱＤＰは全体概念スキーマ（ＧＣＳ）に基づいたＧＣＳ問合せ（ＧＱ）を、各サイト単位又は、２つのサイト間のＬＣＳ問合せ（ＬＱ）へ分割するプロセスである［７］。このとき、ＱＤＰは、ＤＩＰＳで生成された分散情報（ＤＩ）を用いる。

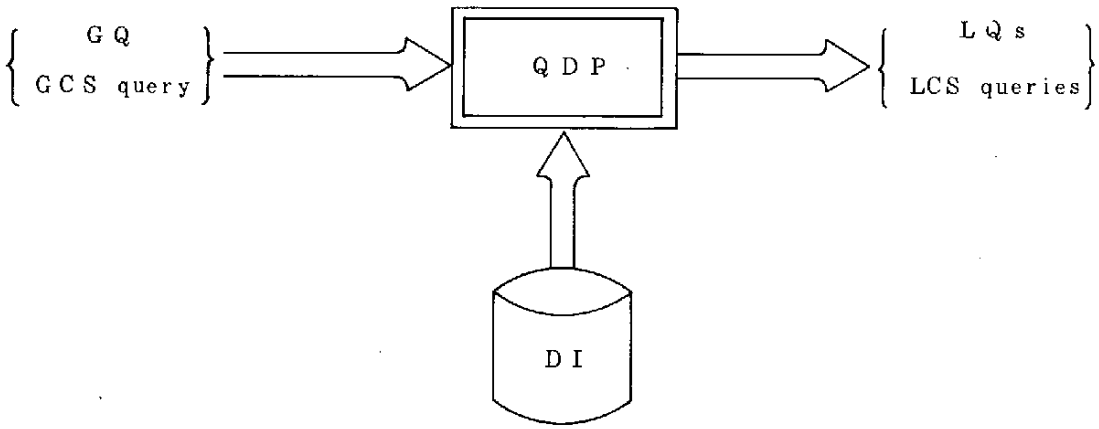


図８－１０１　ＱＤＰの概要

8.4.1　ＱＤＰの構成

ＱＤＰは、次の５つの主要なプロセスから成り立つ。

1）ＧＣＳ問合せの内部表現への変換
（ＧＣＳ　Ｑuery　ＴＲee　Generator　：　ＧＱＴＲＧ）

2）問合せの正規化　（Ｑuery　Normalizer　or　ＱＮ）

3）問合せの変形　（Ｑuery　Modifier or ＱＭ）

4）局所問合せ処理　（Local Query Processing or ＬＱＰ）

5）転送スケジューラ（Transmission Scheduler or ＴＳ）

但し、今回のインプリメンテーションでは、転送スケジューラ（ＴＳ）は含まれていない。

8.4.2　ＱＤＴＢＬ

ＱＤＴＢＬは、ＱＤにおいて使用されるテーブルが格納されている領域である。ＱＴＢＬは、次の２つの領域からなっている。

1）ＤＤＴＢＬ

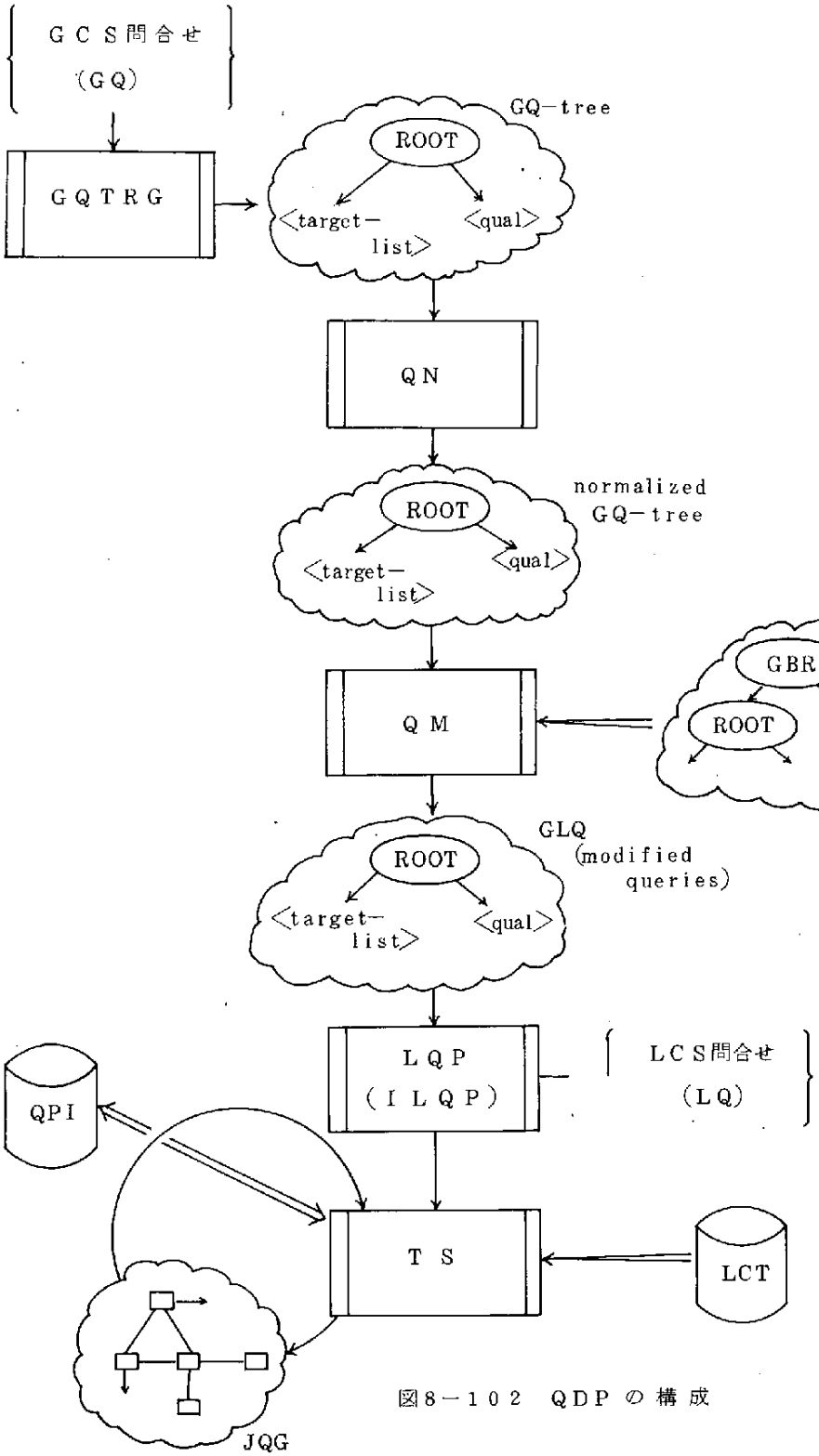


図８－１０２　ＱＤＰ の 構 成

2) ＧＱＴＢＬ

ＤＤＴＢＬは、分散記述に関するテーブル群である。これらは、ＤＩＰＳで生成されたＤＩ／ＣＩ、ＤＩ／ＳＩを主記憶上に読み込んで生成するものであるが、今回のインプリメンテーションでは、ＤＩＰＳシステムそのものをＱＤＰの内に組込み、テーブルおよびＤＤ－tree は、外部記憶を経由せずに、共通自由セル領域（ＦＮＳ）を用いて渡される。

また、ＧＱＴＢＬは、ＧＣＳ問合せに関するテーブル群である。これらのテーブルは、ＧＱＴＲＧで生成されるので、詳細はＧＱＴＲＧの項で述べる。

（例） ＤＤＴＢＬの例

ここでは、分散記述（ＤＤ）の例と、それからＤＤＴＢＬがどのように生成されるかを示す。各テーブルの構成については、表８－４３～表８－４７に示す。

1) 分散記述（ＤＤ）

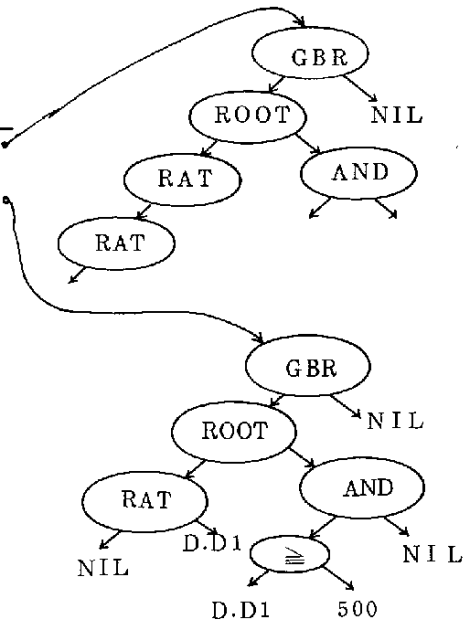
```
DI;
RANGE ( A, B     )  ( AAA:1, BBB:1              );
RANGE ( C, D     )  ( CCC:2, DDD:2              );
RANGE ( E        )  ( EEE:3                     );
RANGE ( F, G, H  )  ( FFF:4, GGG:4, HHH:4       );
DEFINE ESR  D1 ( D11, D12, D13       )
     ( D11 = B.B1 * 8, D12 = C.C1, D13 = F.F1 +2       )
     WHERE
               A.A1 = B.B1     AND  B.B1 = C.C2     AND  B.B2 = F.F2     AND
               A.A2 = "A"
     ;
DEFINE ESR  D2 ( D21, D22  )
     ( D21 = E.E2, D22 = H.H2 )
     WHERE     E.E1 = H.H2     AND  H.H3 = G.G2
     ;
DEFINE ESR  D3 ( D31 )
     ( D31 =  D.D1    )
     WHERE     D.D1 GE 500
     ;
```

2) ＣＩ／ＧＡＴＴ

SEQ	GRELNAME	GATTNO	GATTNAME	DOMAIN
1	D1	1	D11	D11
2	D1	2	D12	D12
3	D1	3	D13	D13
4	D2	1	D21	D21
5	D2	2	D22	D22
6	D3	1	D31	D31

3) C I／G R E L

SEQ	GRELNAME	RTYPE	DEGREE	DDPT
1	D1	ES	3	23
2	D2	ES	2	46
3	D3	ES	1	58



4) C I／R T B L

SEQ	LVAR	LRELNAME	SITENO
1	A	AAA	1
2	B	BBB	1
3	C	CCC	2
4	D	DDD	2
5	E	EEE	3
6	F	FFF	4
7	G	GGG	4
8	H	HHH	4

5) C I／S N T B L

SEQ	SITENO
1	1
2	2
3	3
4	4

6) CI／LATT

SEQ	LVAR	LATTNO	LATTNAME	GCON#	LATPT
1	B	1	B1	1	1
2	C	1	C1	1	4
3	F	1	F1	1	6
4	A	1	A1	1	9
5	C	2	C2	1	11
6	B	2	B2	1	13
7	F	2	F2	1	14
8	A	2	A2	1	16
9	E	2	E2	1	35
10	H	2	H2	1	37
11	E	1	E1	1	39
12	H	3	H3	1	41
13	G	2	G2	1	42
14	D	1	D1	1	54

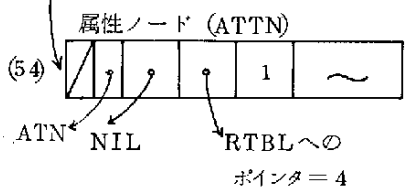


表8－43　CI／GATTの構成

属性名	属性番号	ROLE	TYPE	LENGTH	説明
GRELNAME	1	K	C	16	GCSリレーション名
GATTNO	2	K	D	2	GCS属性の属性番号
GATTNAME	3	N	C	16	GCS属性名
DOMAIN	4	N	C	16	領域（domain）名
TYPE	5	N	C	1	データ・タイプ C or D
LENGTH	6	N	D	2	バイト長

CI／GATTは、DDで定義されたGCSリレーション（GCS relation）の属性に関する情報が格納される。

表８－４４　　ＣＩ／ＧＲＥＬの構成

属　性　名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　　明
ＧＲＥＬＮＡＭＥ	1	Ｋ	Ｃ	１６	ＧＣＳリレーション名
ＲＴＹＰＥ	2	Ｋ	Ｃ	2	ＥＳ　ｏｒ　ＲＳ
ＤＥＧＲＥＥ	3	Ｎ	Ｄ	2	ＧＣＳ属性の数
ＤＤＰＴ	4	Ｎ	Ｄ	4	ＤＤ－ｔｒｅｅへのポインタ

ＣＩ／ＧＲＥＬは、ＤＤで定義されたＧＣＳリレーション（ＧＣＳ ｒｅｌａｔｉｏｎ）についての情報と、生成されたＤＤ－ｔｒｅｅへのポインタとの対応表となっている。

表８－４５　　ＣＩ／ＲＴＢＬの構成

属　性　名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　　明
ＬＶＡＲ	1	Ｋ	Ｃ	4	ＬＣＳ組変数名
ＬＲＥＬＮＡＭＥ	2	Ｎ	Ｃ	１６	ＬＣＳリレーション名
ＳＩＴＥＮＯ	3	Ｎ	Ｄ	2	所在サイト番号

ＣＩ／ＲＴＢＬは、ＤＤ内で用いられている組変数（ｔｕｐｌｅ ｖａｒｉａｂｌｅ）と、ＬＣＳリレーション及びその所在サイトとの対応表になっている。

表８－４６　　ＣＩ／ＳＮＴＢＬ

属　性　名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　　明
ＳＩＴＥＮＯ	1	Ｎ	Ｄ	2	所在サイト番号

ＣＩ／ＳＮＴＢＬは、ＤＤのＲＡＮＧＥ文により使用されたサイト番号を格納するテーブルである。

表８－４７　ＣＩ／ＬＡＴＴの構成

属　性　名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　　明
ＬＶＡＲ	1	Ｋ	Ｃ	4	ＬＣＳ組変数名
ＬＡＴＴＮＯ	2	Ｋ	Ｄ	2	ＬＣＳ属性の属性番号
ＬＡＴＴＮＡＭＥ	3	Ｎ	Ｃ	１６	ＬＣＳ属性名
ＧＣＯＮ＃	4	Ｋ	Ｄ	2	ＤＤ内の<sub-def>の番号
ＬＡＴＰＴ	5	Ｋ	Ｄ	2	<sub-def>内で使われるこの属性に対応するＡＴＴＮへのポインタ

ＣＩ／ＬＡＴＴは、１つの<sub-def>毎に、そこで使われているＬＣＳ属性と、対応するＡＴＴＮ（属性ノード）へのポインタとを格納している。１つのＤＤ－ｔｒｅｅ内で、ある組変数と属性の対に対しては、ユニークな属性ノード（ＡＴＴＮ）が生成される。

８．４．３　ＧＱＴＲＧ

ＧＱＴＲＧは、ＧＣＳ問合せ（ＧＱ）の文字列を入力として、ＧＱ－ｔｒｅｅ及び、ＧＱＲＡＴＴ、ＧＱＲＲＥＬ、ＧＱＲＴＢＬ、ＧＱＧＡＴＴの４つのテーブルを生成する〔図８－１０３〕。

ＧＱは、ＱＵＥＬを用いて記述される。ＱＵＥＬの形式的定義は、付記１を参照されたい。また木（ｔｒｅｅ）の生成については、ＱＴＧ〔８.２.３〕を参照されたい。

次に、ＧＱＲＲＥＬ、ＧＱＲＡＴＴ、ＧＱＲＴＢＬ、ＧＱＲＡＴＴの４つのテーブルについて説明する〔表８－４８～表８－５１〕。

表８－４８　ＧＱＲＲＥＬ

属　性　名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　　明
ＧＱＲＥＬＮＡＭＥ	1	Ｋ	Ｃ	１６	結果リレーション名
ＤＥＧＲＥＥ	2	Ｎ	Ｄ	2	結果属性の数
ＧＱＰＴ	3	Ｎ	Ｄ	2	ＧＱ－ｔｒｅｅへのポインタ

ＧＱＲＲＥＬは、ＧＱによって生成される結果リレーション（ｒｅｓｕｌｔ－ｒｅｌａｔｉｏｎ）名、及びＧＱ－ｔｒｅｅへのポインタが格納される。１つのＧＱに対して、ＧＱＲＲＥＬの組（ｔｕｐｌｅ）が１つ生成される。また、このＧＱの処理が終ると、ＧＱＲＲＥＬは次のＧＱのために使われる。

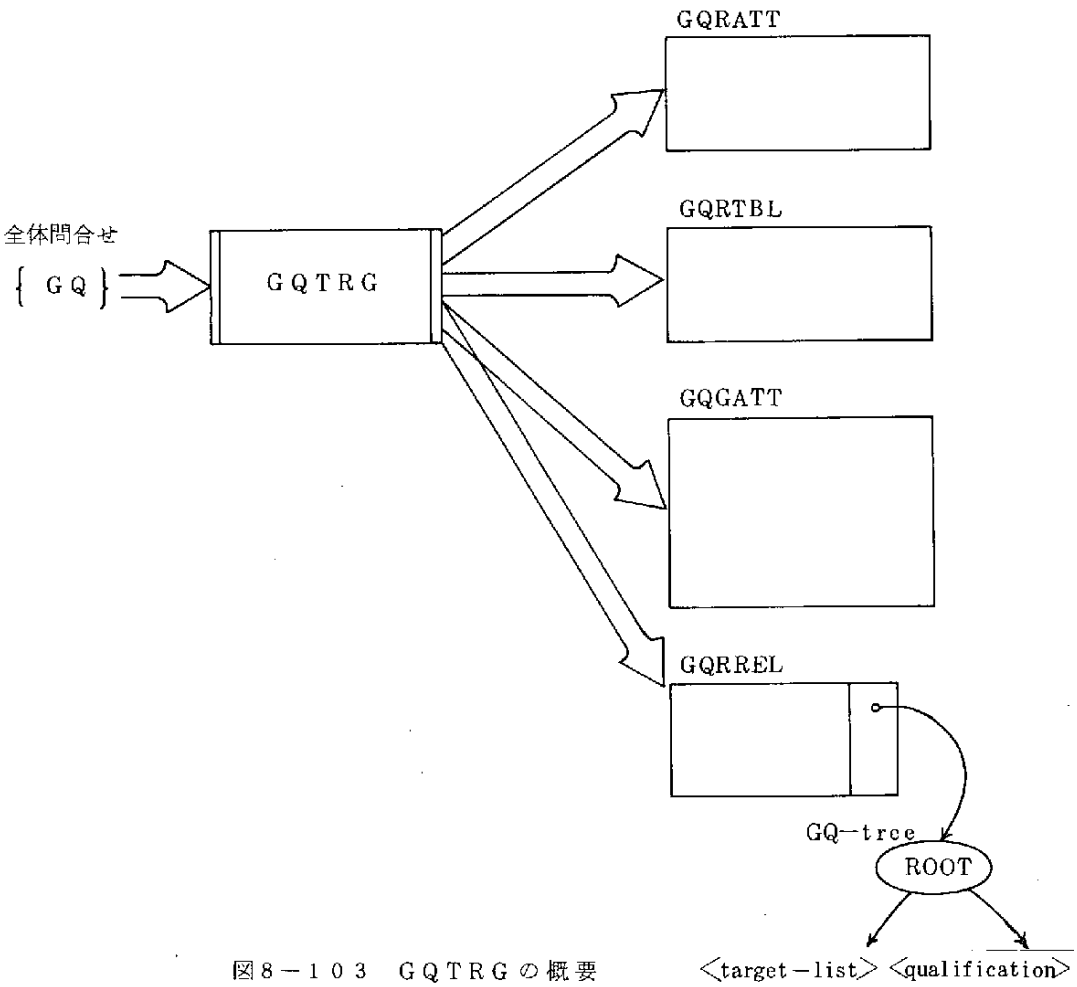


図８－１０３　ＧＱＴＲＧの概要

表８－４９　　ＧＱＲＡＴＴの構成

属　性　名	属性番号	ROLE	TYPE	LENGTH	説　　明
GQRELNAME	1	K	C	16	結果リレーション名
GQATTNO	2	K	D	2	結果属性番号
GQATTNAME	3	N.	C	16	結果属性名

ＧＱＲＡＴＴは、ＧＣＳ問合せ（ＧＱ）内に現われる結果属性（result−attribute）を格納する。このＧＱの処理が終ると、ＧＱＲＡＴＴは、次のＧＱのために使われる。

表８－５０　ＧＱＲＴＢＬ

属　性　名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＨ	説　　明
ＧＶＡＲ	1	Ｋ	Ｃ	4	ＧＣＳリレーション（ＧＲ）に対応する組変数
ＧＲＥＬＮＡＭＥ	2	Ｎ	Ｃ	１６	ＧＣＳリレーション（GR）名
ＤＤＰＴ	3	Ｎ	Ｄ	2	ＧＲを定義しているＤＤ－treeへのポインタ
ＭＡＲＫ	4	Ｎ	Ｄ	2	作業用
ＧＧＰＴ	5	Ｎ	Ｄ	2	ＱＭでＤＤ－treeへのポインタとして使用される。

ＧＱＲＴＢＬは、ＧＣＳ変数とＧＣＳリレーションとの対応表である。更に、ＧＣＳリレーション（ＧＲ）を定義しているＤＤ－treeへのポインタも持つ。

表８－５１　ＧＱＧＡＴＴの構成

属　性　名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　　明
ＧＶＡＲ	1	Ｋ	Ｃ	4	ＧＣＳ変数名
ＧＡＴＴＮ	2	Ｋ	Ｄ	2	ＧＣＳ属性番号
ＧＡＴＴＮＡＭＥ	3	Ｎ	Ｃ	１６	ＧＣＳ属性名
ＧＣＯＮ#	4	Ｋ	Ｄ	2	ＧＶＡＲのＤＤ内でのsub−defの番号
ＧＡＴＰＴ	5	Ｎ	Ｄ	2	ＧＣＳ属性に対応するＡＴＴＮへのポインタ
ＭＡＲＫ	6	Ｎ	Ｄ	2	作業用（ＱＭで使用）

ＧＱＧＡＴＴは、あるＧＣＳ変数ＧＶＡＲに対応したＤＤ－tree内に現われるＧＣＳ属性名（ＧＣＳattribute）と、これに対応したノード（ＡＴＴＮ）へのポインタとの対応表となっている。

8.4.4　QN

QNは、GQTRGで生成されたGQ－treeを入力として、このtree内の＜qualification＞treeを正規化する（QNP）。但し、GQの入力は、積正規形（conjunctive normal form）であるとする。

＜qualification＞::= ＜clause＞{{AND ＜clause＞ }}

＜clause ＞::=＜predicate＞〔{ OR ＜predicate＞ }〕

ここで＜predicate ＞は、比較述語（例えば、x.a＝5 ）を表わす。

また、正規化された木（tree）のROOTNとANDNに変数についてのbit-mapを付加し（VPERP）、更に、全ての変数を連結にする結合節（ join clause ）があることもチェックする（CC）。

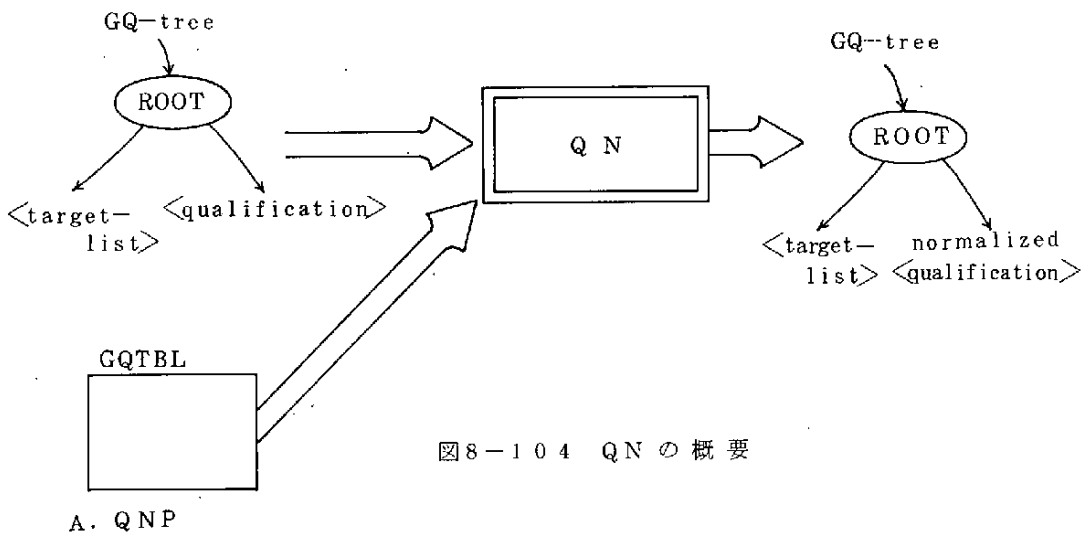
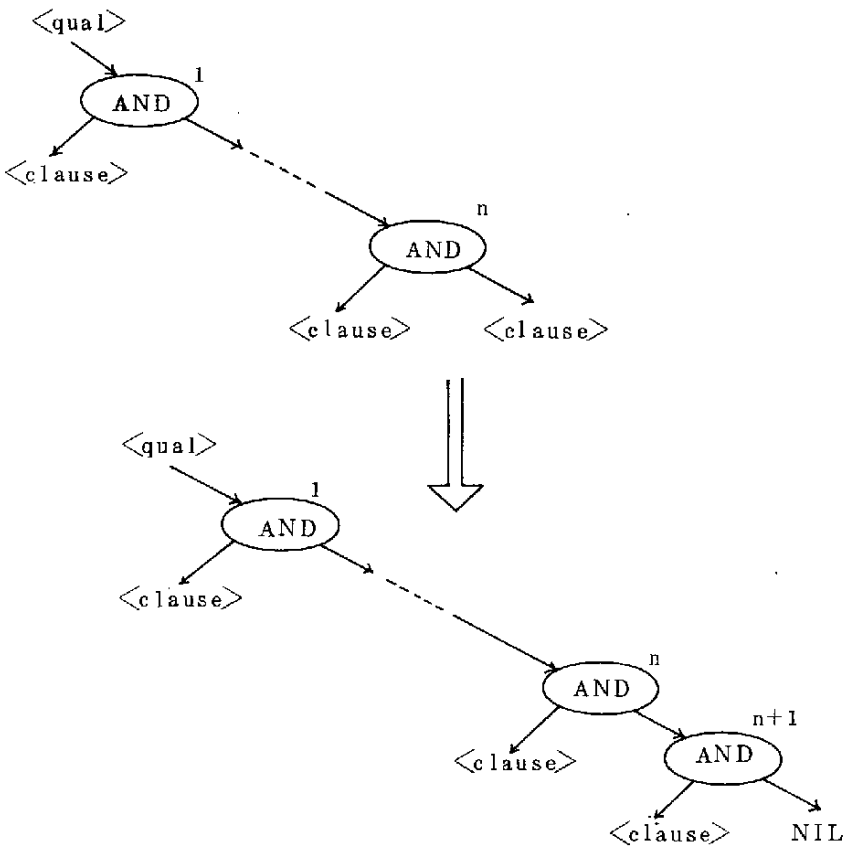


図8－104　QN の 概 要

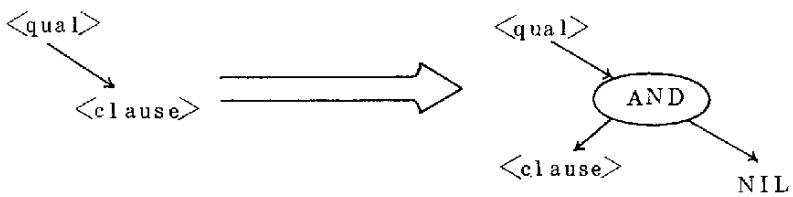
A. QNP

QNPは、次のような木（ tree ）の変換のみを行なう。

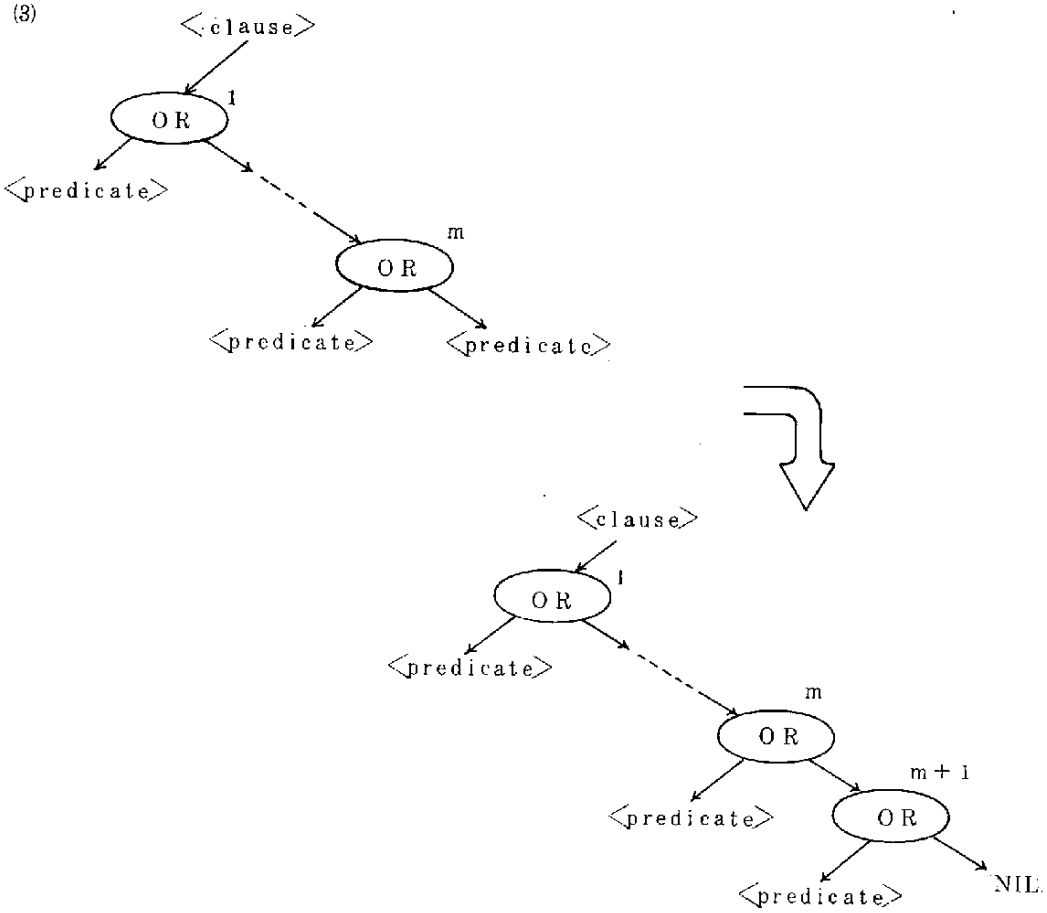
(1)


<qual>
AND
1
<clause>
n
AND
<clause>
<clause>
<qual>
AND
1
<clause>
n
AND
n+1
AND
<clause>
<clause>
NIL

(2)


<qual>
<clause>
<qual>
AND
<clause>
NIL

(3)

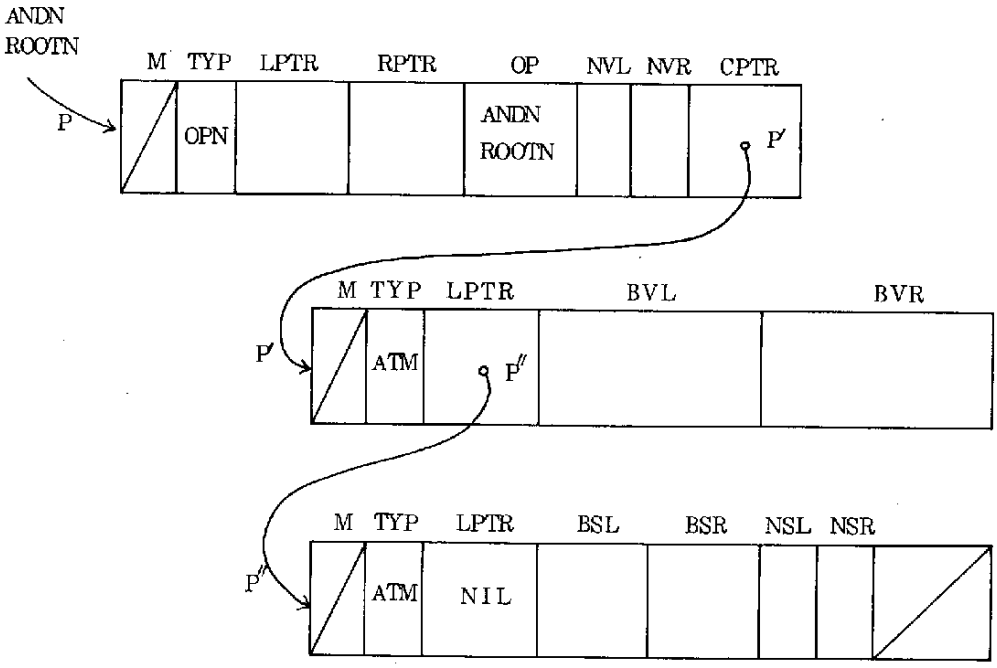


これ以上の正規化の能力を現在のＱＮＰは持っていない。ユーザは、ほとんど正規化された論理式を入力せねばならない。

Ｂ．ＶＰＥＲＰ

ＶＰＥＲＰは、正規化されたＧＱ－ｔｒｅｅ のＡＮＤノード、およびＲＯＯＴノードに、変数（ｖａｒｉａｂｌｅ）についてのｂｉｔ-ｍａｐをセットする。変数は、ＡＴＴＮ（属性ノード）内のＲＰＴＲに格納されているＧＱＲＴＢＬへのポインタを用いて求める。

また、ＲＯＯＴＮは、＜ｔａｒｇｅｔ-ｌｉｓｔ＞ｔｒｅｅと＜ｑｕａｌｉｆｉｃａｔｉｏｎ＞ｔｒｅｅ とのルートを表わし、ＡＮＤＮは、ＡＮＤ演算子を表わしている。これらのノードは、この他に、そのノードの部分木（ｓｕｂｔｒｅｅ）内の変数を保有しているため、ノード内のＣＰＴＲフィールドを用いて多変長ノードとなっている。ｂｉｔ-ｍａｐ（ＢＶＬとＢＶＲ）は、ＧＱＲＴＢＬと対応している。即ち、各ビットの位置はＧＱＲＴＢＬ内の各変数の位置に対応しており、ビットがＯＮであることは、対応する木（ｔｒｅｅ）内にこの変数が存在することを示している。



フィールド名	説　　明
NVL〔P〕	このノードの左手の部分木内の変数の数　（≦32）
NVR〔P〕	このノードの右手の部分木内の変数の数　（≦32）
BVL〔P'〕	このノードの左手の部分木内の変数についてのbit-map（32ビット）
BVR〔P'〕	このノードの右手の部分木内の変数についてのbit-map（32ビット）
BSL〔P''〕	このノードの左手の部分木内のサイトについてのbit-map(16ビット)
BSR〔P''〕	このノードの右手の部分木内のサイトについてのbit-map(16ビット)
NSL〔P''〕	このノードの左手の部分木内のサイト数（≦16）
NSR〔P''〕	このノードの右手の部分木内のサイト数（≦16）

図8－105　　ROOTNとANDNの構造

図8－106の木内で㊀印のついたANDノードにチェインされるbit-mapは次のようになる。

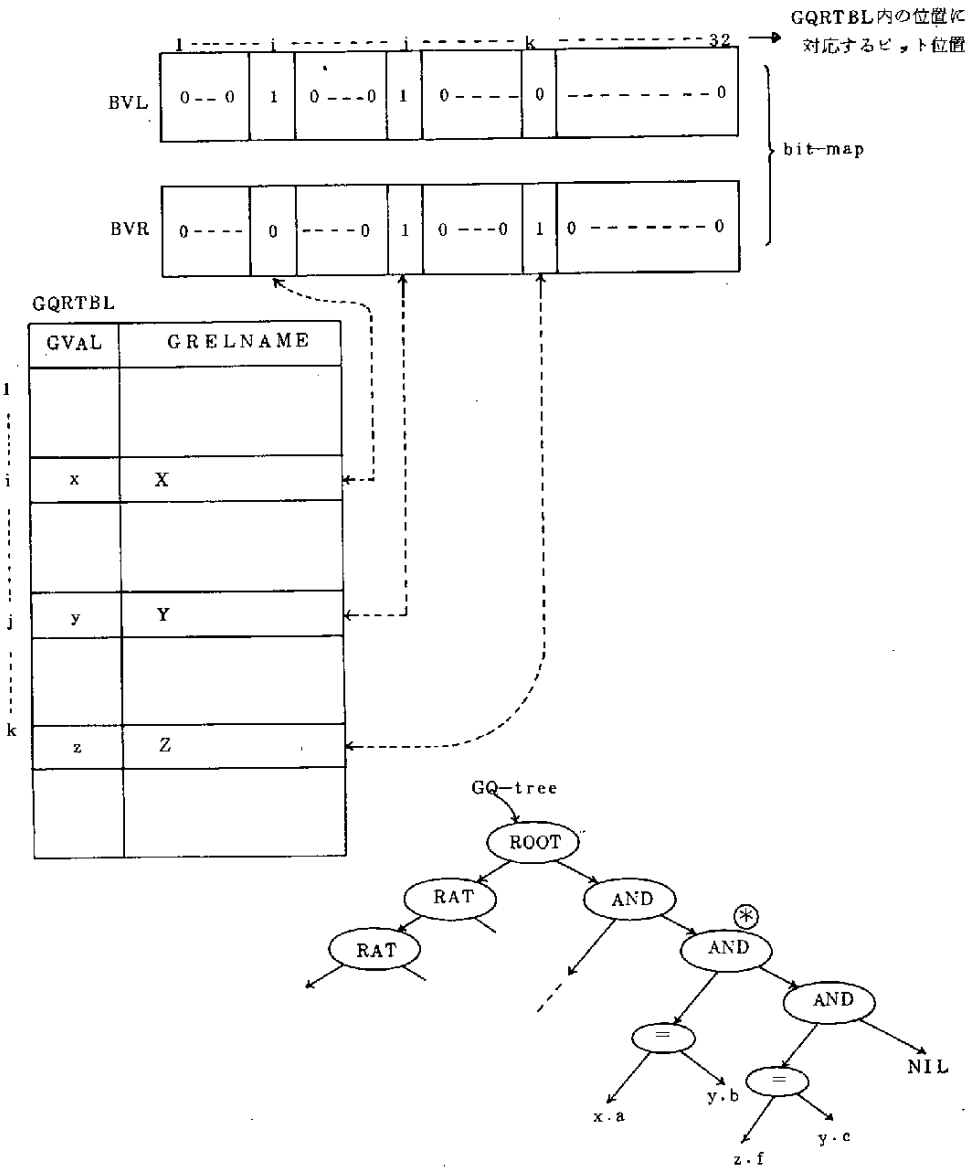


図８－１０６　bit－mapとＧＱＲＴＢＬとの対応

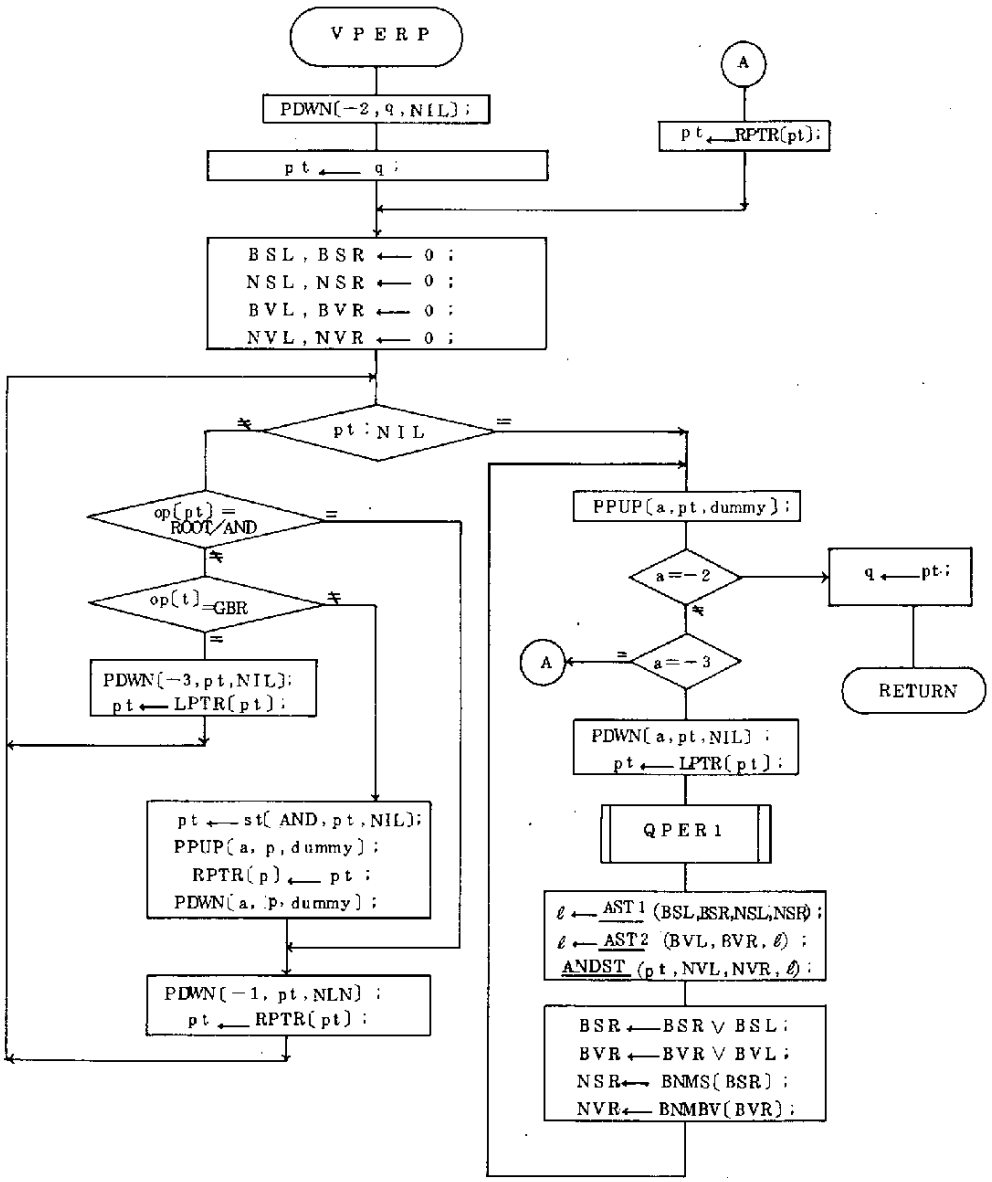


図8－107　VPEPRの処理概要

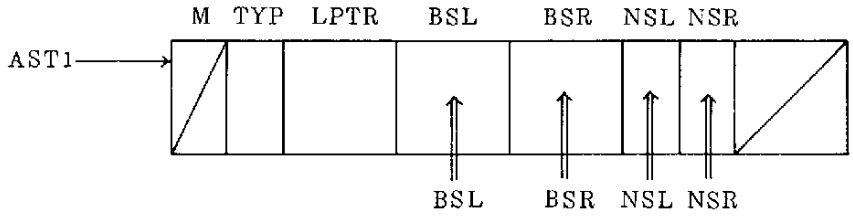
ＶＰＥＲＰで用いられているサブルーチンの概要を次に述べる。

1) ＡＳＴ１（ＢＳＬ、ＢＳＲ、ＮＳＬ、ＮＳＲ）；

自由ノード・リストより１ノードをget する。

このノードをＡＮＤＮ（ＲＯＯＴＮ）にchain される２番目のＡＴＭＮとしてＢＳＬ、ＢＳＲ、ＮＳＬ、ＮＳＲをset する。

このノードのpointer をＡＳＴ１としてreturn する。

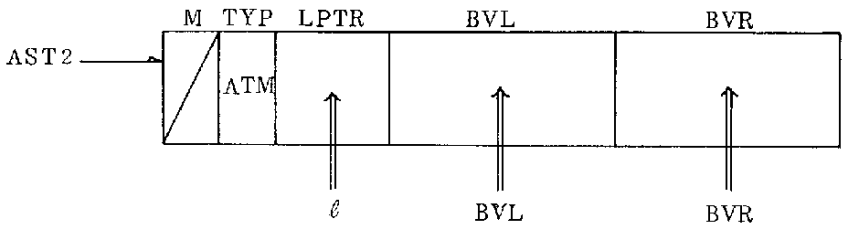


2) ＡＳＴ２（ＢＶＬ、ＢＶＲ、ℓ）；

自由ノード・リストより１ノードをget する。

このノードをＡＮＤＮ（ＲＯＯＴＮ）にchain されているℓ番目のＡＴＭＮとしてＢＶＬ、ＢＶＲをset する。ℓは２番目のＡＴＭＮへのpointer である。

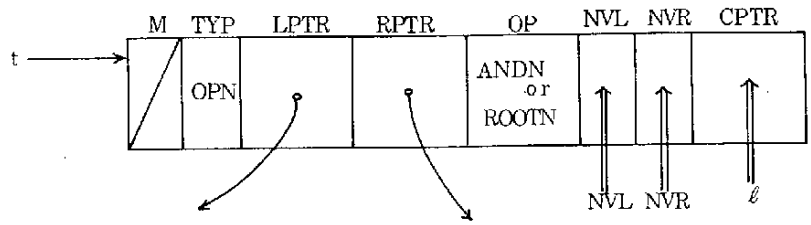
このノードへのpointer をＡＳＴ２としてreturn する。



3) ＡＮＤＳＴ（ｔ、ＮＶＬ、ＮＶＲ、ℓ）；

ｔはＡＮＤＮ（ＲＯＯＴＮ）へのpointer である。

このノードにＮＶＬ、ＮＶＲをset し、ＣＰＴＲにＡＴＭＮへのpointer ℓをset する。



4) BNMBS（BSR）；

bit-map BSR内のONであるbit数をカウントする。（16ビット）

5) BNMBV（BVR）；

bit-map BVR内のONであるbit数をカウントする。（32ビット）

6) QPER1（t、BSL、BVL、NSL、NVL）；

ANDN（又はROOTN）の左手のsubtreeへのpointer（t）を入力とし、このsubtreeに関するbit-mapを生成する。return時のtはANDN（又はROOTN）を指している。

C. CC（connectivity checker）

CCの機能は次の2つである。

1) qualification内の各述部（predicate）が参照する全ての変数を連結（connective）にするjoin clauseが存在することを確かめる。

2) target-listが参照する全ての変数が、qualification内に含まれることをチェックする。ここで、問合せがqualificationを持たないとき、target-listは、唯一つの変数のみを参照していなければならない。

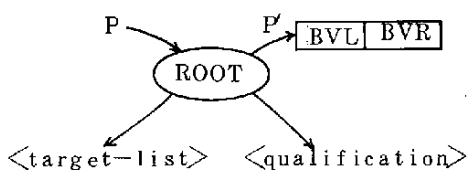
次に、CCの処理概要を述べる〔図8－108〕。

1) VBM〔表8－52〕の生成

GQ－treeを、ROOTノードから右端のANDノードまでサーチして、対応するbit-map（BVL）を順にVBM〔表8－52〕にセットする。また、VBMの2番目の組（tuple）以降のtupleからVBLについて冗長のものを除去する。<qualification>が無い場合（VBMが1 tupleしかない場合）は、VBMの第1組（tuple）のVBLでONとなっているビットが唯一つであるかどうかをチェックする。

2) DFAVBM（depth-first search）

ROOTノードにチェインされているBVLとBVRのORは、GQ－treeが参照する全ての変数（variable）についてのbit-mapとなる。これを変数allbmに格納する。即ち、allbm←BVL（P'）∨ BVR（P'）



VBMの第2 tuple以下のVBLを用いて、縦型サーチ（depth-first search）によって連結性のチェックを行なう。このとき、VBMは、変数をノードとし、join clauseをリンクとするグラフとみなすことができる。全てのノードがNEWとマークされているとする。ここでは、allbmのビットがON

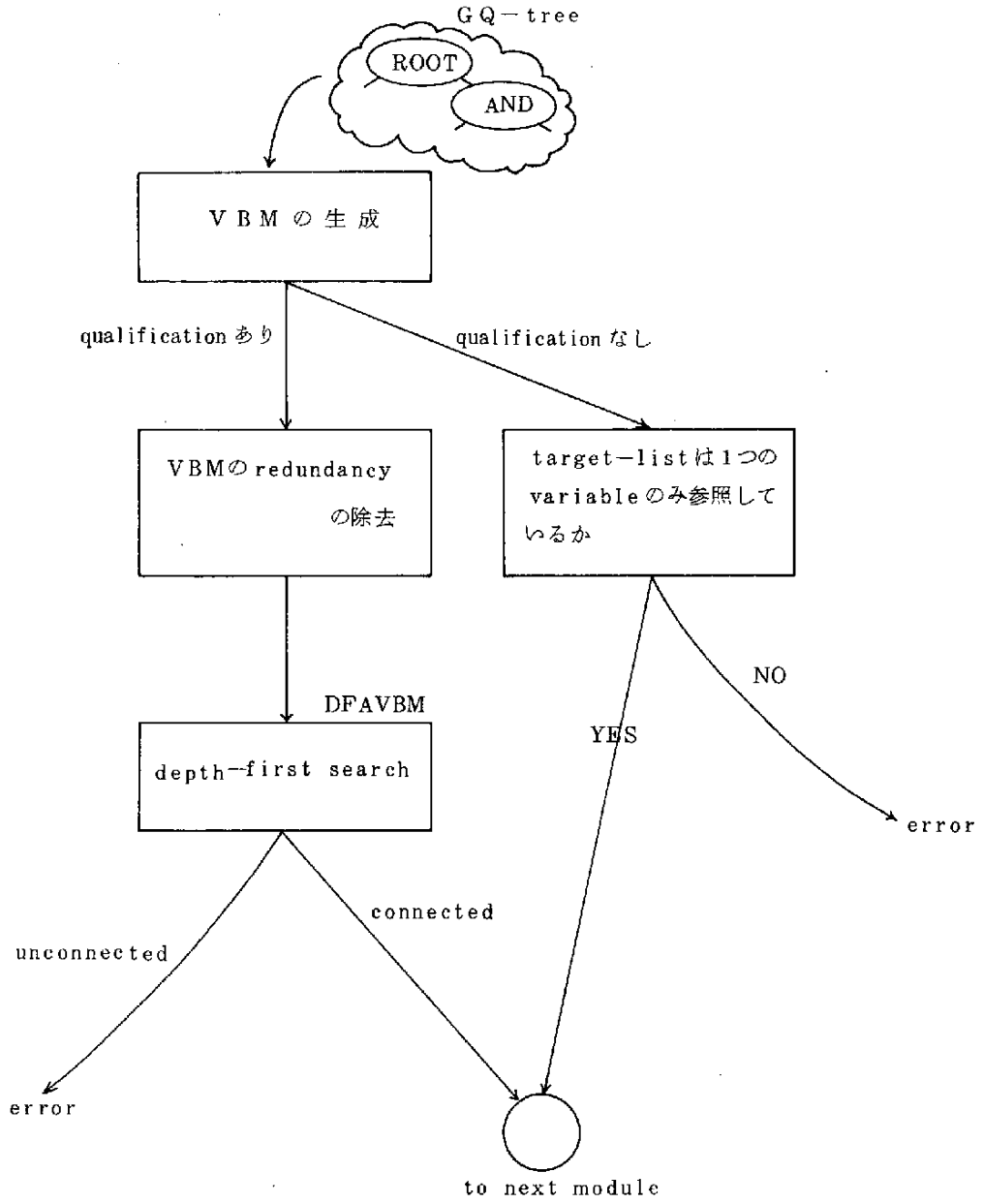


図８－１０８　ＣＣの処理概要

になっていることである。まず、あるノードを決め、これをＯＬＤとマークする。このノードの隣接ノードのうち、ＮＥＷであるものをさがす。即ち、ＶＢＭのＶＢＬで、allbm のビット位置と同一のビットがＯＮとなっている組（ tuple ）をさがし、ＯＬＤとマークする。そのＶＢＬが制限（他にＯＮのビットがない）のときは、次の tuple をさがす。join clause のときは、他のＯＮとなっているビットで、再びＶＢＭをサーチする。もし、ＮＥＷが一つもなければ、親のノードに戻って、再びＮＥＷである隣接ノードをさがす。こうして、最初の親ノード（ＲＯＯＴノード）まで戻ったときに、ＮＥＷであるノードがあれば、即ち、allbm にＯＮ のビットがあれば、このグラフは、連結ではなかったことになる。このとき、次のようなメッセージを出力する。

```
*  ERROR  DFAVBM  UNCONNECTED  VARIABLE  EXIST
      VAR      RELNAME
      ××××     ××××××××××××××××××
      ××××     ××××××××××××××××××

          allbm でＯＮのビットに対応した変数名とリレーション名
```

表８－５２　ＶＢＭ

属　性　名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　　明
ＰＴＲ	1	Ｎ	Ｄ	2	部分木（subtree）へのポインタ
ＶＢＬ	2	Ｎ	Ｄ	4	部分木の変数についての bit-map
ＭＡＲＫ	3	Ｎ	Ｄ	2	作業用、初めはＯＦＦ

８．４．５　ＱＭ

ＱＭは、ＤＤＡＴＣＨ、ＱＭＰ、ＳＰＲＭＡＴＣＨという３つのサブプロセスから成る。

ＤＤＡＴＣＨは、ＤＩ／ＣＩを用いて、ＧＱＲＴＢＬのＤＤＰＴ、ＧＧＰＴ属性のセット、及び、ＬＡＴＴ、ＬＲＴＢＬの生成を行なう。

ＱＭＰは、ＤＤ－tree を用いてＧＱ－tree のＧＣＳリレーションとＧＣＳ属性を対応するＬＣＳリレーションとＬＣＳ属性におきかえ、global ＬＣＳ　query（ＧＬＱ）tree に変形する。

ＳＰＲＭＡＴＣＨは、ＧＱＬ内のＲＯＯＴノード、ＡＮＤノードに、変数および所在サイト番号に

ついてのbit-mapをセットする。

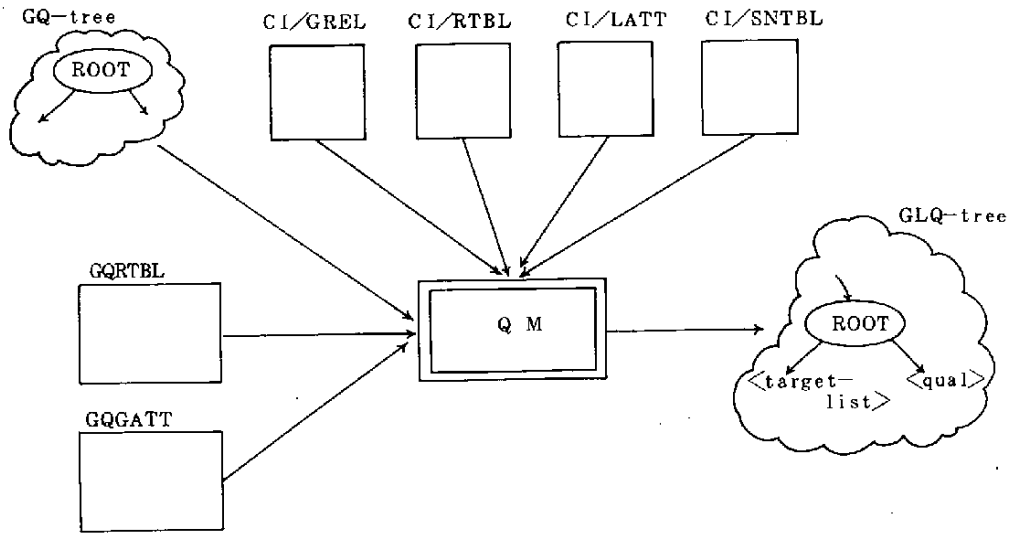


図8－109　QMの概要

A．DDATCH

DDATCHは、LRTBL、LATTの生成及び、GQRTBLのDDPT、GGPT属性のセットを行なう。このとき、DD－treeは、自由ノード空間内の他の位置にコピーされる。また、1つのリレーションに対し、2つ以上の変数が宣言されている場合は、それぞれの変数に対してDD－treeがコピーされる。それらの木（tree）のポインタは、GQRTBLのDDPTにセットされる〔下図斜線の部分〕。

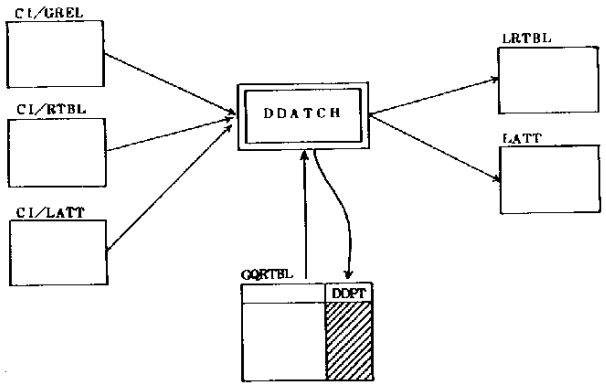
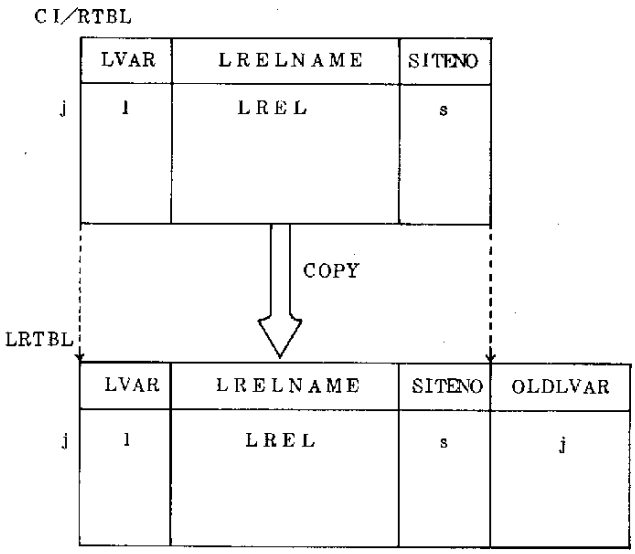


図8－110　DDATCHの概要

a) ＤＤＡＴＣＨの処理概要

1) 〔ＣＩ／ＲＴＢＬをＬＲＴＢＬ、ＣＩ／ＬＡＴＴをＬＡＴＴへコピー〕

ＣＩ／ＲＴＢＬをＬＲＴＢＬへ、ＣＩ／ＬＡＴＴをＬＡＴＴへコピーする。ＯＬＤＬＶＡＲ属性には、対応するＣＩ／ＲＴＢＬの組（tuple）番号を入れる〔図８－１１１〕。



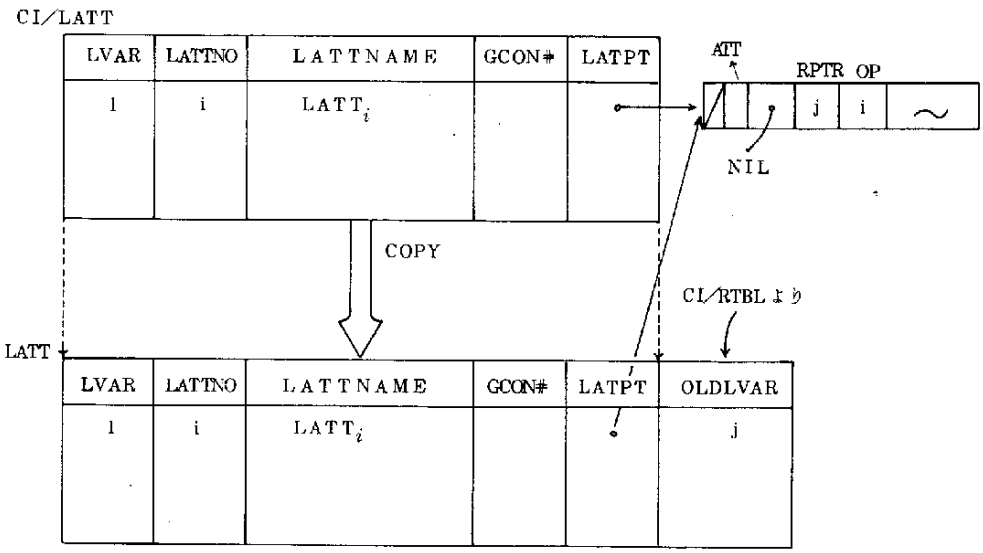


図８－１１１　ＣＩ／ＲＴＢＬとＣＩ／ＬＡＴＴのコピー

2) ＧＱＲＴＢＬの先頭の組（ tuple）を取り出す。

ｉ ← ＭＩＮＧＱＲＴＢＬ（＝１）

（ＧＱＲＴＢＬには、ＧＱの参照するＧＣＳリレーションが格納されている。）

3) この組（ tuple)のＧＲＥＬＮＡＭＥを取り出し（ gname とする）、ＭＡＲＫをＯＮにする。

4) 〔ＤＤ－tree のコピー〕

ＣＩ／ＧＲＥＬをサーチして、ＣＩ／ＧＲＥＬ．ＧＲＥＬＮＡＭＥ（ｊ）＝gname なる組（tuple）を見つけ、ＣＩ／ＧＲＥＬ．ＤＤＰＴ（ｊ）のさすＤＤ－tree をコピーして、そのポインタをＧＱＲＴＢＬのＤＤＰＴ（ｉ）とＧＧＰＴ（ｉ）にセットする。このとき、ＣＯＰＹルーチン〔 9.4.5.A , b)〕を使用する。ここでは、演算子ノード、定数ノードはコピーされるが、属性ノードはコピーされない。

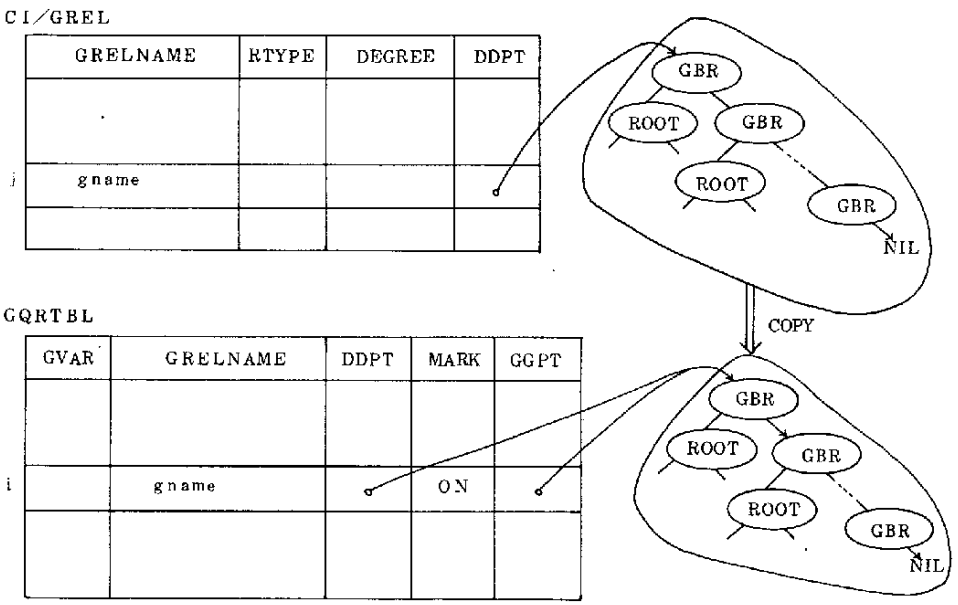


図８－１１２　ＤＤ－tree のコピー

5) 〔同一のＧＣＳリレーション名があるかどうかを調べる〕

ＧＱＲＴＢＬをｉ＋１番目からＣＧＱＲＴＢＬまでサーチして、ＧＲＥＬＮＡＭＥがgname のものがあるかどうかを調べる。

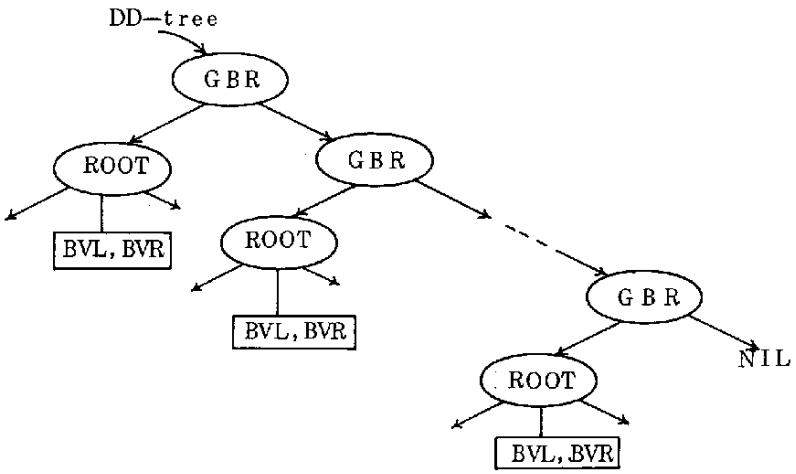
6) なければ、ｉに続く組（ tuple)の内で、ＭＡＲＫがＯＦＦのものをみつけ、3)へ戻る。

7) 〔１つのＧＣＳリレーションが複数の変数をもつ時〕

あれば、同一のgnameをもつＧＱＲＴＢＬがある。即ち、１つのリレーション名に対して２つ以上の変数が宣言されていることになる。この組（tuple)番号をｋとする。また、このtupleのＭＡＲＫをＯＮにする。

8）〔ＤＤ－treeの参照する変数の明確化〕

ＣＩ／ＧＲＥＬ．ＤＤＰＴ（ｊ）のさすＤＤ－treeをサーチして、この木（tree）が参照するＬＣＳ変数を明らかにする。即ち、全てのＲＯＯＴノードにチェインされているＢＶＬとＢＶＲの論理和をとる。



これらのＬＣＳ変数をＬＲＴＢＬのＣＬＲＴＢＬ＋１番目から順に格納する。（ＣＬＲＴＢＬは、ＬＲＴＢＬの使用中のbottomを示している。）但し、ＬＲＴＢＬのＯＬＤＬＶＡＲには、このＬＣＳ変数が格納されているＣＩ／ＲＴＢＬの組（tuple）番号をセットし、ＬＶＡＲには、ＬＲＴＢＬ内でユニークな名前を附して格納する。

ここでは、ＬＶＡＲの頭に¥nを附けて、名前が重複しないようにしている。nはＬＶＡＲ毎につけられた通し番号である。

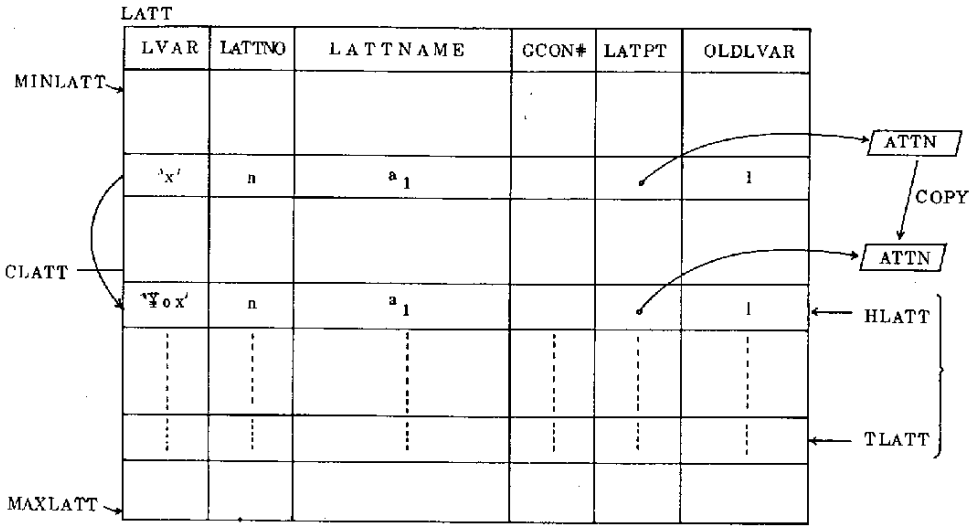
9）〔ＬＣＳ属性ノードのコピー〕

ＬＡＴＴをアクセスして、8）で見つかったＬＣＳ変数をもつ属性（attribute）を見つけ、ＬＶＡＲに対応する新しいＬＣＳ変数をセットして、ＬＡＴＴのＣＬＡＴＴ＋１番目から格納する。（ＣＬＡＴＴは、ＬＡＴＴの使用されているbottomを表わしている。）更に、対応する属性ノード（ＡＴＴＮ）をコピーし、そのポインタをＬＡＴＰＴにセットする。

以上で、ＬＡＴＴのＨＬＡＴＴからＴＬＡＴＴまでの間に、新しい変数をつけた属性が格納されることになる。

10）ＣＩ／ＧＲＥＬ・ＤＤＰＴ（ｊ）のさすＤＤ－treeをコピーして、このコピーされたＤＤ－treeのポインタをＧＱＲＴＢＬのＤＤＰＴ（ｋ）とＧＧＰＴ（ｋ）にセットする。ここでも4）と同じＣＯＰＹルーチンを用いる。

11）ＧＱＲＴＢをｋ＋１番目以降もサーチして、ＧＲＥＬＮＡＭＥがgnameのものを見つける。



もしあれば、7）へ。なければ、6）へ戻る。

12）ＧＱＲＴＢのＭＡＲＫがＯＦＦのものが無くなったら処理を終る。

表８－５３　　ＬＲＴＢＬ

属　性　名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　　明
ＬＶＡＲ	1	Ｋ	Ｃ	4	ＬＣＳ変数名
ＬＲＥＬＮＡＭＥ	2	Ｎ	Ｃ	１６	ＬＣＳリレーション名
ＳＩＴＥＮＯ	3	Ｎ	Ｄ	2	サイト番号
ＯＬＤＬＶＡＲ	4	Ｎ	Ｄ	2	ＣＩ／ＲＴＢＬへのポインタ

表８－５４　　ＬＡＴＴ

属　性　名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　　明
ＬＶＡＲ	1	Ｋ	Ｃ	4	ＬＣＳ変数名
ＬＡＴＴＮＯ	2	Ｋ	Ｄ	2	ＬＣＳ属性番号
ＬＡＴＴＮＡＭＥ	3	Ｎ	Ｃ	１６	ＬＣＳ属性名
ＧＣＯＮ#	4	Ｋ	Ｄ	2	sub-def の番号
ＬＡＴＰＴ	5	Ｎ	Ｄ	2	sub-def 内で用いられるこのＬＣＳ属性に対応するＡＴＴＮへのポインタ
ＯＬＤＬＶＡＲ	6	Ｎ	Ｄ	2	改名される前のＬＣＳ変数の格納されているＣＩ／ＲＴＢＬへのポインタ

b) ＣＯＰＹルーチンの概要

呼出し手順は、

p ← ＣＯＰＹ（dd、head−latt、tail−latt、table−id）

である。dd はコピーされる木（ tree ）へのポインタを表わし、ＣＯＰＹルーチンは、dd をコピーして新たな tree を生成し、その新しい tree のポインタを値として返すサブルーチンである。

table−id は、木（ tree）内の属性ノード（ＡＴＴＮ）についての情報が格納されているテーブルのＩＤ−番号である。table−id としては、

- ＃ＬＡＴＴ　（＝１）　（ＤＤＡＴＣＨで使用）
- ＃ＧＱＧＡＴＴ（＝２）　（ＱＭで使用）

という、２つの値をとり得る。

head−latt と tail−latt は、table−id に対応するテーブルをサーチするときの、テーブルの上限と下限を定めている。head−latt ＞ tail−latt の場合は、木（ tree）内のＡＴＴＮは全てコピーされ、新たにテーブルの tail−latt＋１番目から格納される。

また、ＣＯＰＹルーチンは、次のような４つのサブルーチンをもっている。

- ＣＯＰＹ
  - ＣＯＰＹ−ＡＴＴ
    - ＣＯＰＹ−ＬＡＴＴ
    - ＣＯＰＹ−ＧＱＧＡＴＴ
  - ＣＯＰＹ−ＣＮＴＮ

即ち、ＣＯＰＹルーチンは、木（ tree ）を帰りがけ順（ postorder ）になぞりながら、ノードをコピーし、各ノードのポインタを更新してゆく。次に、ＣＯＰＹ−ＡＴＴについてのみ説明する。

p ← ＣＯＰＹ−ＡＴＴＮ〔 dd、head−latt、tail−latt、table−id 〕；

(1) table−id ＝＃ＬＡＴＴ（＝１）のとき

dd は、属性ノード（ATTN)へのポインタである。ＡＴＴＮ内のＣＩ／ＲＴＢＬへのポインタ（ＲＰＴＲ（dd））と属性番号（ＯＰ(dd)）を用いてＬＡＴＴをhead−latt から tail−latt までサーチして、ＬＡＴＴ内の対応するノードをみつけ結果とする。図８−１１３は、ある木（.tree)内の属性ｘ.ａのノードのコピーを示している。このとき、変数ｘは新たにｙとされる。

(2) table−id ＝＃ＧＱＧＡＴＴ（＝２）のとき

dd は属性ノード（ＡＴＴＮ）へのポインタである。ＡＴＴＮ内のＧＱＲＴＢＬへのポインタ（ＲＰＴＲ（dd））を用いて変数名（ＧＱＲＴＢＬのＧＶＡＲ）と属性番号（ＯＰ(dd)）をキーとしてＧＱＧＡＴＴをhead−latt からtail−latt までサーチする。ＧＱＧＡＴＴ内にあれば、そのＧＡＴＰＴ（ＡＴＴＮへのポインタ）を結果とする。もし、見つからないときは、属性ノードをコピーし、そのポインタをＧＱＧＡＴＴの tail−latt＋１番目に格納する。ＧＡＴ

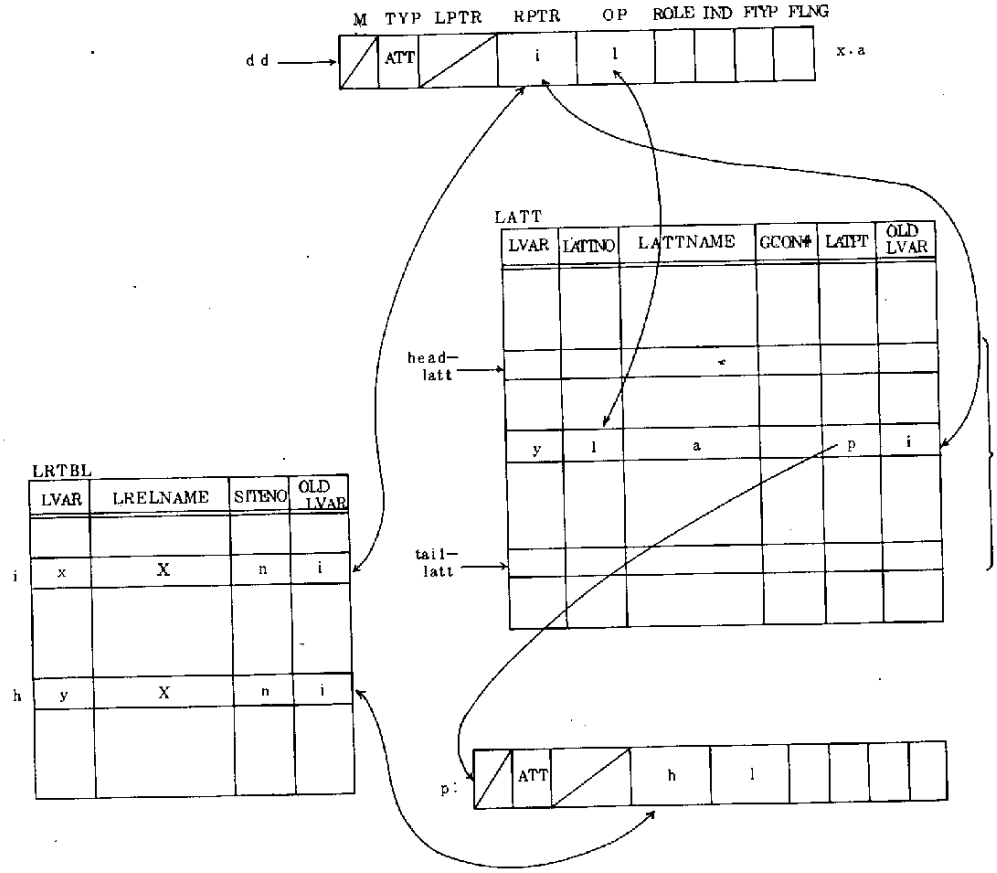


図８－１１３　ＬＣＳ属性ｘ．ａのコピー

ＰＴ以外のフィールドは、ＧＱＧＡＴＴの先頭からhead-latt-1番目の中からみつけ、そのフィールドの内容をコピーする。ＧＣＳ属性ｇ.Ｇ１のコピーの様子を図８－１１４に示す。

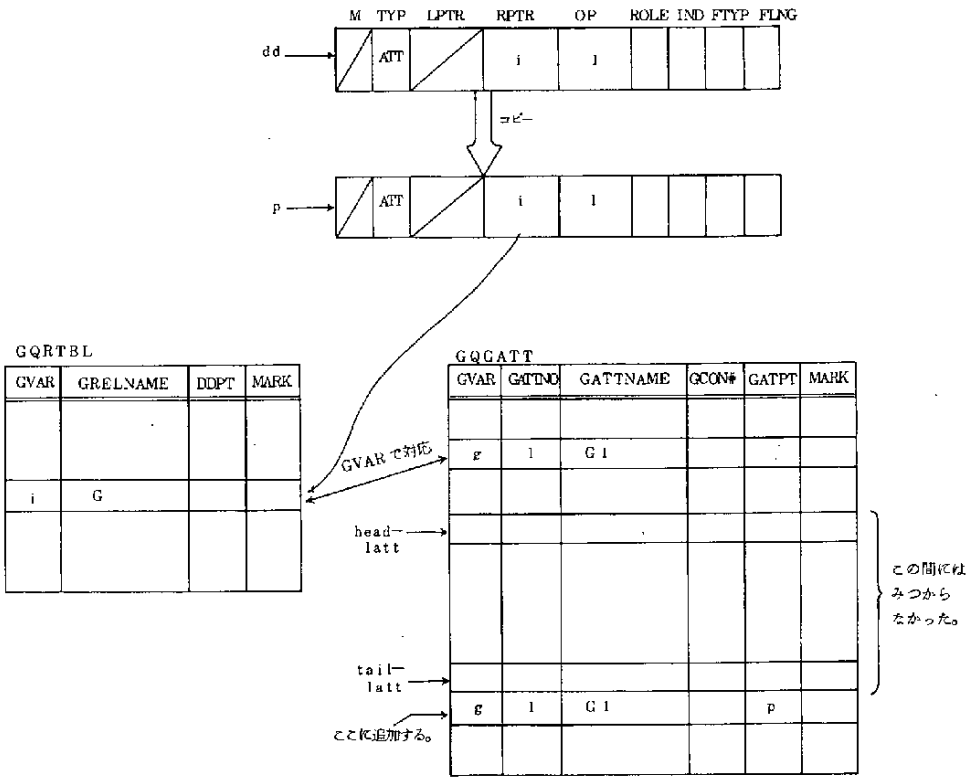


図８－１１４　ＧＣＳ属性のコピー

Ｂ．ＱＭＰ

ＱＭＰは、ＧＣＳリレーションとＧＣＳ属性を参照するＧＣＳ問合せ（ＧＱ－ｔｒｅｅ）を対応するＬＣＳリレーションとＬＣＳ属性を参照する問合せ（ｇｌｏｂａｌ ＬＣＳ　ｑｕｅｒｙ ｏｒ ＧＬＱ）に変形するプロセスである。ＤＤ－ｔｒｅｅ に＜ｓｕｂ-ｄｅｆ＞が複数あるときは、その組合せの数だけの木が生成され、ＬＱＰに渡される。

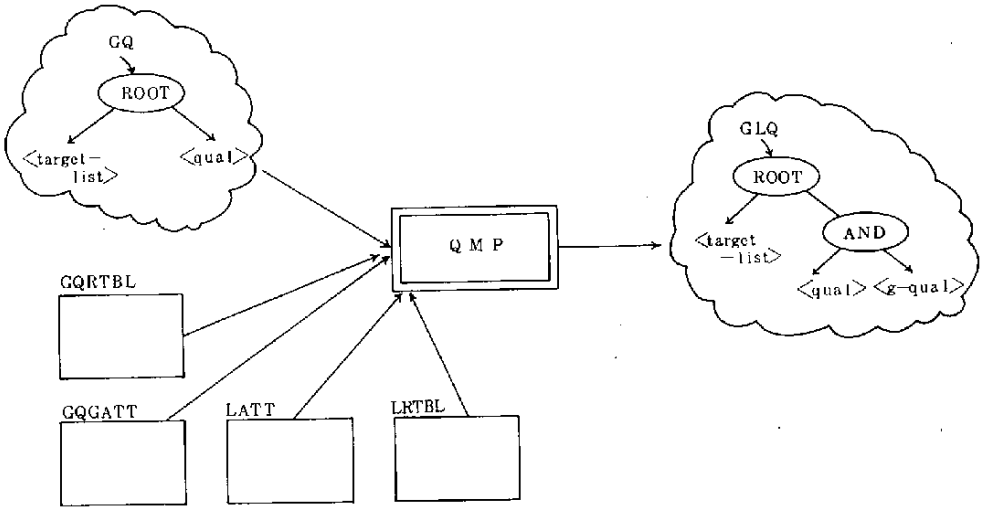


図８－１１５　ＱＭＰの概要

a）ＱＭＰの処理概要

1）〔ＧＱＲＴＢＬの全組を未使用とマーク〕

ＧＱＲＴＢＬの全ての組（tuple）をＯＦＦとマークする。

2）〔ＧＱ－tree で用いられるＧＣＳ変数を求める〕

gq をＧＣＳ問合せ（ＧＱ－tree）へのポインタとするとき、gq が参照するＧＣＳ変数を求める。これを、（g1、g2、……、gn）とする。これはＧＱ－tree のＲＯＯＴノードの bit-map（ＢＶＬとＢＶＲ）から求まる。これをallbm とする。allbm ← ＢＶＬ ∨ ＢＶＲ。

3）〔使用されている変数に対応するＧＱＲＴＢＬを使用中とマーク〕

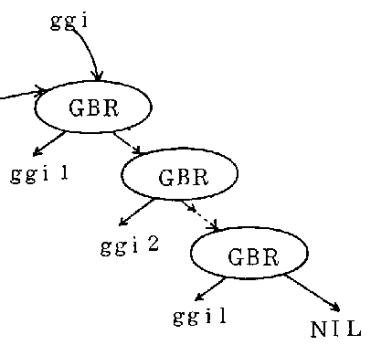
このbit-map（allbm）を用いて、allbm でＯＮとなっているビットに対応する（即ち、ＧＱ－tree内で参照されている）ＧＱＲＴＢＬの組（tuple）をＯＮとマークする。

4）〔全ての＜sub-def＞の組合せをとり出す〕

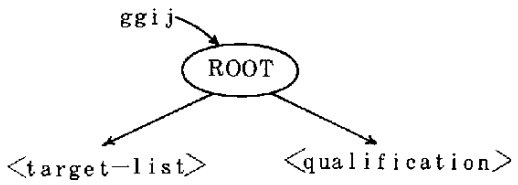
ＧＣＳ変数gi（i＝1、2、……、n）に対して、次のようなＤＤ－tree(ggi)が対応している。

GQRTBL

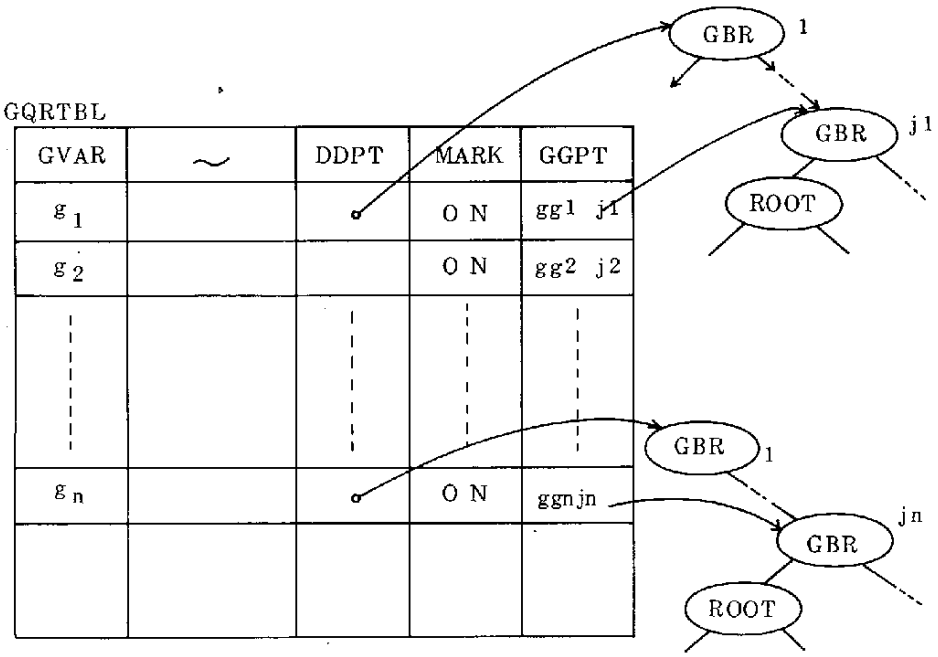
GVAR	～	DDPT	MARK	GGPT
g_i			ON	



ここで、ggijは、DD文内のj番目の＜sub-def＞に対応した木（tree）で、次のような構造をしている。



GQRTBLのマークがONとなっている組（tuple）のDD－treeのそれぞれから、1つずつの＜sub-def＞をとり出し（GGPTにポインタをセット）、その全ての組合せについて以降の処理を行なう。即ち、＜sub-def＞を要素とするGQRTBLのtupleの直積をとり、その直積の要素を、G1、G2、……、Gm と名附ける。G1（1＝1、2、……、m）を順にとり出す。



5）〔GQ－treeのコピー〕

G1 について、以下のことを行なう。GCS問合せ（GQ－tree）をコピーし、これをgg1とする。このコピーは、COPYルーチンによって行なわれ、その結果、GQ－tree内の属性ノード（ATTN）も全てコピーされ、gg1 内のATTNはGQGATTテーブル の

chead からctailまで順に格納されていることになる。

6）〔ＧＱＧＡＴＴテーブルから１組をとり出す〕

ＧＱＧＡＴＴを、chead からctail までシーケンシャルにアクセスする。

7）各ＧＱＧＡＴＴの組（ tuple ）についてＧＶＡＲを取り出す。即ち、ＧＡＴＰＴにリンクされているＡＴＴＮのＲＰＴＲから、ＧＱＲＴＢＬの組（ tuple ）番号（ ｒ ）がわかる。このＧＶＡＲに対応するＤＤ－tree 内の＜sub-def ＞に対応するggijをとり出す。

ggij ← LPTR（ＧＱＲＴＢＬ・ＧＧＰＴ（ｒ））；

→左手のポインタを表わす。

8）〔ＧＣＳ属性をＬＣＳ属性を参照する式でおきかえる〕

ggijの＜target-list＞をアクセスして、このＧＣＳ属性（ＧＣＳattribute ）に対応するＬＣＳ論理式をみつける。これは、ＧＱＧＡＴＴ・ＧＡＴＰＴ（ｈ）のさすＡＴＴＮ内のＯＰ（属性番号）に対応したポインタをみつけることである。これを、npt とする。次にＧＣＳ属性に対応するＡＴＴＮをＬＣＳ論理式をコピーしたものでおきかえる。即ち、ポインタｐのノードの内容を、nptのノードの内容で置き換える。ここで、nptは一般に＜a-exp＞であるが、ＡＴＴＮならば、ＬＯＣＡＴテーブル〔表８－５５〕に格納しておく。

9）〔 ggijの＜qualification＞部をコピーして接合する〕

全てのggij の＜qualification＞を、ＧＣＳ問合せgql の＜qualification ＞の最後にコピーしながら接合する。このときＬＣＳ属性ノード（ＡＴＴＮ）がＬＯＣＡＴテーブルにあれば、そのＡＴＴＮはコピーしないでテーブル内のポインタを使用する。これは、8）でＧＣＳ属性をＬＣＳ属性で置き換えたときに、すでにＡＴＴＮはコピーされたと考えられるからである。また、こうしないと、ＡＴＴＮはユニークでなくなる。ＬＯＣＡＴにないときは、ＬＯＣＡＴに追加しておく。

10）こうして、ＧＱＧＡＴＴのchead からctail まで処理したら、その時のgql をパラメータとして、ＳＰＲＭＡＴＣＨ、ＱＩＣ、ＬＱＰ、ＴＳをＣＡＬＬする。（現在、ＱＩＣとＴＳはインプリメントされていない。）

11）全ての組合せＧｌについて、5）～ 10）の処理を繰返す。

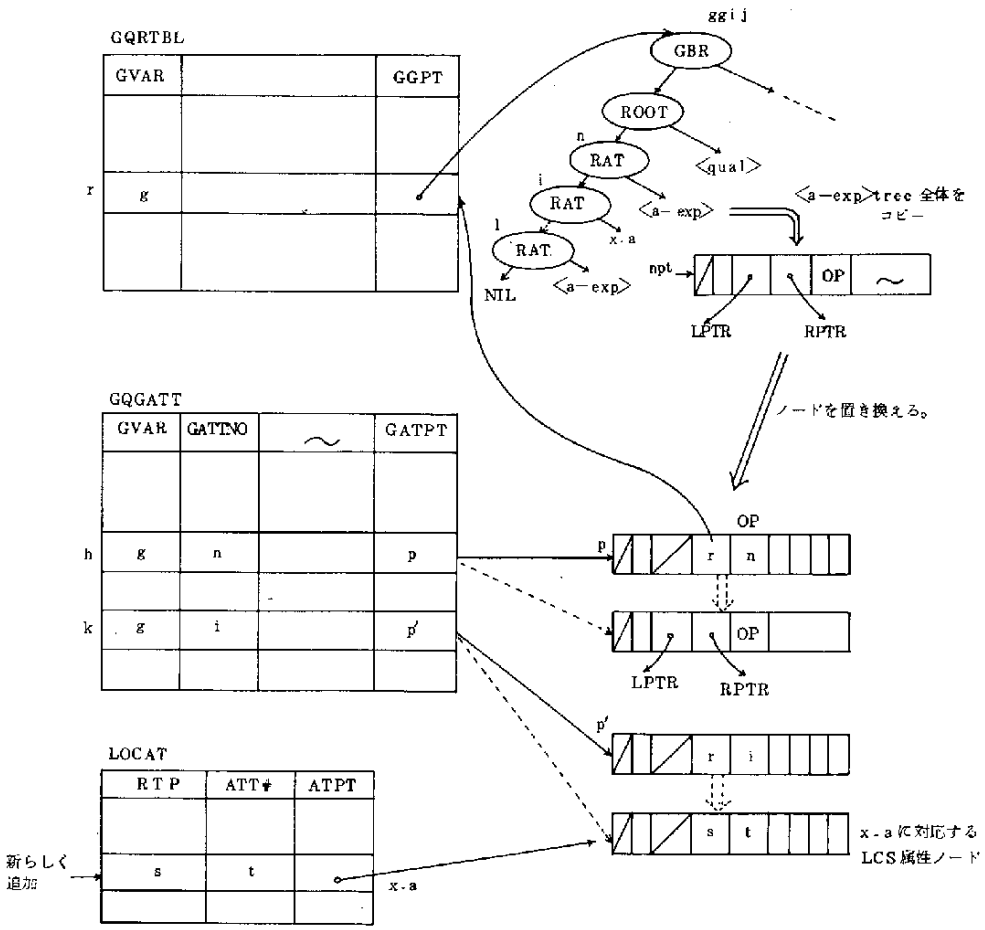
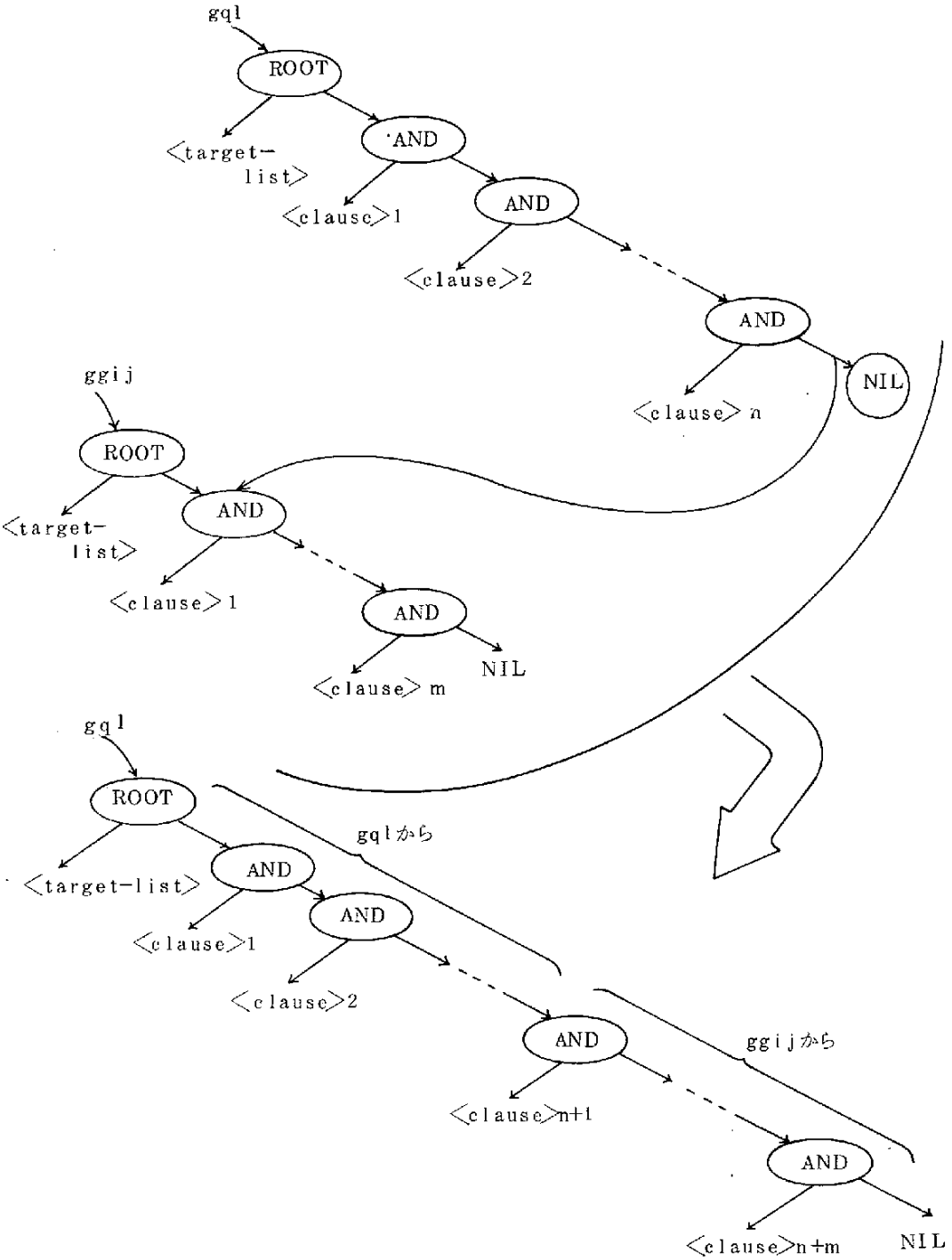


表８－５５　　ＬＯＣＡＴの構成

属　性名	属性番号	ＲＯＬＥ	ＴＹＰＥ	ＬＥＮＧＴＨ	説　　明
ＲＴＰ	1	Ｋ	Ｄ	2	ＬＲＴＢＬへのポインタ ＡＴＴＮのＲＰＴＲに対応する
ＡＴＴ＃	2	Ｋ	Ｄ	2	属性番号。ＡＴＴＮのＯＰに 対応する
ＡＴＰＴ	3	Ｎ	Ｄ	2	ＡＴＴＮへのポインタ


gql
ROOT
<target-list>
AND
<clause>1
AND
<clause>2
AND
<clause> n
NIL
ggij
ROOT
<target-list>
AND
<clause>1
AND
<clause> m
NIL
gql
ROOT
<target-list>
AND
<clause>1
AND
<clause>2
AND
<clause>n+1
AND
<clause>n+m
NIL
gqlから
ggijから

C．SPRMATCH

SPRMATCHは、GLQ内のROOTノード、及び、ANDノードに、変数と所在サイトについてのbit-mapをセットする。

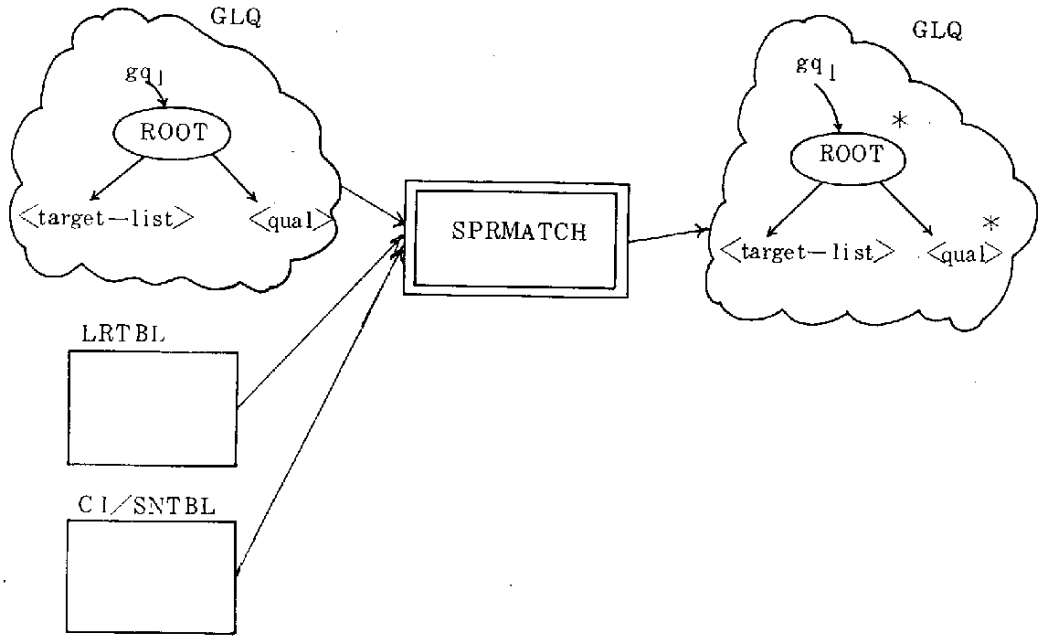


図8－116　SPRMATCHの概要

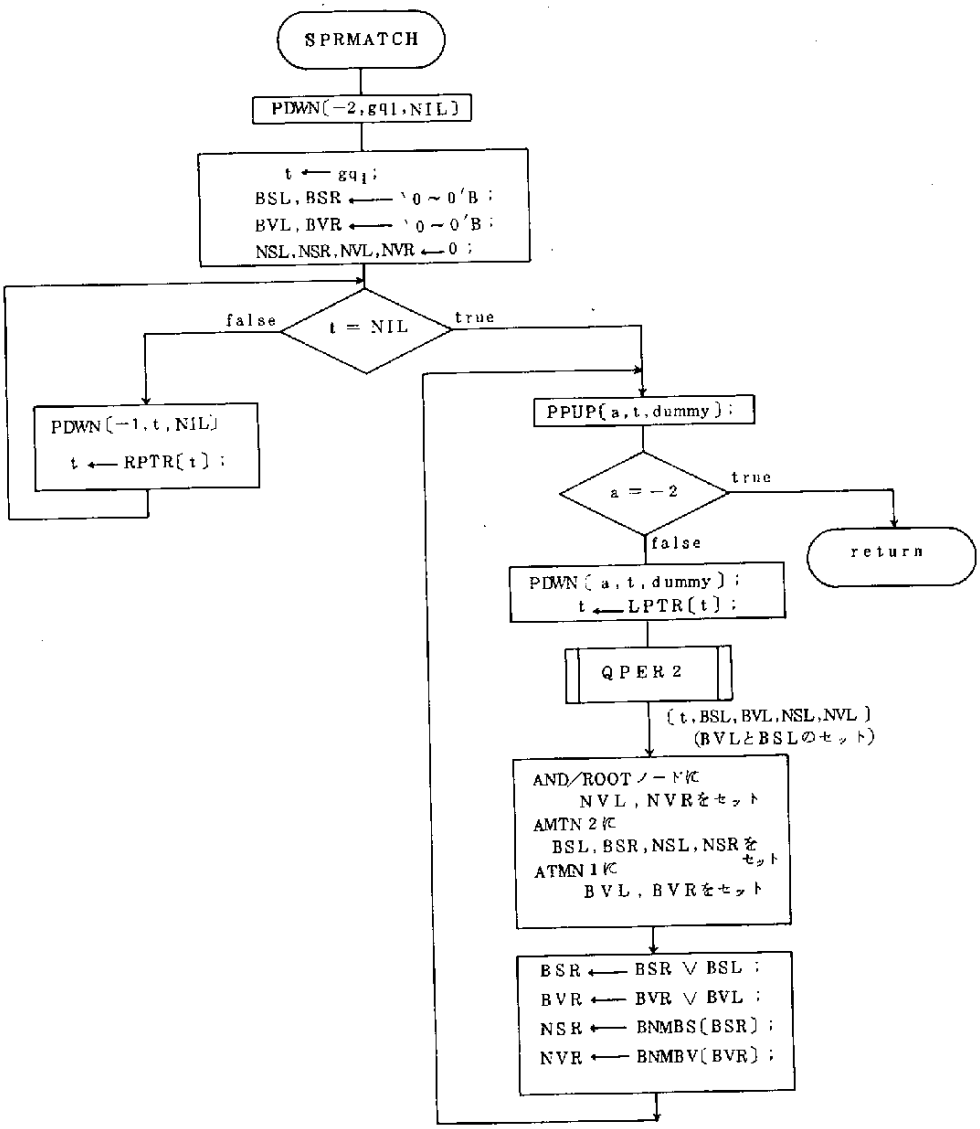


図8−117　SPRMATCHの処理概要

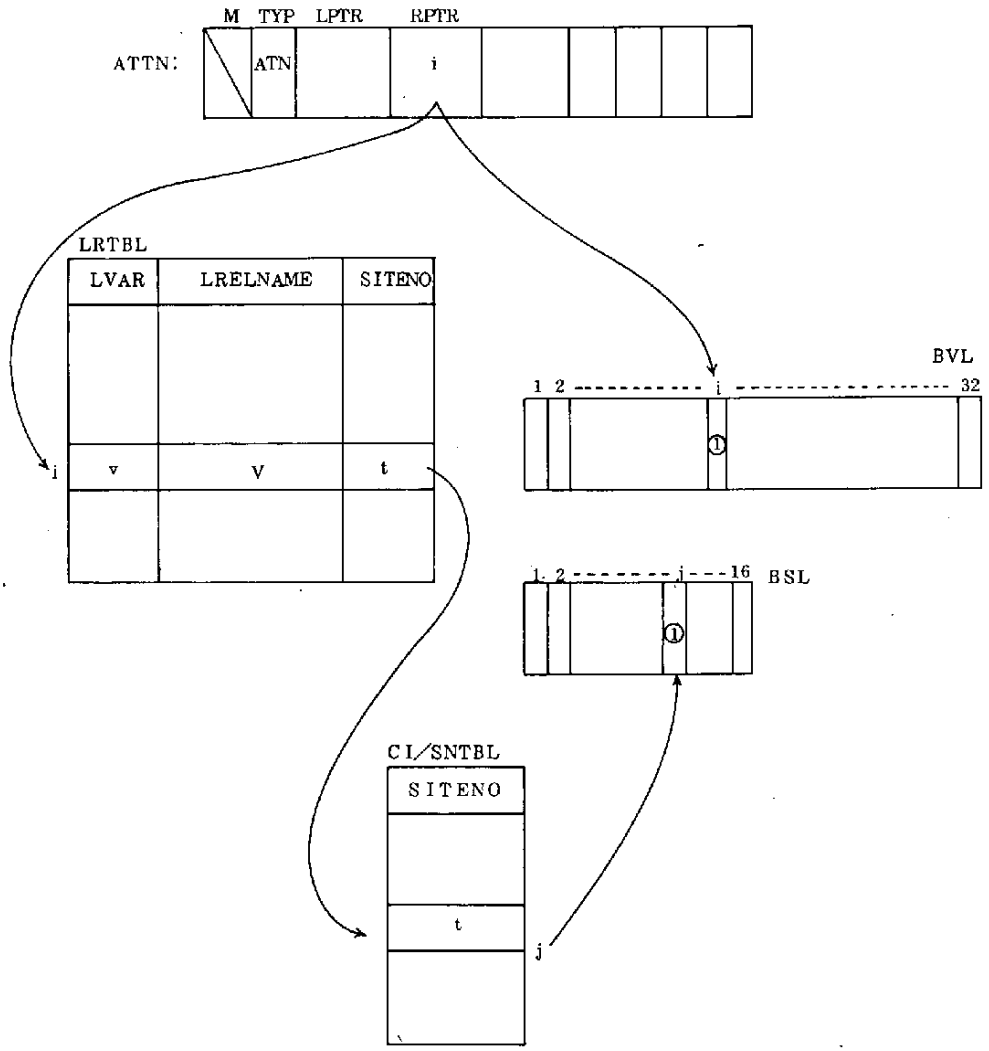


図8－118　BVLとBSLのセット

8.4.6　LQP

ＬＱＰは、サイト単位で閉じて実行できる問合せを生成し、これを処理させる。更に、これらのローカル処理の結果からなるサイト間の問合せを新たに生成する。ＬＱＰは、ＱＭＰで１つのＧＩが取り出される度に呼出されるサブルーチンとして作成されている。

Ａ．　ＬＱＰの処理概要

１）〔ｂｉｔ－ｍａｐテーブル（ＢＭ）の生成〕

ｇｌｏｂａｌ ＬＣＳｑｕｅｒｙ（ＧＬＱ）ｔｒｅｅのポインタｇｑｌがＱＭよりパラメータとして渡される。この木（ｔｒｅｅ）内の＜ｔａｒｇｅｔ－ｌｉｓｔ＞及び＜ｃｌａｕｓｅ＞へのポインタと対応するｂｉｔ－ｍａｐ（ＢＶＬとＢＳＬ）、及び、サイト数（ＮＳＬ）をもつテーブルＢＭ〔表８－５６〕を生成する。

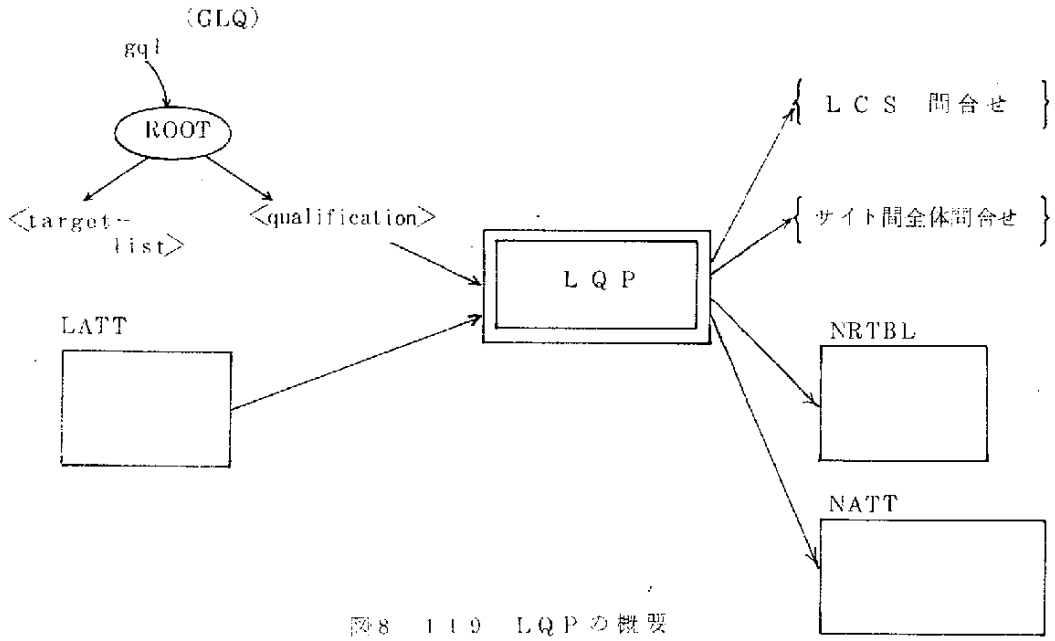
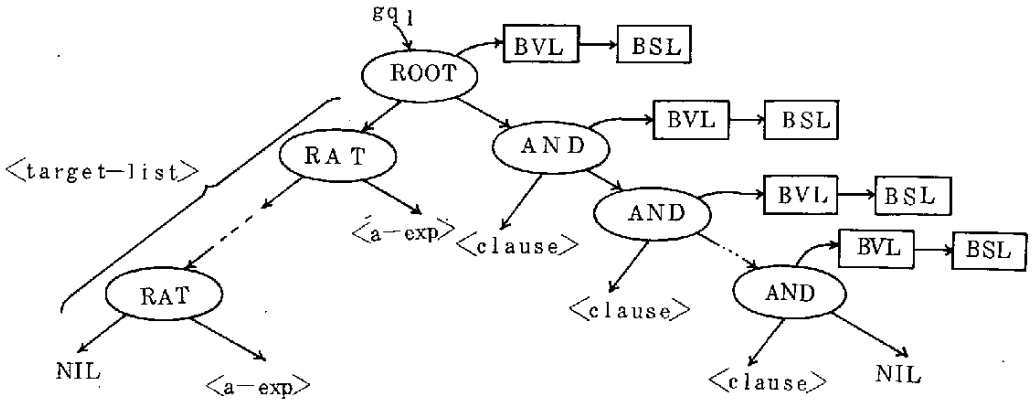


図８　１１９　ＬＱＰの概要



SEQ	CPT	BVL	BSL	NSL	
1	144	0100010000000000000000000000000000	1001000000000000	2	
2	147	0011000000000000000000000000000000	0100000000000000	1	
3	149	0000110000000000000000000000000000	0011000000000000	2	→F.F1+2=E.E2
4	151	0100000100000000000000000000000000	1001000000000000	2	→B.B1*8=H.H2
5	153	0000000100000000000000000000000000	0001000000000000	1	→H.H2="AAA"
6	174	1100000000000000000000000000000000	1000000000000000	1	
7	176	0110000000000000000000000000000000	1100000000000000	2	
8	179	0100010000000000000000000000000000	1001000000000000	2	→B.B2=F.F2
9	182	1000000000000000000000000000000000	1000000000000000	1	
10	187	0001000000000000000000000000000000	0100000000000000	1	
11	190	0000100100000000000000000000000000	0011000000000000	2	→E.E1=H.H2
12	193	0000001100000000000000000000000000	0001000000000000	1	→H.H3=G.G2
		ABCDEFGH	サイト4		

変数名に対応

図８－１２０　ＢＭの例〔 8.4.7を参照 〕

2) 〔同一サイトをもつclause のグループ化〕

ＮＳＬとＢＳＬを用いて、ある１つのサイトのみを参照する組(tuple)を全てＮＥＷとマークする。これらには制限節(restriction clause)と、サイト内結合節(join clause)が含まれている。また、それらの tuple のＢＶＬから参照されているＬＣＳ変数がわかる。

図８－１２０の例で、サイト４について考えると、5番目の tuple (H・H2 ="AAA")と１２番目の tuple (H・H3 =G・G2)がサイト内の閉じた<clause>となっていることがわかる。また、ＢＶＬから参照されている変数は、ＨとＧであることもわかる。ＢＭの５番目と１２番目はＮＥＷとマークされる。

3) 〔サイト内問合せの生成〕

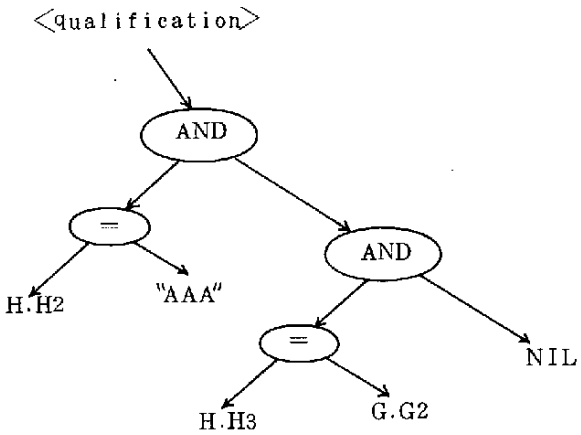
ＬＣＳ変数の bit-mapを用いて、ＮＥＷとマークされている tuple を連結なグループに分ける。これはdepth-first algorithm 〔 9.4.4 －Ｃを参照のこと〕を用いて行なわれ

る。連結なグループはＯＬＤとマークされる。

図８－１２０の例では、変数ＨとＧは連結であり、ＢＭはともにＯＬＤとマークされることになる。

4)〔＜qualification＞の生成〕

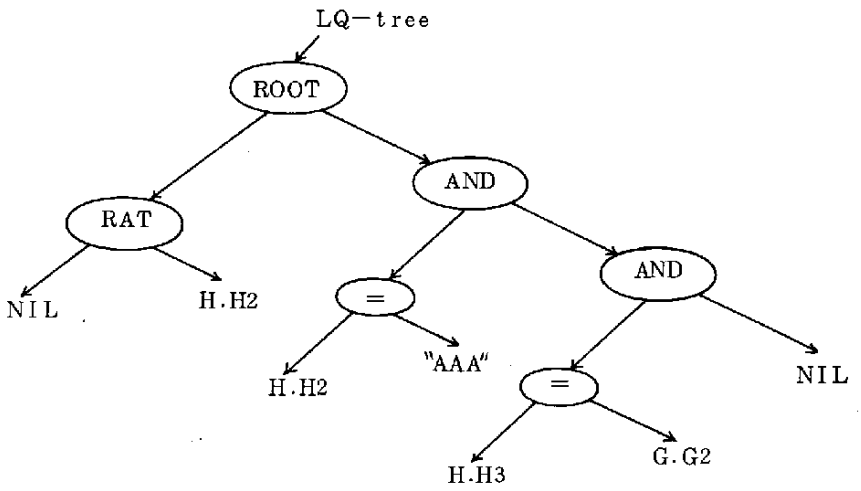
これらの連結なグループ毎に、サイト内に閉じた問合せを生成する。まず、連結なグループのclauseを全て、ＡＮＤノードを生成しながら結合し、qualification を作成する。



5)〔サイト間の join clause に含まれる属性を＜target-list ＞にセット〕

次に、２つのサイトにまたがる join clause の中から連結なグループ内の変数を含む属性をみつけ、結果属性（.result-attribute ）とする。また、gql 内の＜target-list＞から、そのサイト内の属性をとり出し、結果属性に追加する。このとき、結果属性が重複しないようにＮＲＡＴテーブル〔表８－５９〕を用いて管理する。

図８－１２０の例のサイト４については、４番目（Ｂ.B1*8＝H.H2）と１１番目（Ｅ.Ｅ１＝H.H2）がサイトにまたがった＜clause＞である。従って、H.H2が結果属性としてＮＲＡＴに登録される。また、ＧＬＱ－tree の＜target-list＞内には変数Ｇ、Ｈをもつ属性が無いので〔8.4.7を参照〕、結果属性リストは、H.H2のみとなる。サイト内問合せ木（ＬＱ－tree ）は次のようになる。



6)〔ＬＱの結果リレーションの名前づけ〕

この連結なグループから生成されたサイト内問合せの結果リレーション名および変数名として一意な名前をつけてＮＲＴＢＬ〔表８－５８〕に格納する。ＮＲＴＢＬのＱＰＴフィールドにはＬＱ－ｔｒｅｅへのポインタが入る。ここでは、新しい変数名は〝ＮＶ９９〟、結果リレーション名は〝ＮＲＥＬＮＡＭＥ９９〟としている。９９は一連番号である。

7)〔ＬＱの結果属性の名前づけ〕

結果属性リストに対応するＮＲＡＴテーブルから、新しい変数名と対にしてＮＡＴＴテーブル〔表８－５７〕を生成する。このとき、オリジナルのＬＣＳリレーションの対応する組変数（ｔｕｐｌｅ ｖａｒｉａｂｌｅ）も合せてＮＡＴＴに格納する。また、属性名はＬＣＳ属性名をそのまま用いる。但し、重複する場合はＬＣＳ属性名の前にＬＣＳ変数名を付けて一意性を保つようにする。

8)〔ＲＡＮＧＥ文の出力〕

ＬＲＴＢＬを用いて、ＲＡＮＧＥ文を出力する。例えば、サイト４については次のように出力される。

```
** LOCAL QUERY TO SITE 04 **
RANGE OF (F    ) IS (FFF);
RANGE OF (G    ) IS (GGG);
RANGE OF (H    ) IS (HHH);
```

9)〔ＬＱの出力〕

ＬＱ－ｔｒｅｅからＲＥＴＲＩＥＶＥ文を生成する。結果属性リストはＮＲＡＴテーブルから、＜ｑｕａｌｉｆｉｃａｔｉｏｎ＞についてはＯＵＴＰ〔9.2.7－Ｃ参照〕を用いてＬＱ－ｔｒｅｅ から生成される。

5）の例のＬＱ－ｔｒｅｅ では次のように出力される。

```
RETRIEVE INTO  NRELNAME02 ( H2 = H.H2 )
WHERE H.H2 = "AAA" AND H.H3 = G.G2 ;
```

10）〔サイト内 join 又は restriction をもたない変数の処理〕

同一サイト内の join 又は restriction を持たない変数については、以上の処理で新しいリレーションが生成されない。（例えば、サイト４での変数Ｆがこれに当る）。従って、サイト毎に今まで現われなかった変数について、サイト間 join および＜target-list＞をサーチして、結果属性リストのみをもつＬＣＳ問合せを生成する。

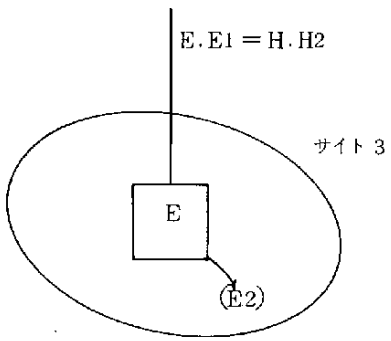
図８－１２０の例では、変数Ｆについて次のＬＣＳ問合せが生成される。

```
RETRIEVE INTO  NRELNAME03 ( F1 = F.F1 , F2 = F.F2 )
```

11）〔ＬＣＳ問合せの生成されていないサイトの処理〕

更に、以上の処理では、サイト内の全ての変数についてサイト間の join しかないようなサイトについての問合せは生成されないので、再びＢＭをサーチして、今まで現われなかった変数について、新しくＢＭを生成する。即ち、ＢＭのＣＰＴ＝ＮＩＬ（qualification無し）として、ＢＶＬ、ＢＳＬ、ＮＳＬをセットする。こうして再び処理 2）から繰り返す。

8.4.7 の例では、サイト３がこれにあたる。



従って、ＢＭに変数Ｅについての組（tuple）を追加する。

SEQ	CPT	BVL	BSL	NSL
1	144	0100010000000000000000000000000000	1001000000000000	2
2	147	0011000000000000000000000000000000	0100000000000000	1
3	149	0000110000000000000000000000000000	0011000000000000	2
4	151	0100000100000000000000000000000000	1001000000000000	2
5	153	0000000100000000000000000000000000	0001000000000000	1
6	174	1100000000000000000000000000000000	1000000000000000	1
7	176	0110000000000000000000000000000000	1100000000000000	2
8	179	0100010000000000000000000000000000	1001000000000000	2
9	182	1000000000000000000000000000000000	1000000000000000	1
10	187	0001000000000000000000000000000000	0100000000000000	1
11	190	0000100100000000000000000000000000	0011000000000000	2
12	193	0000001100000000000000000000000000	0001000000000000	1
13	0	0000100000000000000000000000000000	0010000000000000	1

↓ NIL　↓ E　↓ サイト３

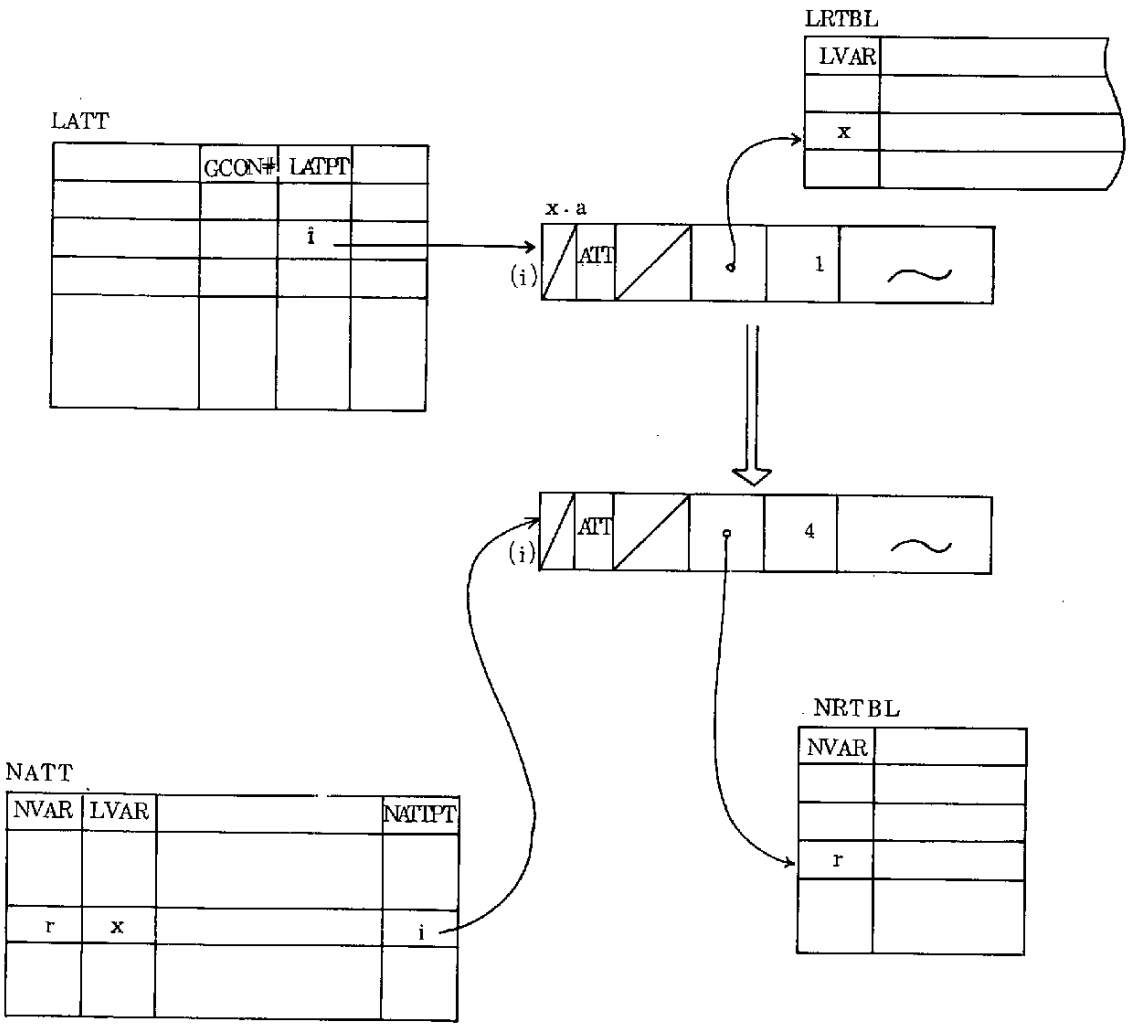
→ サイト３、変数Ｅについての tuple

サイト３でのＬＣＳ問合せは次のようになる。

```
** LOCAL QUERY TO SITE 03 **
RANGE OF (E    ) IS (EEE);

RETRIEVE INTO  NRELNAME05 ( E2 = E.E2 , E1 = E.E1 )
```

12) 全てのサイト内のＬＱの生成が終ったら、新しく生成されたリレーションに基づいて、サイト間の join clause から成る問合せの生成を行なう。

まず、ＬＡＴＴを用いてgqlの参照する属性ノードの組変数(tuple variable) へのポインタを、新しいＮＲＴＢＬの組変数へのポインタに置きかえる。同時に、属性番号もおきかえる。このＡＴＴＮへのポインタをＮＡＴＴにもセットする。



13）〔サイト間問合せの＜qualification ＞の生成〕

BMをサーチして、サイト間のjoin clause を見つけ、ANDノードで結合し、qualificationを生成する。＜target-list ＞及び＜qualification＞内の属性は11）で新しい結果リレーションに関するものに置き換えられている。

14）〔サイト間問合せの出力〕

新しい問合せを出力する。結果リレーション名はGQRRELより、結果属性名はGQRATTより得られる。qualification はOUTPを用いて通常の表現（infix notation）に変換される。

図8－120の例からは、次のサイト間の問合せが生成される。

```
**  INTER-SITE QUERY  **
RANGE OF (NV01) IS (NRELNAME01);
RANGE OF (NV02) IS (NRELNAME02);
RANGE OF (NV03) IS (NRELNAME03);
RANGE OF (NV04) IS (NRELNAME04);
RANGE OF (NV05) IS (NRELNAME05);

RETRIEVE INTO RESULT ( R1 = NV04.B1 * 8 , R3 = NV03.F1 + 2 )
WHERE NV03.F1 + 2 = NV05.E2 AND NV04.B1 * 8 = NV02.H2 AND NV04.B1 = NV01
.C2 AND NV04.B2 = NV03.F2 AND NV05.E1 = NV02.H2 ;
```

B．LQPの用いるテーブル

表8－56　BMの構成

属性名	属性番号	ROLE	TYPE	LENGTH	説明
CPT	1	N	D	2	左手の部分木へのポインタ
BVL	2	N	C	4	変数についてのbit－map
BSL	3	N	C	2	所在サイトについてのbit－map
NSL	4	N	D	2	サイト数
MARK	5	N	D	2	作業用

表８－５７　　ＮＡＴＴの構成

属　性　名	属性番号	ROLE	TYPE	LENGTH	説　　明
NVAR	1	K	C	4	新しい組変数名
LVAR	2	N	C	4	前の組変数名
NATTNO	3	N	D	2	新しい属性番号
NATTNAME	4	N	C	16	新しい属性名
NATTPT	5	N	D	2	ＡＴＴＮへのポインタ
NATTPN	6	N	D	2	ＡＴＴＮへのポインタ

表８－５８　　ＮＲＴＢＬの構成

属　性　名	属性番号	ROLE	TYPE	LENGTH	説　　明
NVAR	1	K	C	4	新しい組変数名
NRELNAME	2	N	C	16	新しい結果リレーション名
DEGREE	3	N	D	2	新しい属性の数
SITENO	4	N	D	2	所在サイト番号
QPT	5	N	D	2	ＬＱ－treeへのポインタ

表８－５９　　ＮＲＡＴの構成

属　性　名	属性番号	ROLE	TYPE	LENGTH	説　　明
LRT	1	K	D	2	ＬＲＴＢＬへのポインタ
LATTN	2	K	D	2	属性番号
LATPT	3	N	D	2	ＡＴＴＮへのポインタ
LATPN	4	N	D	2	

Ｃ．ＬＱＰの出力形式

a）ローカル問合せの出力

```
" ** LOCAL QUERY TO SITE␣"<site-no>" ␣** "
    <range-statement>
[{<range-statement>}]
    <retrieve-statement>
[{<retrieve-statement>}]
```

<range-statement> ::= " RANGE OF ("<lvar>") IS
("lrelname") ; "

<retrieve-statement> ::= " RETRIEVE INTO "<nrelname>
" ("<target-list>") "
[" WHERE "<qualification>]" ; "

<target-list> ::= <t-clause> { " , "<t-clause> }

<t-clause> ::= <nattname> " = "<oa-exp>

<lvar> ::= LRTBL . LVAR(i)
<lrelname> ::= LRTBL . LRELNAME(i)
} ここで LRTBL · SITENO(i) =<site-no>

<nattname> ::= NATT . NATTNAME(j)

<oa-exp> ::= NATT . NATTPT(j) が指すLCS attribute を参照する tree を infix notation に変換したもの

<nrelname> ::= NRTBL . NRELNAME(k)
ここでNRTBL . NVAR(k)=NATT . NVAR(j)

b) サイト間の問合せの出力

" ** INTER-SITE QUERY ** "
<range-statement>
[{ <range-statement> }]
<retrieve-statement>

<range-statement> ::= " RANGE OF ("<nvar>") OF
("nrelname") ; "

<retrieve-statement> ::= " RETRIEVE INTO␣ "<result-relname>" ("<target-list>") "
[" WHERE " <qual>]" ; "

<nvar> ::= NRTBL . NVAR(i)

<nrelname> ::= NRTBL . NRELNAME(i)

<target-list> ::= $<t\text{-}clause>_1$ { " , "$<t\text{-}clause>_k$ }

$<t\text{-}clause>_k$ ::= $<result\text{-}attname>_k$ = <na-exp>

<result-relname> ::= GQRREL . GRELNAME

$<result\text{-}attname>_k$ ::= GQRATT . GQATTNAME(i)
ここで、GQRATT . GATTNO = k

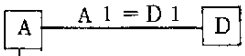
＜na−exp＞::＝new attribute を参照する式。＜target−list＞ tree の属性番号＝k に対応する表現を infix notation に変換したもの。

８．４．７　ＱＤＰの例

問合せ図では、

□ はリレーションの組変数を表わす。

A、B、C、……は、変数名であり、それらのリレーション名はＡＡＡ、ＢＢＢ、ＣＣＣ、……である〔 variable と schema による〕。

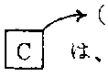


は、分散記述において、

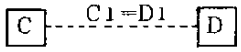
〝 A.A1 ＝D.D1 〟という、join clause があることを示している。

は、restriction clause A.A2 ＝〝Ａ〟があることを示す。

＝
A2 ＝〝Ａ〟

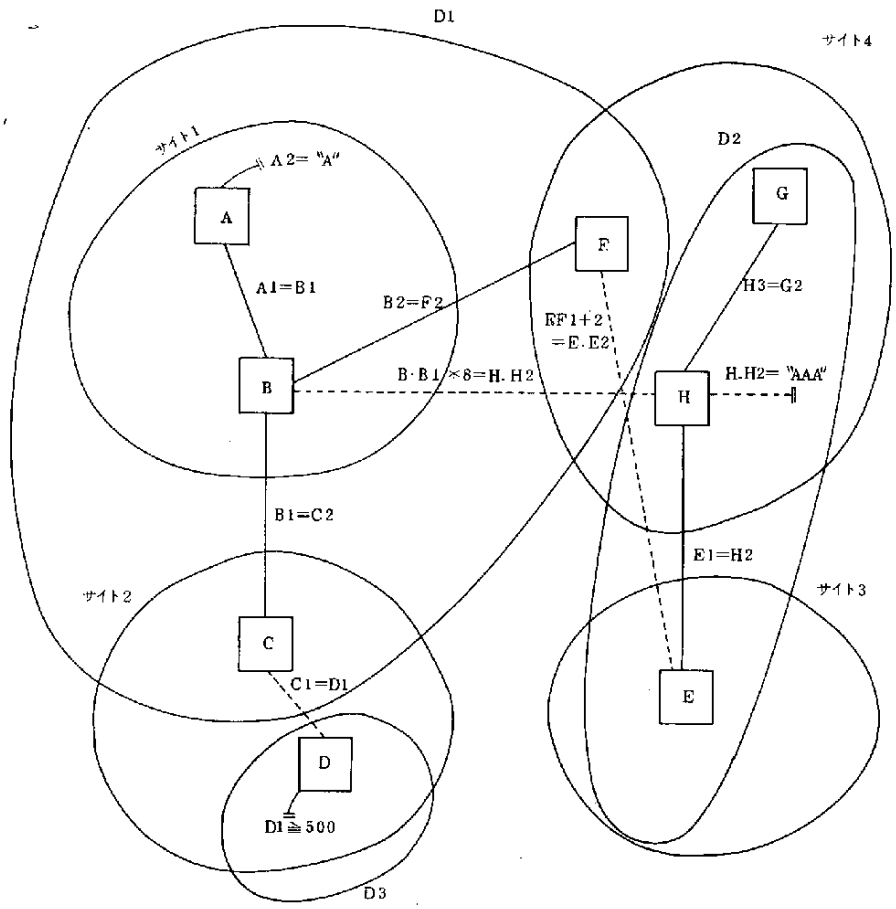


は、C.C1 が result−attribute であることを表わす。



は、ＧＣＳ問合せによって join link がはられていることを表わす。

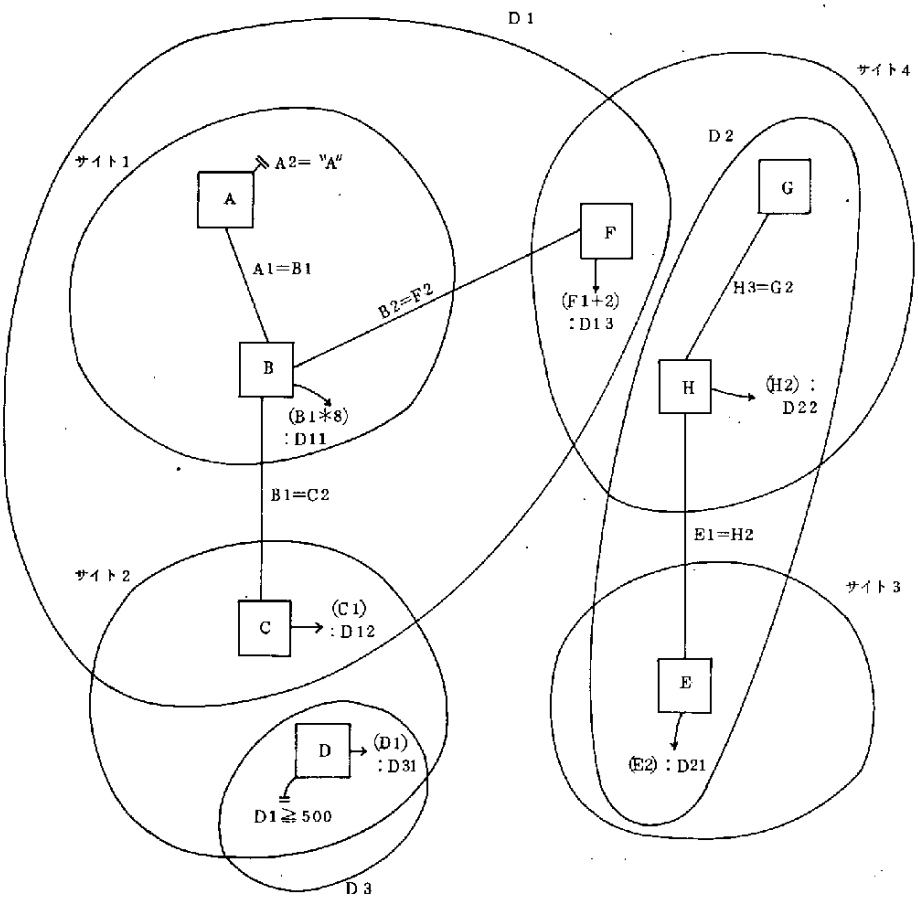
1) 問合せ図



2) variable と schema

variable	schema	サイト
A	AAA (A1、A2、A3)	1
B	BBB (B1、B2、B3)	1
C	CCC (C1、C2、C3)	2
D	DDD (D1、D2、D3)	2
E	EEE (E1、E2、E3)	3
F	FFF (F1、F2、F3)	4
G	GGG (G1、G2、G3)	4
H	HHH (H1、H2、H3)	4

3) 分散記述（ＤＤ）

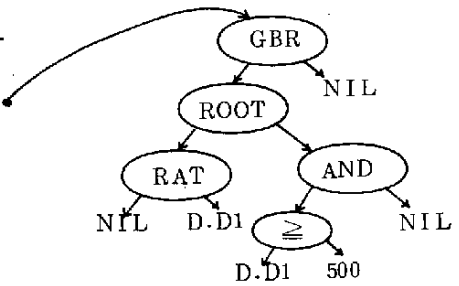



```
DI;
RANGE ( A, B     )  ( AAA:1, BBB:1              );
RANGE ( C, D     )  ( CCC:2, DDD:2              );
RANGE ( E        )  ( EEE:3                     );
RANGE ( F, G, H  )  ( FFF:4, GGG:4, HHH:4       );
DEFINE ESR  D1 ( D11, D12, D13       )
     ( D11 = B.B1 * 8, D12 = C.C1, D13 = F.F1 +2       )
     WHERE
               A.A1 = B.B1    AND  B.B1 = C.C2    AND  B.B2 = F.F2    AND
               A.A2 = "A"
     ;
DEFINE ESR  D2 ( D21, D22  )
     ( D21 = E.E2, D22 = H.H2 )
     WHERE     E.E1 = H.H2    AND  H.H3 = G.G2
     ;
DEFINE ESR  D3 ( D31 )
     ( D31 =  D.D1    )
     WHERE     D.D1 GE 500
     ;
```

4) ＤＩＰＳにより次のテーブルが生成される。

ＣＩ／ＧＲＥＬ

SEQ	GRELNAME	RTYPE	DEGREE	DDPT
1	D1	ES	3	23
2	D2	ES	2	46
3	D3	ES	1	58



ＣＩ／ＧＡＴＴ

SEQ	GRELNAME	GATTNO	GATTNAME	DOMAIN
1	D1	1	D11	D11
2	D1	2	D12	D12
3	D1	3	D13	D13
4	D2	1	D21	D21
5	D2	2	D22	D22
6	D3	1	D31	D31

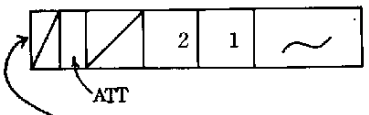
ＣＩ／ ＲＴＢＬ

SEQ	LVAR	LRELNAME	SITENO
1	A	AAA	1
2	B	BBB	1
3	C	CCC	2
4	D	DDD	2
5	E	EEE	3
6	F	FFF	4
7	G	GGG	4
8	H	HHH	4

ＣＩ／ＳＮＴＢＬ

SEQ	SITENO
1	1
2	2
3	3
4	4

B.B1を表わす 属性ノード



CI／LATT

SEQ	LVAR	LATTNO	LATTNAME	GCON#	LATPT
1	B	1	B1	1	1
2	C	1	C1	1	4
3	F	1	F1	1	6
4	A	1	A1	1	9
5	C	2	C2	1	11
6	B	2	B2	1	13
7	F	2	F2	1	14
8	A	2	A2	1	16
9	E	2	E2	1	35
10	H	2	H2	1	37
11	E	1	E1	1	39
12	H	3	H3	1	41
13	G	2	G2	1	42
14	D	1	D1	1	54

5) GCS問合せ（GQ）

```
@D;
RANGE ( D1, D2, D3 )     ( D1, D2, D3    );
RETRIEVE INTO RESULT ( R1 = D1.D11,                    R3 = D1.D13 )
WHERE
          D1.D12  =  D3.D31   AND  D1.D13  =  D2.D21   AND
          D1.D11  =  D2.D22   AND
          D2.D22  =  "AAA"
          ;
```

6) GQTRGにより、次のテーブルが生成される。

GQRREL

SEQ	GRELNAME	DEGREE	GQPT
1	RESULT	2	80

GQRATT

SEQ	GQRELNAME	GQATTNO	GQATTNAME
1	RESULT	1	R1
2	RESULT	2	R3

GQRTBL

SEQ	GVAR	GRELNAME	DDPT	MARK	GQPT
1	D1	D1	117	1	117
2	D2	D2	131	1	131
3	D3	D3	140	1	140

DDATCHでセットされる

GQGATT

SEQ	GVAR	GATTNO	GATTNAME	GCON#	GATPT	MARK
1	D1	1	D11	0	64	0
2	D1	3	D13	0	66	0
3	D1	2	D12	0	68	0
4	D3	1	D31	0	69	0
5	D2	1	D21	0	71	0
6	D2	2	D22	0	73	0
7	D1	1	D11	0	141	0
8	D1	3	D13	0	143	0
9	D1	2	D12	0	145	0
10	D3	1	D31	0	146	0
11	D2	1	D21	0	148	0
12	D2	2	D22	0	150	0

7) QNでは、次のテーブルが生成される。

VBM

SEQ	PTR	VBL	MARK
1	67	1000000000000000000000000000000000	0
2	70	1010000000000000000000000000000000	1
3	72	1100000000000000000000000000000000	1
4	76	0100000000000000000000000000000000	1

D3
D2
D1

8) QMでは、次のテーブルを生成する。

LRTBL

SEQ	LVAR	LRELNAME	SITENO	OLDLVAR
1	A	AAA	1	1
2	B	BBB	1	2
3	C	CCC	2	3
4	D	DDD	2	4
5	E	EEE	3	5
6	F	FFF	4	6
7	G	GGG	4	7
8	H	HHH	4	8

⟸ CI/RTBLより

LATT

SEQ	LVAR	LATTNO	LATTNAME	GCON#	LATPT	OLDLVAR
1	B	1	B1	1	1	2
2	C	1	C1	1	4	3
3	F	1	F1	1	6	6
4	A	1	A1	1	9	1
5	C	2	C2	1	11	3
6	B	2	B2	1	13	2
7	F	2	F2	1	14	6
8	A	2	A2	1	16	1
9	E	2	E2	1	35	5
10	H	2	H2	1	37	8
11	E	1	E1	1	39	5
12	H	3	H3	1	41	8
13	G	2	G2	1	42	7
14	D	1	D1	1	54	4

⟸ CI/LATTより

LOCAT

SEQ	RTP	ATT#	ATPT
1	2	1	169
2	6	1	171
3	3	1	145
4	4	1	146
5	5	2	148
6	8	2	150
7	1	1	173
8	3	2	175
9	2	2	177
10	6	2	178
11	1	2	180
12	5	1	189
13	8	3	191
14	7	2	192

9) LQPでは、次のようなテーブルを1つの＜sub-def＞の組み合せ毎に生成する。

BM

SEQ	CPT	BVL	BSL	NSL	MARK
1	144	0100010000000000000000000000010000	1001000000000000	2	0
2	147	0011000000050000000000000000000000	0100000000000000	1	1
3	149	0000110000000000000000000000000000	0011000000000000	2	0
4	151	0100000100000000000000000000000000	1001000000000000	2	0
5	153	0000000100000000000000000000000000	0001000000000000	1	1
6	174	1100000000000000000000000000000000	1000000000000000	1	1
7	176	0110000000000000000000000000000000	1100000000000000	2	0
8	179	0100010000000000000000000000000000	1001000000000000	2	0
9	182	1000000000000000000000000000000000	1000000000000000	1	1
10	187	0001000000000000000000000000000000	0100000000000000	1	1
11	190	0000100100000000000000000000000000	0011000000000000	2	0
12	193	0000001100000000000000000000000000	0001000000000000	1	1
13	0	0000100000000000000000000000000000	0010000000000000	1	1

10) 1つのサイト内問合せ毎に結果属性リストを管理するテーブルとして、NRATがある。

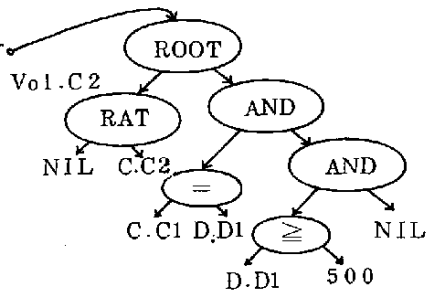
NRAT

SEQ	LRT	LATTN	LATPT	LATPN	
1	5	2	148	0	＜ E.E2
2	5	1	189	0	＜ E.E1 [NATTを参照]

11) 新しいリレーション名と属性名を生成し、サイト内問合せを出力する度に、次のテーブルに組(tuple)を追加する。

NRTBL

SEQ	NVAR	NRELNAME	DEGREE	SITENO	QPT
1	NV01	NRELNAME01	1	2	199
2	NV02	NRELNAME02	1	4	203
3	NV03	NRELNAME03	2	4	206
4	NV04	NRELNAME04	2	1	211
5	NV05	NRELNAME05	2	3	215



NATT

SEQ	NVAR	LVAR	NATTNO	NATTNAME	NATTPT
1	NV01	C	1	C2	175
2	NV02	H	1	H2	150
3	NV03	F	1	F1	171
4	NV03	F	2	F2	178
5	NV04	B	1	B1	169
6	NV04	B	2	B2	177
7	NV05	E	1	F2	148
8	NV05	E	2	E1	189

12) 各サイト内での問合せ

```
** LOCAL QUERY TO SITE 02 **
RANGE OF (C    ) IS (CCC);
RANGE OF (D    ) IS (DDD);

RETRIEVE INTO  NRELNAME01 ( C2 = C.C2 )
WHERE C.C1 = D.D1 AND D.D1 GE 500 ;
```

（サイト２）

```
** LOCAL QUERY TO SITE 04 **
RANGE OF (F    ) IS (FFF);
RANGE OF (G    ) IS (GGG);
RANGE OF (H    ) IS (HHH);

RETRIEVE INTO  NRELNAME02 ( H2 = H.H2 )
WHERE H.H2 = "AAA" AND H.H3 = G.G2 ;

RETRIEVE INTO  NRELNAME03 ( F1 = F.F1 , F2 = F.F2 )
;
```

（サイト４）

（サイト１）

```
** LOCAL QUERY TO SITE 01 **
RANGE OF (A    ) IS (AAA);
RANGE OF (B    ) IS (BBB);

RETRIEVE INTO  NRELNAME04 ( B1 = B.B1 , B2 = B.B2 )
WHERE A.A1 = B.B1 AND A.A2 = "A" ;
```

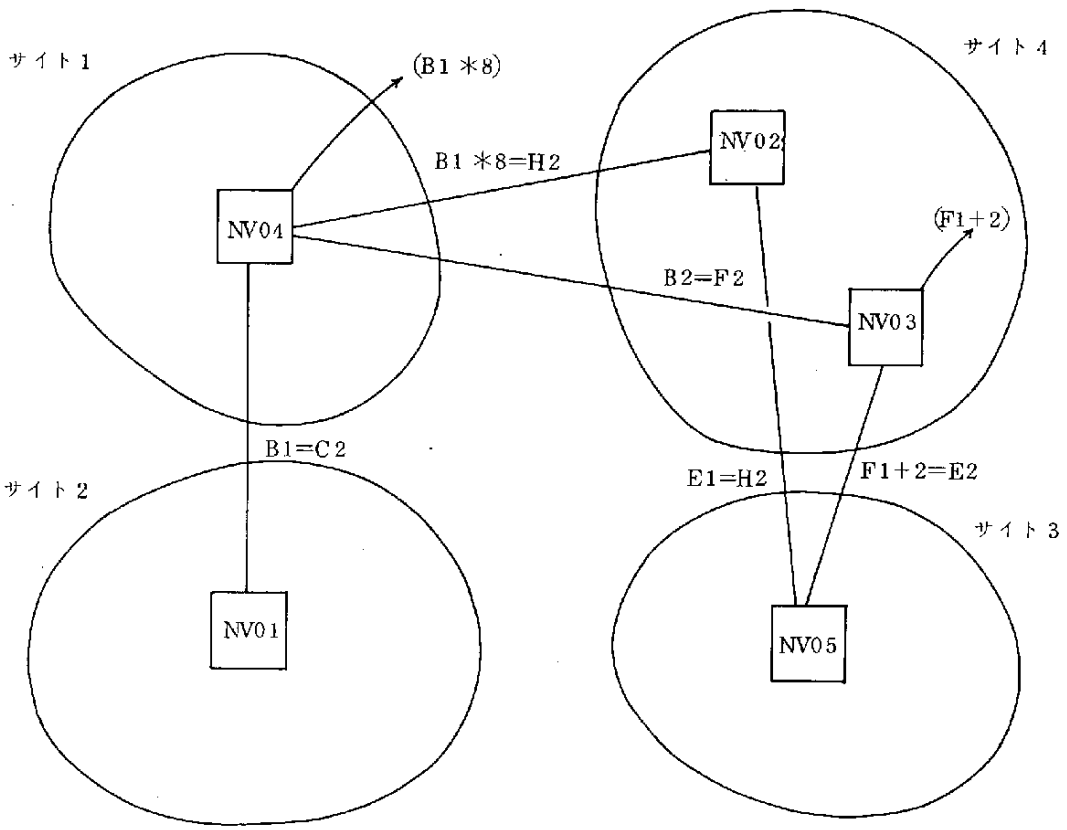
（サイト３）

```
** LOCAL QUERY TO SITE 03 **
RANGE OF (E    ) IS (EEE);

RETRIEVE INTO  NRELNAME05 ( E2 = E.E2 , E1 = E.E1 )
;
```

13) サイト間問合せの出力

```
**  INTER-SITE QUERY  **
RANGE OF (NV01) IS (NRELNAME01);
RANGE OF (NV02) IS (NRELNAME02);
RANGE OF (NV03) IS (NRELNAME03);
RANGE OF (NV04) IS (NRELNAME04);
RANGE OF (NV05) IS (NRELNAME05);

RETRIEVE INTO RESULT ( R1 = NV04.B1 * 8 , R3 = NV03.F1 + 2 )
WHERE NV03.F1 + 2 = NV05.E2 AND NV04.B1 * 8 = NV02.H2 AND NV04.B1 = NV01
.C2 AND NV04.B2 = NV03.F2 AND NV05.E1 = NV02.H2 ;
```

14) サイト間の問合せ図



8.5 操 作 説 明

ＱＴＰ及びＱＤＰはＴＳＳ環境下で動くように作成されている。即ち、ＱＴＰではＴＳＳ端末からＬＣＳ問合せを入力すると端末上にＤＢＴＧ　ＤＭＬから成るＣＯＢＯＬプログラムが出力される。また、ＱＤＰでは、輸送スケジューラ（ＴＳ）がインプリメントされていないので、分散記述及びＧＣＳ問合せを入力すると、全てのサイトについてのサイト内問合せ、及びサイト間問合せが端末上に出力される。

ＨＩＰＳ及びＤＩＰＳは、ともにバッチシステムとして動作する。

以下では、各システムの実行に必要なジョブ制御文、又はＴＳＳ端末での操作等を説明する。

8.5.1 ＨＩＰＳの操作説明

操作の点からみると、ＨＩＰＳの処理は次の３つに分けることができる。

1） ＬＣＳ／ＨＩ記述から、ＨＩ情報としての１０個の出力ファイル（リレーション）を生成する。（ＨＩＬＣＳＧＥＮ）

2） 生成されたリレーションの内容をプリントする。（ＨＩＰＲＩＮＴ）

3） リレーショナル記述により、生成されたＬＣＳ／ＨＩ情報に対して問合せを行なう。（ＲＤＣＯＭＰ）

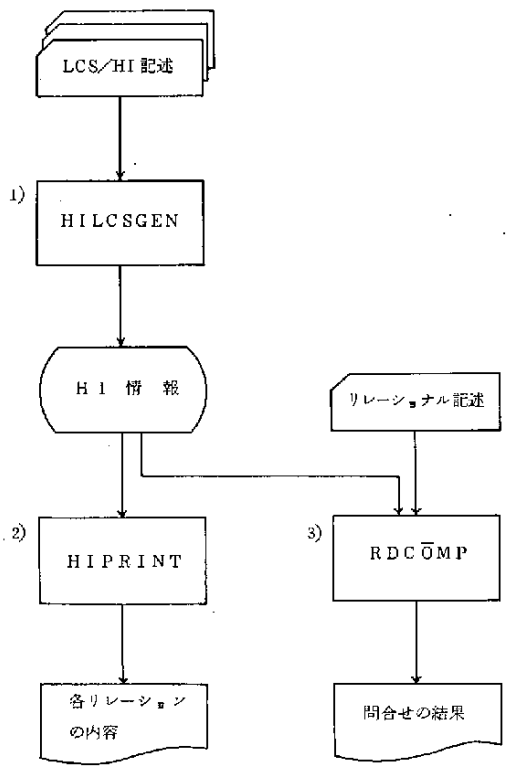


図８－１２１　ＨＩＰＳの処理概要

A　ＬＣＳ／ＨＩ記述の解読

図８－１２２にジョブ制御文を示す。出力ファイルは既に登録されているものとする。

```
//BUNSAN    JOB  0106,DIS213,REGION=500K
//JOBLIB  DD  DSN=HAMA.LOAD,UNIT=SYSDA,VOL=SER=USER03,DISP=SHR
// EXEC  PGM=HILCSGEN,REGION=144K
//RDC           DD  DDNAME=CARD
//OUT           DD  DDNAME=PRINT
//HIESR         DD  DSNAME=HIESR,UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE),
//              DCB=BLKSIZE=480,DISP=(OLD,KEEP),VOL=SER=USER03
//HIATT         DD  DSNAME=HIATT,UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE),
//              DCB=BLKSIZE=480,DISP=(OLD,KEEP),VOL=SER=USER03
//HIRSR         DD  DSNAME=HIRSR,UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE),
//              DCB=BLKSIZE=480,DISP=(OLD,KEEP),VOL=SER=USER03
//HIITGVAL      DD  DSNAME=HIITGVAL,UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE),
//              DCB=BLKSIZE=480,DISP=(OLD,KEEP),VOL=SER=USER03
//HIITGATT      DD  DSNAME=HIITGATT,UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE),
//              DCB=BLKSIZE=480,DISP=(OLD,KEEP),VOL=SER=USER03
//HIITGREL      DD  DSNAME=HIITGREL,UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE),
//              DCB=BLKSIZE=480,DISP=(OLD,KEEP),VOL=SER=USER03
//HIPRTREL      DD  DSNAME=HIPRTREL,UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE),
//              DCB=BLKSIZE=480,DISP=(OLD,KEEP),VOL=SER=USER03
//HIPRTMEM      DD  DSNAME=HIPRTMEM,UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE),
//              DCB=BLKSIZE=480,DISP=(OLD,KEEP),VOL=SER=USER03
//HIPRTATT      DD  DSNAME=HIPRTATT,UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE),
//              DCB=BLKSIZE=480,DISP=(OLD,KEEP),VOL=SER=USER03
//HIITGPRT      DD  DSNAME=HIITGPRT,UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE),
//              DCB=BLKSIZE=480,DISP=(OLD,KEEP),VOL=SER=USER03
//PRINT         DD  SYSOUT=A
//CARD          DD  *
##DB/PROJECT/PROJ/DBTG/ACOS/C
ESRC REPRESENTATIVE   101 PROAREA.          CD PN IN 0010
##END
/*
//
```

LCS/HI記述の一部分

図８－１２２　ＨＩＬＣＳＧＥＮのジョブ制御文

B　HI情報のプリント

HILCSGENにより生成されたリレーションの内容をプリントするためのジョブ制御文を図8－123に示す。

```
//GUNSAN   JOB  0106,DIS213,REGION=800K
//  EX C  ,GO=HI,PRINT,SEGLVL=125K
//OUT      DD   FDAA=E=PRINT
//HIESR    DD   DSNAME=HIESR,UNIT=SYSDA,VOL=SER=LS=R03,DISP=OLD
//HIRSD    DD   DSNAME=HIRSD,UNIT=SYSDA,VOL=SER=USER03,DISP=OLD
//HIATT    DD   DSNAME=HIATT,UNIT=SYSDA,VOL=SER=USER03,DISP=OLD
 /HIITGREL DD   DSNAME=HIITGREL,UNIT=SYSDA,VOL=SER=USER03,DISP=OLD
//HIITGATT DD   DSNAME=HIITGATT,UNIT=SYSDA,VOL=SER=USER03,DISP=OLD
//HIITGVAL DD   DSNAME=HIITGVAL,UNIT=SYSDA,VOL=SER=USER03,DISP=OLD
//HIPRTREL DD   DSNAME=HIPRT.EL,UNIT=SYSDA,VOL=SER=USER03,DISP=OLD
//HIPRTATT DD   DSNAME=HIPRTATT,UNIT=SYSDA,VOL=SER=USER03,DISP=OLD
//HIPRTMEM DD   DSNAME=HIPRTMEM,UNIT=SYSDA,VOL=SER=USER03,DISP=OLD
//HIITGPRT DD   DSNAME=HIITGPRT,UNIT=SYSDA,VOL=SER=USER03,DISP=OLD
//PRINT    DD   SYSOUT=A
//
```

図8－123　HIPRINTのジョブ制御文

C．HI情報に対する問合せ

生成されたLCS／HI情報に対する問合せのジョブ制御文を図8－124に示す。

```
//GUNSAN   JOB   0106,DIS213
//JOBLIB   DD    DSN=HANA.LOAD,UNIT=SYSDA,VOL=SER=USER03,DISP=SHR
//  EXEC   PGM=RDCOMP,REGION=128K
//HIATT          DD   DSNAME=HIATT,UNIT=SYSDA,VOL=SER=USER03,
//                    DISP=OLD
//OUT      DD,   DSNAME=PRINT
//PC       DD    DDNAME=CARD
//PRINT    DD    SYSOUT=A
//CARD           DD   *
    HI/ATT   ( TYPE = R5, PRODUCT , PROJ ,    ACCS   )      } リレーショナル
             ( REPR-PROJ, PROJ-SPON-LINK ,  ATT     )      } 記述の一例
/*
//
```

図8－124　RDCOMPのジョブ制御文

8.5.2 ＱＴＰの操作説明

ＴＳＳ環境下でのＱＴＰの実行に必要な処理およびコマンドについて述べる。下線の部分が入力コマンドであり、その他はシステムからのメッセージである。

1） セッションの開始（ＬＯＧＯＮコマンド）

```
LOGON TSS JNW0264/JNW
JNW0264 LOGON IN PROGRESS AT 15:30:28 ON MARCH 28, 1980

READY
```

ここで、ＪＮＷ０２６４はユーザ登録名、ＪＮＷはパスワードの例である。

2） ＱＴＰで必要とするデータセット（ファイル）を割り当てる。（ＡＬＬＯＣＡＴＥコマンド）

ＱＴＰでは、ＨＩ情報のリレーションのうち、ＨＩ／ＥＳＲ、ＨＩ／ＲＳＲ、ＨＩ／ＡＴＴを用いる。また、ＤＭＬの基本型及びアクセスパスライブラリとしてのファイルも必要とする。

即ち、次の４つのデータセットを割り当てなければならない。

```
ALLOC DA('HIESR') FILE(HIESR) OLD VOL(USER03)
READY
ALLOC DA('HIRSR') FILE(HIRSR) OLD VOL(USER03)
READY
ALLOC DA('HIATT') FILE(HIATT) OLD VOL(USER03)
READY
ALLOC DA('DBTGDML') FILE(DMLF) OLD VOL(USER03)
READY
```

3） ＱＴＰを実行させる。（ＣＡＬＬコマンド）

```
CALL BUNSAN(QTPTSS)

* JIPNET-DDB STARTED TIME 15:31:36
$01
```

ここで、ＢＵＮＳＡＮは、ロードモジュールの格納されている区分データセット名がＪＮＷ０２６４．ＢＵＮＳＡＮ．ＬＯＡＤであることを示している。ロードモジュール名はＱＴＰＴＳＳである。下の２行は、ＱＴＰから出力されたメッセージである。

これ以降は、ＱＴＰ内のコマンド及びＬＣＳ問合せの入力が可能となる。

4） ＬＣＳ問合せを入力する。

次に続くＲＡＮＧＥ文、ＲＥＴＲＩＥＶＥ文がＬＣＳ問合せであることを示すために、コマンドＱＴがある。

ＬＣＳ問合せの入力例を次に示す。

```
$01 QT;
$02 RANGE OF  (R,P,RP) (REPRESENTATIVE,PROJECT,
$03                     REPR-PROJ);
$04 RETRIEVE INTO RESULT (R.REPRNAME, P.PROJNAME)
$05    WHERE  R.REPR-NO  =  RP.REPR-NO  AND  RP.PROJ-NO  =  P.PROJ-NO
$06      AND  R.REPR-NO  =  2000        AND  P.PROJ-NO  =  1999;
$07
```

ＬＣＳ問合せの詳細については、4.3.2　及び付記１に述べてある。

5）　ＬＣＳ問合せを処理する。

ＧＯコマンドを入力すると、ＬＣＳ問合せに対応したＤＢＴＧ　ＤＭＬから成るＣＯＢＯＬプログラムが出力される。

```
$18 GO;
      *---------- DBTG DML ----------

                MOVE FALSE TO LFOUND.
       L0101.   CALL  GET-NEXT-VALUE ((PROJ-NO =  1999 ),VAL,MODE).
                IF  MODE = 'END'  GO TO TERM.
                MOVE VAL TO PROJ-NO IN PROJECT.
                FIND ANY PROJECT.
                IF  NOTFOUND = 'YES'  GO TO L0101.
                GET PROJECT.
                CALL  RESULT (PROJNAME ).
      *
       L0201.   FIND OWNER WITHIN REPR-PROJ.
                IF  NOTFOUND = 'NO'  GO TO L0203.
       L0202.   MOVE FALSE TO LFOUND.
                GO TO L0101.
       L0203.
                GET REPRESENTATIVE.
                IF  NOT (REPR-NO =  2000  )  GO TO L0202.
                MOVE TRUE  TO LFOUND.
                CALL  RESULT (REPRNAME ).
                GO TO L0101.
$19
```

6）　ＱＴＰを終了させる。

ＱＴＰを終了させるにはＥＮＤコマンドを入力する。

これ以降はＴＳＳコマンドの入力が可能となる。

```
$21 END;
*JIPNET-DDB TERMINATED TIME 15:37:50

READY
```

7）　ＴＳＳのセクションの終了（ＬＯＧＯＦＦコマンド）

```
LOGOFF
```

ＱＴＰをバッチシステムとして動作させる場合には、次のようなジョブ制御文を必要とする。

バッチプログラムは入力バッファ長が異なるほかは、ＴＳＳ下のプログラムと同じである。

```
//BUNSAN  JOB  0106,DIS213,REGION=500K
//JOBLIB DD DSN=JNW0264.BUNSAN.LOAD,UNIT=SYSDA,VOL=SER=USER03,DISP=SHR
// EXEC  PGM=QTP
//HIESR   DD  DSN=HIESR,UNIT=SYSDA,VOL=SER=USER03,DISP=OLD
//HIRSR   DD  DSN=HIRSR,UNIT=SYSDA,VOL=SER=USER03,DISP=OLD
//HIATT   DD  DSN=HIATT,UNIT=SYSDA,VOL=SER=USER03,DISP=OLD
//DMLF  DD  DSN=DBTGDML,UNIT=SYSDA,VOL=SER=USER03,DISP=OLD
//SYSPRINT  DD  SYSOUT=A
//SYSIN  DD  *

QT;
RANGE OF  (R,P,RP) (REPRESENTATIVE,PROJECT,
                    REPR-PROJ);
RETRIEVE INTO RESULT (R.REPRNAME, P.PROJNAME)
    WHERE
          R.REPR-NO  =  RP.REPR-NO  AND  RP.PROJ-NO  =  P.PROJ-NO
      AND R.REPR-NO  =  2000        AND  P.PROJ-NO   =  1999;
GO;

END;

//*
```

LCS問合せの例

図8－125　ＱＴＰのジョブ制御文

8.5.3　ＱＤＰの操作手引

ＴＳＳ環境下でのＱＤＰの実行に必要な処理およびコマンドについて述べる。ここで、ＱＤＰにはＤＩＰＳの処理が含まれているので、バッチシステムとしての単独のＤＩＰＳに関する操作の説明は省略する。

1）　セッションの開始（ＬＯＧＯＮコマンド）

```
LOGON TSS JNW0264/JNW
JNW0264 LOGON IN PROGRESS AT 16:08:33 ON MARCH 28, 1980

READY
```

2）　ＱＤＰで必要とするデータセットを割り当てる。（ＡＬＬＯＣＡＴＥコマンド）

ＱＤＰではＬＩ／ＡＴＴリレーションを使用する。

```
ALLOC DA('LIATT') FILE(LIATT) OLD VOL(USER03)
READY
```

3） ＱＤＰを実行させる。（ＣＡＬＬコマンド）

ロードモジュールの名はＱＤＰＴＳＳである。

```
CALL BUNSAN(QDPTSS)

* JIPNET-DDB STARTED TIME 16:10:50
$01
```

4） 分散記述を入力する。

次に続く、ＲＡＮＧＥ文，ＤＥＦＩＮＥ文が分散記述であることを示すためにＤＩコマンドがある。

分散記述の入力例を次に示す。

```
$01 DI;
$02 RANGE OF (A,B)    ( AAA:1, BBB:1              );
$03 RANGE OF (C,D)    ( CCC:2, DDD:2              );
$04 RANGE OF (E  )    ( EEE:3                     );
$05 RANGE OF (F,G,H)  ( FFF:4, GGG:4, HHH:4       );
$06 DEFINE ESR D1 ( D11, D12, D13)
$07                (D11=B.B1 * 8, D12=C.C1, D13=F.F1 + 2  )
$08   WHERE  A.A1 = B.B1   AND  B.B1 = C.C2   AND  B.B2 = F.F2
$09     AND  A.A2 = 'A';
$10 DEFINE ESR D2 ( D21, D22)
$11                (D21=E.E2, D22=H.H2)
$12   WHERE  E.E1 = H.H2   AND  H.H3 = G.G2+
$13 ;
$14 DEFINE ESR D3 ( D31)
$15                ( D31=D.D1)
$16   WHERE D.D1 GE 500;
```

分散記述については、6.2及び付記1に述べてある。

5） ＧＣＳ問合せを入力する。

ＱＤコマンドを入力し、続くＲＡＮＧＥ文、ＲＥＴＲＩＥＶＥ文がＧＣＳ問合せであることを明確にさせる。

ＧＣＳ問合せの入力例を次に示す。

```
$17 QD;
$18 RANGE OF (D1,D2,D3)  (D1,D2,D3);
$19 RETRIEVE INTO RESULT (R1=D1.D11, R2=D1.D13)
$20    WHERE  D1.D12=B3.D31  AND  D1.D13=D2.D21
$21      AND  D1.D11=D2.D22  AND  D2.D22='AAA';
$22
```

ＧＣＳ問合せについては、4.3及び付記1を参照のこと。

6） ＧＣＳ問合せを処理する。

ＱＴＰと同じようにＧＯコマンドを用いる。次のようにＬＣＳ問合せが出力される。

```
#37 GO;

** LOCAL QUERY TO SITE 02 **
RANGE OF (C    ) IS (CCC);
RANGE OF (D    ) IS (DDD);

RETRIEVE INTO  NRELNAME01 ( C2 = C.C2 )
WHERE C.C1 = D.D1 AND D.D1 GE 500 ;


** LOCAL QUERY TO SITE 04 **
RANGE OF (F    ) IS (FFF);
RANGE OF (G    ) IS (GGG);
RANGE OF (H    ) IS (HHH);

RETRIEVE INTO  NRELNAME02 ( H2 = H.H2 )
WHERE H.H2 = 'AAA' AND H.H3 = G.G2 ;

RETRIEVE INTO  NRELNAME03 ( F1 = F.F1 , F2 = F.F2 )
;


** LOCAL QUERY TO SITE 01 **
RANGE OF (A    ) IS (AAA);
RANGE OF (B    ) IS (BBB);

RETRIEVE INTO  NRELNAME04 ( B1 = B.B1 , B2 = B.B2 )
WHERE A.A1 = B.B1 AND A.A2 = 'A' ;


** LOCAL QUERY TO SITE 03 **
RANGE OF (E    ) IS (EEE);

RETRIEVE INTO  NRELNAME05 ( E2 = E.E2 , E1 = E.E1 )
;


**  INTER-SITE QUERY  **
RANGE OF (NV01) IS (NRELNAME01);
RANGE OF (NV02) IS (NRELNAME02);
RANGE OF (NV03) IS (NRELNAME03);
RANGE OF (NV04) IS (NRELNAME04);
RANGE OF (NV05) IS (NRELNAME05);

RETRIEVE INTO RESULT ( R1 = NV04.B1 * 8 , R2 = NV03.F1 + 2 )
WHERE NV03.F1 + 2 = NV05.E2 AND NV04.B1 * 8 = NV02.H2 AND NV04.B1 = NV01
.C2 AND NV04.B2 = NV03.F2 AND NV05.E1 = NV02.H2 ;
```

7) ＱＤＰを終了させる。

ＥＮＤコマンドを入力する。これ以降はＴＳＳコマンドの入力が再び可能となる。

```
#80 END;
* JIPNET-DDB TERMINATED TIME 16:19:54

READY
```

8) ＴＳセッションの終了（ＬＯＧＯＦＦコマンド）

```
LOGOFF
```

ＱＤＰをバッチシステムとして用いる場合は、次のようなジョブ制御文を必要とする。ＱＴＰと同じように、バッチシステムのプログラムは、ＴＳＳ下のプログラムと入力バッファ長が異なるだけである。

```
//BUNSAN  JOB  0106,DIS213,REGION=500K
//JOBLIB DD DSN=JNW0264.BUNSAN.LOAD,UNIT=SYSDA,VOL=SER=USER03,DISP=SHR
// EXEC   PGM=QDP
//LIATT   DD  DSN=LIATT,UNIT=SYSDA,VOL=SER=USER03,DISP=OLD
//SYSPRINT  DD  SYSOUT=A
//SYSIN  DD  *

DI;
RANGE ( A, B     )  ( AAA:1, BBB:1          );
RANGE ( C, D     )  ( CCC:2, DDD:2          );
RANGE ( E        )  ( EEE:3                 );
RANGE ( F, G, H  )  ( FFF:4, GGG:4, HHH:4   );
DEFINE ESR  D1 ( D11, D12, D13      )
      ( D11 = B.B1 * 8, D12 = C.C1, D13 = F.F1 +2      )
      WHERE
            A.A1 = B.B1    AND  B.B1 = C.C2    AND  B.B2 = F.F2    AND
            A.A2 = "A" ;
DEFINE ESR  D2 ( D21, D22  )
      ( D21 = E.E2, D22 = H.H2 )
      WHERE
            E.E1 = H.H2    AND  H.H3 = G.G2
      ;
DEFINE ESR  D3 ( D31 )
      ( D31 =  D.D1   )
      WHERE
            D.D1 GE 500  ;
QD;
RANGE ( D1, D2, D3 )    ( D1, D2, D3 ) ;
RETRIEVE INTO RESULT (R1=D1.D11, R2=D1.D13)
      WHERE
          D1.D12  =  D3.D31    AND    D1.D13  =  D2.D21    AND
          D1.D11  =  D2.D22    AND    D2.D22  =  "AAA" ;
GO;

END;

/*
//
```

分散記述の例

GCS問合せの例

図８－１２６　ＱＤＰのジョブ制御文

8.6 ま と め

これまで述べてきた4つのシステム、即ちＨＩＰＳ、ＤＩＰＳ、ＱＴＰ、ＱＤＰのインプリメンテーションを終了している。ＨＩＰＳは、ＤＢＴＧモデルからＥ－Ｒモデルに基づいたリレーショナル記述を生成できる。ＤＢＴＧモデルは、現在かなりの制約を置いている。より一般的なデータモデル構造〔48〕を導入するためには、ＨＩＰＳへの入力言語形式の検討が必要となる。ＱＴＰは、局所概念スキーマに基づいたＬＣＳ問合せ（ＱＵＥＬ）からＤＢＴＧ　ＤＭＬから成るＣＯＢＯＬプログラムの手続部を生成する。5.8で述べた様に、完全なＣＯＢＯＬ　ＤＭＬプログラムとするためには、データ部の生成、ＲＥＳＵＬＴ、ＧＮＶサブルーチンの生成等が次の課題としてある。ＱＴＰ内で行なった最適化の有効性も、実際のＤＭＬプログラムを生成し実働させることによって検討していきたい。ＱＴＰとＨＩＰＳは、局所データベースプロセッサ（ＬＤＰ）〔3.3〕を構成する。

ＤＩＰＳは、ＧＳＤＬ〔付記1を参照〕で記述された分散記述（ＤＤ）を入力にして、全体概念スキーマ（ＧＣＳ）と、これとＬＣＳとの対応情報を生成する。この対応情報は分散情報（ＤＩ）である。ＤＩＰＳの主たる機能は、分散記述（ＤＤ）からこれの木構造表現（ＤＤ－ｔｒｅｅ）を生成することである。ＱＤＰは、この分散情報をもとにして、ＧＣＳ問合せからＬＣＳ問合せを生成する。現在問合せの正規化（ＱＮ）と転送スケジューラ（ＴＳ）は、まだインプリメントしていない。

ＱＤＰとＤＩＰＳは、全体データベースプロセッサ（ＧＤＰ）〔3.3〕を構成する。

ＱＴＰとＱＤＰでは、グラフ，木といった種々の構造を持ったデータを扱かわねばならない。我々は、これらをＰＬ／Ｉでインプリメントしたために、多くのプログラミングとデバッギング努力を、これらの構造を操作するため基本機能であるリスト処理機能実現のためにさかねばならなかった。リスト処理機能を実現するためには、木の生成、管理、自由空間領域の管理とちり集め、スタック等の機能をつくらねばならなかった。より多くの使いよさをユーザへ提供するためには、より多くの構造の操作が必要となる。例として、ＱＤＰ、ＱＴＰにおける問合せの正規化がある。これには、複雑なパターン照合機能と変換機能が必要となり、現在のＰＬ／Ｉによるインプリメンテーションは困難である。よってより使いよさが必要とされる時、ＬＩＳＰのような記号処理言語によるインプリメントが現実性をもってくるように思われる。現在多くのシステムは、本質的な使いよさの実現以前に、パフォーマンス向上に多くの努力がむけられている。しかし、今後、多様なユーザへのサービスを考える時、我々も、ＬＩＳＰ言語を我々のインプリメントツールとして本格的に検討していく必要がある。

9　まとめと今後の課題

本報告書では、第2章で述べた分散型データベースシステム問題のなかで、次の点について論じてきた。

1）全体アーキテクチャとしての四層スキーマ構造（ＦＳＳ）概念

2）同種化

3）異種性情報（ＨＩ）

4）問合せ変換

5）統合化

6）分散情報

7）問合せ分割

8）基本システムアーキテクチャ

9.1　全体アーキテクチャ

我々はまず分散型データベースシステム（distributed database system or DDBS）の全体アーキテクチャ（gross architecture）として、4つのスキーマ階層〔図3.2〕概念を導入した。特にこの概念は、既存の独立かつ自立的なデータベースシステムを統合（integrate）して新たな論理的データベースシステムをつくろうとする統合型（bottom−up）分散型データベースシステムにおいて重要である。分散型データベースシステムの統合型設計において、まず考えねばならない点は、統合すべき既存データベースシステムの異種性（heterogeneity）である。この異種性には、大別して2つの観点がある。1つはデータモデルとデータ言語というデータベースシステムの構文的側面の相違である。他の1つは、データベースシステム内にどの様な意味を持ったデータが格納されているかという意味論的側面である。前者は、後者に対する記述系とアクセス系である。このため、異種性の2つの面は独立して考えられる。

ここで我々は、データベースシステムの異種性とは、そのデータモデルとこれに基づいたデータ言語の相違であると再定義する。又、地理的に分散した異なった意味論的側面をもつデータから新たな意味記述をつくることを分散問題と定義する。既存データベースシステムを統合するためには、まず第1にデータベースシステムの異種性、即ちそれの構文的側面の相違、を解決することが必要となる。これは分散型データベースシステム全体に対して共通なデータモデルとアクセス言語（問合せ言語）とを設定し、データベースシステムのスキーマ（これを局所内部スキーマと呼ぶ）をこの共通データモデルで記述することによって解決できる。この過程を同種化（homogenization）と呼び、生成されたスキーマを局所概念スキーマと呼ぶ。しかる後に、共通記述系（共通データモデル）によって表わされた各データベースシステムの意味記述から、新たな論理的データベースシステムの意味記述（これを全体概念スキーマと呼ぶ）を得ることができる。この過程を統合化（integration）と呼

ぶ。我々の四層スキーマ構造は、局所概念スキーマ（ＬＣＳ）層と全体概念スキーマ（ＧＣＳ）層の存在を大きな特色としている。もう１つのスキーマ層は外部スキーマ（ＥＸＳ）層と呼ばれ、全体概念スキーマからアプリケーションに適合するように必要なデータを、適したデータモデルで記述した階層である。全体概念スキーマ層において、分散型データベースシステムは、各データベースシステムの異種性も、分散問題も不可視（invisible） であり、１つの巨大データベースシステムへと仮想化されている。このため外部スキーマ層は、分散型データベース固有の問題でないと言える。よって我々は本報告書では外部スキーマについては検討しなかった。この様に、局所内部スキーマ層、局所概念スキーマ層、全体概念スキーマ層、外部スキーマ層という４層のスキーマ階層を、分散型データベースシステムの基本的スキーマ階層、即ち全体アーキテクチャとした。この四層スキーマ構造（ＦＳＳ）をもとに、以下のことを論じた。

a）各スキーマ層の設計問題

b）各スキーマ層に発せられた問合せの処理問題、即ちアクセス問題

c）２つの隣接するスキーマ層間の対応情報、a）とb）の必要情報

d）分散型データベースシステムのシステムアーキテクチャ

9.2　スキーマ層設計と必要情報

本報告書では、スキーマ層の統合型設計について検討した。検討は、次の２点について行なった。

1）局所内部スキーマ層から局所概念スキーマ層の設計（同種化）と、この過程で除去される情報（異種性情報）

2）局所概念スキーマ層から全体概念スキーマ層の設計（統合化）と、この過程で除去される情報（分散情報）

9.2.1　同種化と異種性情報

第１に局所内部スキーマ層から局所概念スキーマ層の生成、即ち同種化問題と、この両スキーマの対応情報である異種性情報とについて論じた〔第４章〕。ここで局所内部スキーマとは、既存データベースシステム内のデータのなかで分散型データベースシステムとして使用可能なデータの記述である。よってこれは既存データベースシステムのスキーマ又はサブスキーマに対応している。同種化において、我々は分散型データベースシステム全体の共通データモデルとしてＥ－Ｒモデル（entity－relationship model）〔CHEN76〕を、共通問合せ言語としてＱＵＥＬ〔ＨＥＬＤＧ７５、ＹＯＵＳＫ７７〕を採用した。Ｅ－Ｒモデルは、事象（entity） と事象間の関係性（relationship）との概念を最も簡潔な形式でもつとともに、リレーショナルモデルとよく対応できることが、この理由である。前者は、スキーマの設計ツールとして用いられ、後者はそのインプリメンテーションがリレーショナルモデルで出来ることを示している。

同種化の対象としては、既存の代表的な３つのデータモデル、即ちＣＯＤＡＳＹＬ ＤＢＴＧモデル、ＩＭＳモデル、リレーショナルモデル、を考えた。前者の２つのモデルは、モデルの要素（ＤＢＴＧ

モデルのレコード型、ＩＭＳモデルのセグメント型）と要素間の関係性（ＤＢＴＧモデルのセット型、ＩＭＳモデルの階層パス）との２つを基本構造としている。この２つのモデルは、主に後者の関係性によって特徴づけられる。従って、Ｅ－Ｒモデルの事象集合をモデル要素に、関係性集合を要素間の関係性に１対１に対応させることによって、これらの２つのモデルによって記述された局所概念スキーマから、Ｅ－Ｒモデルに基づいた局所概念スキーマを生成できる。又、この要素 ── 事象集合、関係性 ── 関係性集合の対応関係は、異種性情報内に保持される。特に関係性集合は、ＤＢＴＧとＩＭＳの２つのモデルのスキーマレベル（局所内部スキーマレベル）でのアクセスパスを表わしている。よって、局所内部スキーマレベルの異種性は関係性集合を表わす対応情報内にその多くがふくまれることになる。逆に言えば、局所内部スキーマ要素の関係性と、局所概念スキーマの関係性集合との対応情報として、その異種性を表わすだけでよいことになる。このことは、ＤＢＴＧとＩＭＳ以外の関係性を持つデータモデルへの拡張性を示している。

特に、ＤＢＴＧモデルについては、対応する異種性情報のスキーマを示した。異種性情報は、局所概念スキーマ層で管理される。これは、どの異種性情報も統一的にアクセスされるためである。従がって異種性情報は、リレーショナルモデルとして管理され、我々はそのリレーションスキーマを示した。ＤＢＴＧデータベースシステムから局所概念スキーマ層を生成するシステムとして、異種性情報処理システム（ＨＩＰＳ）をインプリメント〔 8.1を参照〕し、その異種性情報を生成した。

異種性情報は、次の３つの要素から成っている。

a）局所内部スキーマと局所概念スキーマのモデル要素の対応

b）局所内部スキーマ層のパフォーマンス情報

c）局所内部スキーマ層のアクセス言語情報

これらは全てリレーショナルモデルで管理されている。現在、ＤＢＴＧモデルにおいて、レコード型のキーとしてはＣＡＬＣのみを考え、DUPLICATE IS NOT ALLOWED（DNA）であると仮定し、更にセットもAUTOMATICでMANDATORYと仮定している。今後より一般的なＤＢＴＧデータ構造を異種性情報へ反映させる必要がある。ＤＢＴＧモデルは、今後４～５年は、既存データベースシステムの主流となっていくと考えられ、この意味でもより実用的にしていくことは重要である。

しかしリレーショナルモデルではこの様には対応させられない。何故ならリレーショナルモデルでは、リレーション間の関係性を示すモデル構成要素はない。唯一リレーショナル演算（e.g.結合）によって、リレーション内の値を通して他のリレーションと関係性をもたすことができるだけである。又、意味的観点からもリレーションは、これが事象を表わすのか関係性を表わすのか明らかでない。更にあるリレーションは事象としての内容とともに関係性としての情報も持っている。よって我々は一般に、正規化されたリレーション集合から、事象が何であり関係性が何であるかを自動的に導き出せないと考えている。リレーショナルモデルにおけるリレーションとは何かが明らかになって初めて事象と関係性とを分離できるように思う。従って我々の扱おうとするリレーショナルモデルのスキーマは、次のようなリレーションから構成されているとした。

a）事象を表わすリレーション　（これは第３正規化されている）

b）これらの事象間の関係性を表わすリレーション（これは第４正規化されている）

b）のリレーションは、これが関係づける２つの事象を表わすリレーションのキー属性のセットをそのリレーションのキーとしている。このことは、リレーショナルスキーマが局所概念スキーマ層でのスキーマのリレーションによる表現（我々はリレーショナル記述（ＲＤ）と呼んでいる）そのものであることを意味している。

9.2.2 統合化と分散情報

第２の設計問題として、局所概念スキーマ層から全体概念スキーマ層の生成、即ち統合化（integration）と、この両スキーマ層間の対応情報としての分散情報（ＤＩ）とについて論じた〔第６章〕。局所概念スキーマは、Ｅ－Ｒモデルという共通モデルによる各データベースシステムのデータの意味を共通記述したものである。更に、これは実際は、Ｅ－Ｒモデルに基づいたリレーショナル記述（ＲＤ）によって表わされている。次の問題は（共通記述された）異なったデータの意味構造をもったデータベースシステムから、新たに１つの意味構造を持った論理的データベースシステムをつくることである。この設計過程は困難なものであり、一般に全て可能であるとは言えない。我々は、全体管理者がマニュアル的に全体概念スキーマを各サイトの局所概念スキーマから生成するとする。全体概念スキーマは、局所概念スキーマと同じくリレーショナルモデルによって記述され、アクセスされる。この全体概念スキーマ層のリレーションは、構文的には局所概念スキーマのリレーションから視野（view）を定義するのと等価である。よって我々は、全体概念スキーマ（ＧＣＳ）の構文的記述、即ちＧＣＳリレーションの定義用言語ＧＳＤＬを提案した。これは関係計算言語（relational calculas language）ＱＵＥＬをリレーションの和（union）も取れるように拡張したものである。全体概念スキーマの意味記述と設計方法は今後の重要な課題である。

局所概念スキーマ（ＬＣＳ）層と全体概念スキーマ（ＧＣＳ）層との対応情報としての分散情報（ＤＩ）は、異種性情報と同じく局所概念スキーマ層で管理される。よって分散情報はリレーショナルモデルとして管理される。

両スキーマの要素の対応情報は、関係計算形式の論理式として格納される。我々の分散情報は、ＬＣＳリレーションについてのパフォーマンス情報を持たない。パフォーマンス情報がスキーマ情報と比して動的な性質を持ち、かつ異種性情報と共有されているからである。動的情報の冗長保持は深刻なＣＣ(concurrency and consistency)問題をもたらすことになる。この我々の主張は統合化の逆過程である問合せ分割のアルゴリズムを決定している。

9.3 アクセス機能

これまでに設計問題、即ち同種化と異種性情報、統合化と分散情報について述べた。次に各スキーマ層において出された問合せがどのように処理されるかというアクセス問題について考えよう。

アクセス問題としては、次の２つを検討し、インプリメントを行なった。

1）問合せ変換〔第5章及び8.2節〕

2）問合せ分割〔第7章及び8.4節〕

9.3.1 問合せ変換

問合せ変換とは、局所概念スキーマ（ＬＣＳ）層の問合せ（ＬＣＳ問合せ）が局所内部スキーマ（ＬＩＳ）層のアクセス言語から成るプログラムを異種性情報を用いながら生成することである。問合せ変換では、次の２点を行なう必要がある。

a） ＬＣＳ問合せの参照する局所概念スキーマ要素を対応する局所内部スキーマ要素への変換

b） ＱＵＥＬで記述された非手続的なＬＣＳ問合せから、局所概念スキーマ層で有効なアクセス手順の生成。

特に、本報告書では、ＱＵＥＬ問合せから、ＤＢＴＧ　ＤＭＬプログラムの生成について検討した。a）は構造変換と呼ばれ、問合せのグラフ表現を用いて、その変形を通してＤＢＴＧモデル要素から成る問合せのグラフ表現を最終的に得る。ここではＬＣＳ問合せ内に表われない隠れ構造のパターンを明らかにした。これらの隠れ構造は、異種性情報（ＨＩ）をサーチすることによって明らかにされる。最終的に得られるグラフをＤＢＴＧ問合せグラフ（ＤＱＧ）と呼ぶ。ＤＢＴＧ問合せグラフのノード（ＤＱＧノードと呼ぶ）はＤＢＴＧモデルのレコード型を表わし、アーク（ＤＱＧアークと呼ぶ）はセット型を示している。さらに、ＬＣＳ問合せ内に現われる制限条件、結果属性情報も、この中に表わされている。

b）では問合せグラフ表現（ＤＢＴＧ問合せグラフ）から、アクセスのシーケンスを表わすアクセス木の生成について論じた。アクセス木のノードはＤＱＧノードに、枝はＤＱＧアークに対応する。ＤＱＧがグラフであることから、ＤＱＧアークに対してユニークなアクセス木枝を設けると、ＤＱＧに対して複数のアクセス木ノードが存在し得る。この様な冗長なアクセス木ノードを合流ノード（confluent node）と呼ぶ。１つのＤＱＧノードに対して複数の合流ノードがある時、１つのアクセスパス内で、これらの合流ノードは同一のオカーランスを持たねばならないという制限を持っている。これを実現するには、各合流ノードごとに中間結果リレーションを生成し、これらのリレーション間の結合（ｊｏｉｎ）の様な集合演算を必要としてしまう。中間結果リレーションの数の増大は、これらの記憶領域の管理上の問題をもたらし、さらにこれらの間に集合演算機能を設けねばならなくなる。我々の提案する縦型アクセス木生成アルゴリズム（ＤＦＡ）は、グラフを縦型にサーチしながら木を生成するものであり、次の様な長所を持つ。

i） ただ１つの結果リレーション以外の中間結果を必要としない。

ii） アクセスされるオカーランス数を比較的少なくする。

iii） 簡単である。

i）は、我々のＤＦＡの最大の長所である。i）の点は、必要なものだけを結果リレーションとして

格納するので、中間結果の量も少なくし得る。我々は、局所概念スキーマ層でのアクセス手順を得る上で最も重要な点は、中間結果数を最少とすることであると確信している。

アクセス木の各ノードと枝の対（これをアクセス単位と呼ぶ）に対応して、ＤＭＬプログラムのブロックが生成される。ＤＭＬブロックは、１０個のパターンから成っている。アクセス単位は、パターンと照合され、対応したＤＭＬブロックが生成され、これらの集合が、ＤＢＴＧ　ＤＭＬプログラムの手続部をつくり出す。これらのＤＭＬブロックは、異種性情報のアクセス言語情報（HI／DML）となる。我々の生成したＤＭＬプログラムのいくつかの例を5.5.4に示してある。

この様にして、我々の問合せ変換システムは、局所概念スキーマ（ＬＣＳ）に基づいた非手続的なＱＵＥＬ問合せから、ＤＢＴＧモデルの局所内部スキーマ（ＬＩＳ）に基づいたＣＯＢＯＬ　ＤＭＬプログラムの手続部を生成できる。更に、この生成された手続は、局所内部スキーマ層において適当なものである。

問合せ変換システム、同種化システムと異種性情報とは、分散型データベースシステムばかりでなく、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としても使用できる。さらに我々の手法は、他の関係性構造を持つモデル、e. g. ＩＭＳモデル、への拡張も容易である。

次の問題として、ＣＯＢＯＬ　ＤＭＬプログラムのデータ部を生成し、実際のＤＢＴＧデータベースシステム上で実働させることがある。

我々の問合せ変換への入力、即ちＬＣＳ問合せは、等価結合式の論理積と限定している。各等価結合述語は、セット型によるレコード型のリンクに対応している。不等価述語を許す場合には、セット型を介したアクセスではなく、２つの（親と子）レコード型間の集合演算が必要となる。又、ＬＣＳ問合せはＣＯＵＮＴ、ＳＵＭといった aggregate 関数も持たないと仮定している。これらの関数は、ＬＣＳ問合せを実行以前に、これとは独立に１つのＬＣＳ問合せとして処理されるものとしている。実際には aggregate 関数に対応した問合せ結果の集合演算処理が必要となる。この様に今後、ある程度の集合演算処理機能の実現が必要となろう。しかし我々は、多くの問合せのパターンは、等価結合式の論理積の形式をもっており、現在の問合せ処理機能でも十分ユーザの要求に答えられると考えている。

9.3.2　問合せ分割

問合せ分割（ query　decomposition）は、全体概念スキーマ層に出された問合せ（GCS問合せと呼ぶ）から局所概念スキーマ層の問合せ（ＬＣＳ問合せと呼ぶ）を生成することである。この時必要となる両スキーマ層の対応情報は、ネットワークデータディレクトリ内の分散情報（ＤＩ）内に格納されている。問合せ分割では、次の２つのことを行なわねばならない。

a)　ＧＣＳ問合せの参照する全体概念スキーマ要素から、対応する局所概念スキーマ要素への分割

b)　サイト間にまたがった問合せの処理

a）は、分散情報（ＤＩ）内には、両スキーマ層の対応情報が述語論理形式で格納されているので、

問合せ変形（query modificatien） 手法を用いて処理できる。これによって生成された問合せは、ＧＣＳ問合せが参照する全ての局所概念スキーマ層のリレーション（ＬＣＳリレーション）を参照している（この問合せを全体ＬＣＳ問合せ（ＧＬＱ）と呼ぶ）。

次の問題は、複数サイトにまたがった全体ＬＣＳ問合せ（ＧＬＱ）の処理を行なうことである。この処理を行なうためには、サイト間でのデータの転送が必要となる。問合せの処理においてサイト間でのデータの転送コストが支配的である。従がって、データの転送方法の決定は、問合せ分割処理の最も重要な部分を占める。転送方法は、次の３点を目標としてなされる。

ｉ）最少通信コスト、 i. e. 最少全処理時間

ii）最少応答時間

iii）必要情報の最少化と静的化

我々の転送方法決定アルゴリズム（これを、転送スケジューリングアルゴリズム（ＴＳＡ）と呼ぶ）は、特に、iii）の目標達成を目指している。我々のＴＳＡは、各サイドでの処理状態を集中的にモニタリングしながら、動的に転送方法を決めていくものである。実行前に全ての転送スケジューリングを決定してしまう従来の手法に対して、動的な決定手法を用いたのは、次の理由による。

ｉ）従来の手法は、選択度（selectivity）、カーディナリティ（cardinality）といった統計情報に依っている。選択度は、ある属性のもつユニークな値の数を、これの属するリレーションのカーディナリティで割ったものとして定義されている。我々は、これが現実によく適合するか疑問である。

ii）カーディナリティ、選択度といったパフォーマンス情報は、比較的短かいライフサイクルを持つので、これらの情報をコンシステントに保つことは、大きなＣＣ制御（concurrency and consistency control）問題をもたらす。

１つの転送が終了し生成された中間結果についてのパフォーマンス情報は、ＡＣＫメッセージにのせられて帰ってくる。この情報をもとにして、次の可能な転送のなかで、転送コストが最も安いものを決める。この時ある閾値をもうけ、転送コストがこれ以下のものは並行して転送させる。問合せは結合問合せグラフ（ＪＱＧ）と呼ばれるグラフ表現によって管理され、１つの転送はこのグラフの縮退をもたらす。最終的に１つのノードのグラフになるまで上述したことをくり返す。

転送スケジューリングシステム（転送スケジューラ）は、まだインプリメントしていない。しかし、分散型データベースシステム、とりわけ統合型のシステムにおいては、実行前に全ての転送方法を決めてしまうことは困難であると考える。何故なら、各データベースシステムは独立に構築されてきたものであり、これに対しての結合演算の結果リレーションの大きさを見積もることは不可能だからである。又、分散情報の管理上も、動的なパフォーマンス情報をこれにもたせることは望ましくない。これらのことより、問合せのサイト間処理は実用上、動的な転送方法決定が必要であると考えている。これは我々の１つの仮設である。今後、我々のコンピュータネットワークＪＩＰＮＥＴ上での実験を通して、我々の仮設の検証を行なっていきたい。

9.4 システム アーキテクチャ

これまで述べてきた全体アーキテクチャ、スキーマ層設計法、アクセス方法を基にした、分散型データベースシステムのシステムアーキテクチャ（これを、我々は統一アーキテクチャ（unified architecture）と呼ぶ）を提案した。統一アーキテクチャ〔図3.9〕は、次の2つから成る。

1）トランスポートネットワーク

2）ネットワークサイト

トランスポートネットワークは、2つのサイト間での基本的通信手段を提供する。

ネットワークサイトは、1つまたは複数のデータベースシステムから成っているホストコンピュータシステムと考えられる。サイトは、次のものから構成されている。

a）ネットワークアクセスシステム　（network accessor）

b）全体データベースプロセッサ　（global database processor）

c）局所データベースプロセッサ　（local database processor）

d）データベースシステム

e）ユーザインタフェース　（user friendly interface）

ネットワークアクセスシステムは、自分のサイト又は他のサイトの全体データベースプロセッサ（以降ＧＤＰと訳す）と局所データベースプロセッサ（以降ＬＤＰと訳す）との間の通信をネットワークプロトコルを用いて行なう。全体データベースプロセッサ（ＧＤＰ）は、問合せ分割、統合化、及び分散情報の管理を行ない、各サイトに1つづつある。

局所データベースプロセッサ（ＬＤＰ）は各データベースシステムに対して1つあり、問合せ変換、同種化、異種性情報の管理を行なう。ＬＤＰは更にサイト間処理のための作業領域（ＷＳ）を持っている〔図7.23〕。ＷＳは、ＬＤＰがリレーショナルデータベースシステムとして直接に管理する。ＷＳは、他のサイトから転送されたり、そのサイトで生成された中間結果としてのリレーションを格納するためにある。

我々が実現した問合せ変換システム（ＱＴＰ）〔第5章、第8.2節〕、同種化システム（異種性情報生成システム（ＨＩＰＳ））〔第4章、第8.1節〕とは、局所データベースプロセッサ（ＬＤＰ）の構成モデュールとなり得る。今後サイト間処理のための作業領域管理システム（WS manager or ＷＳＭ）の実現が必要となる。ＷＳＭ実現の主たる問題は、リレーショナルデータベースシステムとしてのＷＳとＪＯＩＮモデュールとをいかに容易につくるかである。サイト間結合（join）処理は、一方のリレーションが転送されてきた後に行なえる。このため我々は、一方のリレーションの受信中に、そのサイト内（ＷＳ内）にある他のリレーションをソートし、受信中のリレーションも受信しながらソートし、しかる後に結合を行なうことを考えている。この結合演算は容易であり、結合後の射影（projection）において冗長（duplicate）組の除去も容易に行なえる。

問合せ分割システム（ＱＤＰ）〔第7章、第8.4節〕と統合化システム（分散情報生成システム（ＤＩＰＳ））〔第6章、第8.3節〕とは、全体データベースプロセッサ（ＧＤＰ）を構成する。問

合せ分割システムの残された部分は、転送スケジューラ（transmission scheduler）である。これは、ネットワークプロトコルを管理するネットワークアクセスシステム（network accessor or ＮＡ）とともに検討していく必要がある。

ネットワークアクセスシステム（ＮＡ）、全体データベースプロセッサ（ＧＤＰ）、局所データベースプロセッサ（ＬＤＰ）といった分散型データベース管理システム（ＤＤＢＭＳ）を、実際のシステム（ホストコンピュータ、フロントエンドとしてのミニコンピュータ）にどのように実現していくかは今後の検討項目である。

9.5　我々の成果と残された問題点

以上みてきた様に、我々は本プロジェクトにおける成果を、次のようにまとめられる。

1）分散型データベースシステムの全体アーキテクチャとしての四層スキーマ構造（ＦＳＳ）概念を確立した。

2）四層スキーマ構造に基づいたスキーマ設計法と必要情報として次のものを明らかにし、実現した。

a）局所内部スキーマから局所概念スキーマの生成（同種化）と異種性情報（ＨＩ）、

b）局所概念スキーマから全体概念スキーマの生成（統合化）と分散情報（Ｄ１）。

3）四層スキーマ構造に基づいたアクセス方法として次のものを明らかにし、実現した。

a）局所概念スキーマ層の問合せ（ＬＣＳ問合せ）から局所内部スキーマ層のアクセス言語（ＬＩＳ　ＤＭＬ）によるプログラムの生成（問合せ変換）。

b）全体概念スキーマ層の問合せ（ＧＣＳ問合せ）から局所概念スキーマ層の問合せ（ＬＣＳ問合せの生成（問合せ分割）。

a)では特に、ＬＩＳ　ＤＭＬとしてＤＢＴＧ　ＤＭＬを設定した場合を検討した。

これらは、分散型データベースシステムを考える上で、最も基本となる点である。よって我々は、本プロジェクトにおいて、分散型データベースシステム、特に統合型（bottom－up）システムの基本機能を実現し得たと考えている。1）の四層スキーマ構造（ＦＳＳ）と、3）のアクセス機能とは、分割型（top-down）分散型データベースシステムに対しても適用可能である。これらの基本機能のうえに、本格的分散型データベースシステムを実現していくためには、第３章で述べた様に更に次の点を検討していく必要がある。

1）制御機能と通信プロトコル

2）更新アクセス機能

3）全体概念スキーマ層から外部スキーマ層の生成（特殊化（specialization）と外部スキーマ層の問合せから全体概念スキーマ層の問合せ生成（一般化（generalization））

4）アプリケーション

第１の問題は、分散型データベースシステムの制御機能である。制御機能としては、次のものがある。

a) 同時実行制御

b) 一致性制御

c) 障害回復

ここで、我々は各データベースシステム（局所内部スキーマ層）は、それ自身内のデータの完全な制御機能を持っていると仮定する。よって、我々の考えるべき制御機能は、各データベースシステム間にまたがった一致性（consistency）とインテグリティ（integrity）を保持し、障害からの回復を行なうことである。更にこれらの制御機能は、分散型データベースシステムの局所概念スキーマ層レベルに存在しているとする。何故なら、この層において各データベースシステムのスキーマがサイト単位に共通モデルによって表わされているからである。

統合型分散型データベースシステムでは、あまりあり得ぬが、データが複数サイトに冗長にコピーされている時、これらの冗長コピーに対する更新の同期を保ち、一致性を保障する問題は重要である。この問題は、一般に全てのコピーをロックすることによって解かれる。解決されるべき問題は、ロックし、更新し、ロックをはずすための通信オーバヘッドを、いかに少なくするかである。現在までに多くの提案が、この問題に対してなされてきている。この中で、SDD－1〔ROTHJ77〕における〔BERNP78〕等による手法は、各アプリケーションごとに、どの程度までロックする必要があるかを明らかにして、必要な程度だけロックするものである。この点は、制御機能を考える上で重要なポイントである。

分散型データベースシステムに対する更新の制御機能に加えて重要な問題は、上位のスキーマ層における更新要求を下位の層における更新への完全なマッピングを行なうことである。1つは、全体概念スキーマ層における更新を、局所概念スキーマ層の更新へ変遷（transfer）する問題である。これは、リレーショナルモデルにおける視野の更新問題〔CHAMD76，DAYAU78，PAOLP77〕と等価である。更に統合型分散型データベースシステムでは、全体概念スキーマ層のリレーションはその外延として未定義値（null value）を含んでいる。この値に対する更新をどの様に局所概念層へ伝えるかは問題である。

もう1つは、局所概念スキーマ層の更新要求を、局所内部スキーマ層に変遷する問題である。局所内部スキーマ層におけるデータモデルのインテグリティ条件が、局所概念スキーマ層においてどれだけ完全に表わされているかに依っている。このインテグリティ条件は、局所内部スキーマ層におけるモデル要素間の関数従属性、即ち1：n，n：1，n：m，1：1関係性を、どれだけ局所概念スキーマ層で表わせるかにも依っている。我々は、CODASYL DBTGモデルを局所内部スキーマ層として、これに次のように仮定を設けている。

a) レコード型のキー項目はDNA（DUPLICATE IS NOT ALLOWED）

b) セット型も DNA

c) セット型のメンバーシップクラスはAUTOMATICでMANDATORY

この仮定の範囲では、我々の局所概念スキーマは、局所内部スキーマのインテグリティ条件を反映している。

今後、これ以外のより一般的なＤＢＴＧモデル構造につても検討が必要である。

データモデルによって表わされるインテグリティ条件の他の条件、セキュリティ／プロテクション条件についても各スキーマ層間での伝播について検討していく必要がある。これらは、今後重要なテーマとなろう。

我々は、本プロジェクトにおいて、全体概念スキーマ層までのスキーマ層設計問題とアクセス機能問題とについて検討してきた。何故なら、この層までが分散型データベースシステム固有の問題だからである。しかし、実際の分散型データベースシステムの利用（アプリケーション）を考える上では、１つの全体概念スキーマとアクセス言語ＱＵＥＬとをもつ巨大データベースシステムのうえに、各アプリケーションに適したデータ記述である外部スキーマ層を考える必要がある。外部スキーマ層では次の点を検討する必要がある。

a）アプリケーションに適したデータモデル

b）アプリケーションに適したアクセス言語

c）ユーザへのインタフェース（端末、対話手法）

外部スキーマ層で重要な点は、ユーザの多くはコンピュータについての学習意欲も持たない一般ユーザが大半となるということである。データモデルでは、従来の１次元的データ記述から、２次元的表示も必要となろう。又、対話を通して、段々とデータ記述を明らかにしていく方法も考えられる。

アクセス言語としては、リレーショナルモデルにおける関係計算言語の様な非手続的問合せ言語よりは、概念的なスキーマ上を航海（navigation）していく様な言語のほうが、一般ユーザには望ましい。又、自然言語的な問合せ言語も有効であろう。

最後に重要な点は、ユーザの用いるインタフェース端末である。これはコンパクトであるとともに、従来の端末に加えて、種々の入出力のメディア変換機能を備えている。例えば、音声入力、手書き文字入力装置がこれである。ユーザの使いやさという点から、人間工学的に、キーの配列、ディスプレィの色等の検討が必要となろう。

３年間にわたる本プロジェクトの成果は、本節の最初でのべた様に、分散型データベースシステムの基本スキーマ階層を明らかにし、これに基づいたスキーマ層設計方法とアクセス機能とを確立したことである。分散型データベースシステム問題は、本節で述べた様に、まだ多くの未解決の問題を多く持っている。これらの多く、特に制御問題は、実際の分散型データベースシステムの設計開発を通して検討していく必要があるだろう。

分散型データベースシステムが、コンピュータネットワーク技術とデータベースシステム技術との結合点にあることから、今後最も主要なコンピュータ技術となるだろう。

参 照 文 献

[ABRLJ74]

Abrial, J. R.,

"Data Semantics",

Proc. of the IFIP TC-2 Working Conf. on DBMS,

Cargese, Corsia, France,

April 1974, pp.1-60

[ADIBM 78b]

Adiba, M., and Euzet, C.,

"A Distributed Data Base System Using Logical Relational Machines,"

Proc. 4th International Conf. on VLDB,

Berlin, Sept. 1978, pp.450-461

[AHOA 74]

Aho, A.V., Hopcroft, J. E., and Ullman, J. D.,

"The Design and Analysis of Computer Algorithms,"

Addison-Wesley, 1974

[ANSIX 75]

"Interim Report of the Study Group on Data Management Systems,"

ANSI/X3/SPARC DBMS Study Group, Report 75-02-08, Feb. 1975

[ASTRM 76]

Astrahan, M. M., et al., "System R: Relational Approach to Data Base Management," ACM TODS, Vol. 1, No. 2, June 1976, pp.97-137

[ASTRM 79]

Astrahan, M. M., Blasgen, M. W., Chamberlin, D. D., Gray, J. N., King, W. F., et al., "System R: A Relational Data Base Management System," IEEE Computer, Vol. 12, No. 5, May 1979, pp.42-48

[BACHC 78]

Bachman, C. W., "Provisional Model of Open System Architecture," Proc. of the 3rd Barkeley Workshop on Distributed Data Management and Computer Networks, San Francisco, Aug. 1978, pp. 1-18

[BACKJ78]

Backus, J., "Can Programming Be Liberated from the von Newmann Style? A Functional Style and Its Algebra of Programs," CACM, Vol. 21, No. 8, Aug. 1978, pp. 613-641

[BANEJ79]

Banerjee, J., Hsiao, D. K., and Kannan, K.,

"DBC - A Database Computer for Very Large Databases,"

IEEE Trans. on Computers, Vol. C-28, No. 6, June 1979, pp. 414-429

[BAYER72]

Bayer, R. and McCreight, E. M.,

"Organization and Maintenance of Large Ordered Indexes," Acta

Informatiza, Vol. 1, No. 4, Dec. 1972, pp. 290-306

[BERNP78]

Bernstein, P. A., Rothnie, J. B., Goodman, N., and Paradimitrion,

C. A., "The Concurrency Control Mechanism of SDD-1: A System for

Distributed Databases (The Fully Redundant Case)," IEEE Trans. on

SE, Vol. SE-4, No. 3, May 1978, pp. 154-168

[BERNP79]

Bernstein, P. A. and Goodman, N.,

"Approaches to Concurrency Control in Distributed Data Base Systems,"

AFIPS Conf. Proc., Vol. 48, New York, June 1979, pp. 813-820

[CARDA79]

Cardenas, A. F., "Database Management Systems," Allyn and Bacon, Inc.,

Boston, MA., 1979

[CHAMD75]

Chamberlin, D. D., Gray, J. N., and Traiger, I. L., "Views, Authorization, and Locking in a Relational Data Base System," AFIPS Conf. Proc., Vol. 44, May 1975, pp.425-430

[CHAMD 76]

Chamberlin, D. D., Astrahan, M. M., et al., "SEQUEL 2: A Unified Approach to Data Definition, Manipulation and Control," IBM Journal of Research and Development, Vol. 20, No. 6, Nov. 1976

[CHENP76]

Chen, P.P.S., "The Entity-Relationship Model - Toward a Unified View of Data," ACM TODS, Vol. 1, No. 1, March 1976, pp.9-36

[CHUWU73]

Chu, W. W., "Optimal File Allocation in a Computer Network," Computer Communications Networks (N. Abramson and F. Juo, eds.), Prentize Hall, 1973, pp.82-94

[CHUWU79]

Wesley W. Chu and Hurley, P., "A Model for Optimal Query Processing for Distributed Data Bases," IEEE Compcon 79 Spring, Feb. 1979, pp.116-122

[CODAS73]

CODASYL, "CODASYL Data Description Language Journal of Development," June 1973, NBS Handbook 113 (1974)

[CODAS 78b]

CODASYL DDL Committee, "Report of the CODASYL Data Description Language Committee," Information Systems, Vol. 3, No. 4, 1978, pp.247-320

[CODDE 70]

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

[CODDE 78]

Codd, E. F., "How About Recently? (English Dialog with Relational Data Bases Using Rendezuous Version 1)," Databases: Improving Usability and Responsiveness (Shneiderman, B. ed.), Academic Press, 1978, pp.3-28

[DATEC77]

Date, C. J., "An Introduction to Data Base System," (Second Edition), Addison-Wesley, June 1977, p.536

[DAYAU 78]

Dayal, U. and Bernstein, P. A., "On the Updatability of Relational Views," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.368-377

[DEWID 79a]

De Witt, D. J., "DIRECT - A Multiprocessor Organization for Supporting Relational Database Management Systems," IEEE Trans. on Computers, Vol. C-28, No. 6, June 1979, pp.395-406

[ELMAR 79]

El-Masri, R., and Wiederhold, G., "DATA MODEL INTEGRATION USING THE STRUCTURAL MODEL," Proc. of the ACM. SIGMOD International Conf. on Management of Data, Boston, MA., May 1979, pp.191-202

[FUTOI78]

Futó, I., Darvas, F., and Szeredi, P.,
"The Application of PROLOG to the Development of QA and DBM System,"
Logic and Data Bases (Gallaire, H. and Minker, J. eds.),
Plenum Press, 1978, pp.347-376

[GARDG 79]

Gardarin, G. and Jouve, M., "The Execution Kernel of a Distributed Data Base Management System," Proc. of the IFIP TC-2 Working Conf. on Data Base Architecture, Venice, May 1979, pp.3-21

[HAERT 78]

Haerder, T., "Implementing a Generalized Access Path Structure for a Relational Database System," ACM TODS, Vol. 3, No. 3, Sept. 1978, pp.285-298

[HELDG 75]

Held, G. D., Stonebraker, M. R., and Wong, E., "INGRES - A Relational Data Base System," AFIPS Conf. Proc., May 1976, pp.409-416

[HEVNA 78a]

Hevner, A. R. and Yao, S. B., "Query Processing on a Distributed Database," Proc. 3rd Berbeley Workshop on Distributed Data Management and Computer Networks, San Francisco, California, Aug. 1978, pp.91-107

[HEVNA 78b]

Hevner, A. R. and Yao, S. B., "Optimization of Data Access in Distributed Systems," TR281, Computer Science Dept., Purdue University, July 1978

[HOUSB79]

Housel, B. C., Waddle, V., and Yao, S. B., "The Functional Dependency Model for Logical Database Design," Proc. of the 5th International Conf. on VLDB, Rio De Jeneiro, Brazil, Oct. 1979

[KIMBS79]

Kimbleton S. R., Wang, P. S. - C., and Fong, E. N., "XNDM: An Experimental Network Data Manager," NBS, 1979

[KNUTD73]

Knuth, D. E., "The Art of Computer Programming-Fundamental Algorithms (2nd ed.)," Addison Weslery, 1973

[KOWAR74]

Kowalski, R. A., "Predicate Logic As Programming Languages," Information Processing 74, 1974, pp.569-574

[LIENY78]

Lien, Y. E. and Ying, J. H., "Design of a Distributed Entry-relationship Database System," Proc. of IEEE Compsac 78, Chicago, Ill., Nov. 1978, pp.277-282

[MAHMS79]

Mahmoud, S. A., Riordon, J. S., and Toth, K. C., "Distributed Database Partitioning and Query Processing," IFIP TC-2 Working Conf. on Data Base Architecture, Venice, May 1979, pp.35-50

[MORGH77]

Morgan, H. L., and Levin, K. D., "Optimal Program and Data Locations in Computer Networks," CACM, Vol. 20, No. 5, May 1977, pp.315-322

[MUVZR79]

Munz, R., "Gross Architecture of the Distributed Database System VDN," Proc. of the IFIP Working Conf. on Database Architecture, Venice, May 1979, pp.23-34

[MUROK80]

Muroi, K. and Tanaka, Y., "CODASYL DBMS のリレーショナルインタフェース," 北大 工, 1980.

[MYLOJ70]

Mylopoulos, J., Borgida, A., Cohen, P., Roussopoulos, N., Tsotsos, J., and Wong, H., "TORUS: A Step Towards Bridging the Gap Between Data Bases and the Casual User," Information Systems, Vol. 2, No. 2, 1976, pp.49-64

[NEUHE77a]

Neuhold, E. J., and Biller, H., "POREL: A Distributed Data Base on An Inhomozeneous Computer Network," Proc. 3rd International Conf. on VLDB, Tokyo, Japan, Oct. 1977, pp.380-395

[OZKAE77]

Ozkarahan, E. A., Schuster, S. A., and Sevcik, K. C., "Performance Evaluation of a Relational Associative Processor," ACM TODS, Vol. 2, No. 2, June 1977, pp.175-195

[PAOL77]

Paolini, P. and Pelagatti, G., "Formal Definition of Mappings in a Data Base," Proc. ACM SIGMOD International Conf. on Management of Data, Toronto, Canada, Aug. 1977, pp.40-46

[POUZL77]

Pouzin, L., "Packet Networks - Issues and Choices," Proc. IFIP Congress 77, Toronto, Aug. 1977, pp.515-521

[ROBEL 70]

Roberts, L. G. and Wessler, B. D., "Computer Network Development to Achieve Resource Sharing," AFIPS Conf. Proc. (FJZZ), 1970, pp.543-549

[ROTHJ77]

Rothnie, J. B. and Goodman, N., "An Overview of the Preliminary Design of SDD-1: A System for Distributed Databases," Proc. of 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, UC Berkeley California, May 1977, pp.39-57

[SAGAD77]

Sagalowicz, D., "IDA: An Intelligent Data Access Program," Proc. 3rd International Conf. on VLDB, Tokyo, Oct. 1977, pp.293-302

[SELIP79]

Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., and Price, T. G., "Access Path Selection in a Relational Database Management System," Proc. of the ACM SIGMOD, Boston, May 1979, pp.23-34

[STONM76]

Stonebraker, M., Wong, E., Kreps, P., and Held, G., "The Design and Implementation of INGRES," ACM Trans. on Database Systems, Vol. 1, No. 3, Sept. 1976, pp.189-222

[SIGMD74]

ACM SIGMOD, "Data Models: Data-Structure-Set versus Relational," Workshop on Data Description Access and Control, May 1974

[STONM77a]

Stonebraker, M., and Neuhold, E., "A Distributed Data Base Version of INGRES," Proc. on 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, UC. Berkeley, Calif., May 1977, pp.19-36

[SUS 79a]

Su, S.Y.W., "Cellular-Logic-Devices: Concepts and A-plications," IEEE Computer, Vol. 12, No. 3, March 1979, pp.11-25

[TAKIM78]

Takizawa, M., Hamanaka, E., and Ito, T., "Resource Integration and Data Sharing on Heterogeneous Resource Sharing System," Proc. ICCC '78, Kyoto, Japan, Sept. 1978, pp.253-258

[TAKIM79a]

Takizawa, M. and Hamanaka, E., "The Four-Schema Concept as the Gross-Architecture of Distributed Databases and Heterogeneity Problems," Journal of Information Processing, IPSJ, Vol. 2, No. 3 Nov. 1979, pp.134-142

[TAKIM79b]

Takizawa, M. and Hamanaka, E., "Query Translation in Distributed Databases," JIPDEC TR79/04, JIPDEC, Oct. 1979 (to appear at IFIP80 Congress, Tokyo, Oct. 1980)

[TAKIM80a]

Takizawa, M. and Hamanaka, E.,

"分散型データベースにおける問合せ変換,"

Proc. of the 21st Programming Symp., Hakone, Japan, Jan. 1980

[TAKIM80b]

Takizawa, M., "Distribution Problems in Distributed Database - Integration and Query Decomposition," Proc. of the JIPDEC Information Systems Seminar on Semantic Aspects of Databases (also available from JIPDEC TR80/01), Tokyo, Japan, Feb. 1980

[TAKIM80c]

Takizawa, M., "Operational Query Decomposition Algorithm in Distributed Databases," JIPDEC TR80/02, JIPDEC, March 1980

[WALTD78]

Waltz, D. L., "An English Language Question Answering System for a Large Relational Database," CACM, Vol. 21, No. 1, July 1978, pp.526-539

[WONGE77]

Wong, E., "Retrieving Dispersed Data from SDD-1: A System for Distributed Databases," Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, Berkeley, May 1977, pp.217-235

[YOUSK77]

Youssefi, K., Whyte, N., Ubell, M., Ries, D., Hawttorn, P., Epstein, B., Berman, R., and Allman, E., "INGRESS Reference Manual-Version 6.1," Memo. ERL-M579, Electronics Research Lab., Univ. of California, Berkeley, Oct. 1977

[ZANIC79a]

Zaniolo, C., "Design of Relational View Over Network Schemas," Proc. of the ACM SIGMOD International Conf. on Management of Data, Boston, MA., May 1979, pp.179-190

[ZLOOM75a]

Zloof, M. M., "Query By Example," AFIPS Conf. Proceedings, May 1975, pp.431-437

付記1. QUELとGSDL

```
<QUEL-query> ::= <range-st> | <retrieve-st> | <update-st>

<range-st> ::=  RANGE [ OF ] ( <var-list> )( <rel-list> );

<retrieve-st> ::= RETRIEVE [ INTO <rel-name> ]
                                    ( <target-list> )
                              [ WHERE <qualification> ];

<var-list> ::=  <variable>  { , <variable> }

<rel-list> ::=  <rel-name>  { , <rel-name> }

<target-list> ::=  <t-clause> { , <t-clause> }

<t-clause> ::=  <att-name> = <a-exp> | <v-attribute>

<v-attribute> ::=  <variable> . <att-name>

<qualification> ::=  <c-pred> | ( <qualification> ) |
                     NOT <qualification> |
                     <qualification>  AND  <qualification> |
                     <qualification>  OR  <qualification>

<c-pred> ::=   <a-exp> <cop> <a-exp>

<a-exp> ::=  <constant> | <v-attribute> | ( <a-exp> ) |
             <a-exp><aop><a-exp> | -<a-exp> |
             <a-function> | <g-function>

<cop> ::=  LT | LE | EQ | GE | GT | NE | =

<aop> ::=  + | - | * | / | **
```

```
<a-function> ::=  <a-func-name> ( <a-exp> {,<a-exp> } )
<a-func-name> ::=  ABS | MOD | ---
<g-function> ::=  <g-func-name> ( <a-exp>
                  [BY <v-attribute> { BY <v-attribute> } ]
                  [ WHERE  <qualification>  ] )
<g-func-name> ::=  COUNT | UCOUNT | SUM | USUM |
                  AVER  | UAVER | MAX  | MIN |
                  ANY
<constant> ::=  <d-constant> | <c-constant>
<d-constant> ::=  <digit> { <digit> }
<c-constant> ::= " <b-letter> {<b-letter> }"
<variable> ::=  <alpha> { <letter> }      ( 4文字以内 )
<rel-name> ::=   <alpha> { <letter> }    ( 16文字以内 )
<att-name> ::=  <alpha> { <letter> }     ( 16文字以内 )
<letter>  ::=  <alpha> | <digit> | <s-letter>
<alpha>  ::= アルファベット ( A~Z ) 及び  ¥
<digit>  ::= 数 字      ( 0~9 )
<s-letter> ::=  _ | #
<b-letter> ::=  任意の文字、記号又は空白
<update-st> ::= <append-st> | <delete-st> | <replace-st>
<append-st> ::=  APPEND [ TO ] <rel-name>  ( <target-list> )
                        [  WHERE    <qualification> ];
```

<delete-st> ::= DELETE <variable> WHERE <qualification> ;

<replace-st> ::= REPLACE <variable> (<target-list>)

[WHERE <qualification>] ;

DD ::= {{ <range-st> } { <define-st> }}

<range-st> ::= RANGE OF (<var-list>)(<lrel-list>);

DRANGE (<var-list>) (<lrel-list>);

<define-st> ::= DEFINE <rel-type> <grel-name> (<a-list>)

<sub-def> { : <sub-def> } ;

<var-list> ::= <variable> { , <variable> }

<lrel-list> ::= <lrel-name> [:<site-no>] {,<lrel-name>[:<site-no>]}

<site-no> ::= <digit> { <digit> } (4桁以内)

<rel-type> ::= ESR | RSR

<a-list> ::= <attribute> [/ <domain-name>][=<format>]

{,<attribute>[/<domain-name>][=<format>]}

<sub-def> ::= (<target-list>) WHERE <qualification>

<lrel-name> ::= <rel-name>

<attribute> ::= <att-name>

<domain-name> ::= <att-name>

付記2. 選　択　度（selectivity）

制限式の選択度とは、この式が参照するノード（リレーション）のオカーランス（組）のうち、この制限式を満足するオカーランスがどの位かを示している。ここでは簡単にするために、原子等価制限式についてのみ考える。原子等価制限式は、x．a＝vの形式をもち、xは組変数（ノードに対応）で、vは離散的な値（整数又は文字列）であり、aは属性を表わしている。よって式の選択度は、属性aの定義域Aのとり得るユニークな値のうち、何個がリレーションx内にあるかを表わしていることになる。

選択度についてさらに検討してみよう。まず、RiをあるDB内のi番目のリレーションとする。α_{ij}をRiのj番目の属性とする。CiをRiの組数（カーディナリティ）、diをRiの次数（Riのもつ属性数）とする。即ち、j＝1、2、…、di。Aijを属性α_{ij}の定義域とする。α_{ij}をリレーションRi内で属性α_{ij}がとるユニークな値の数、β_{ij}を定義域Aijがとり得るユニークな値の数とする。v_{ijk}を、定義域Aij内のk番目のユニークな値とする（k＝1、2、…、β_{ij}）。P_{ijk}を値v_{ijk}が、問合せによって参照される確率とし、$\sum_{k=1}^{\beta_{ij}} P_{ijk}=1$である。$n_{ijk}$を、リレーションRi内で属性$\alpha_{ij}$が値$v_{ijk}$をもつ組数であり、このような組がなければ$n_{ijk}=0$である。ここで、定義域Aij内の値は、先頭から$\alpha_{ij}$番目までがRi内にあり、$\alpha_{ij}+1$番目から$\beta_{ij}$番目まではRi内に存在しないように順序付けられているとする。この仮定は、リレーションのエクステンションは、意味論的に意味のある組の集合であるから、定義域の値がランダムに、エクステンション内にはあらわれないことに基づいている。

以上の用語を用いて、ある値を属性α_{ij}の値としてもつ組数の平均値mは、

$$m=\sum_{k=1}^{\beta_{ij}} n_{ijk}\cdot P_{ijk}$$

$$=\sum_{k=1}^{\alpha_{ij}} n_{ijk}\cdot P_{ijk}+\sum_{k=1\alpha_{ij}+1}^{\beta_{ij}} 0\cdot P_{ijk}$$

$$=\sum_{k=1}^{\alpha_{ij}} n_{ijk}\cdot P_{ijk}\quad \text{となる。}$$

ここで、$n_{ijk}=Ci/\alpha_{ij}$であるので、

$$\therefore\quad m=\frac{Ci}{\alpha_{ij}}\sum_{k=1}^{\alpha_{ij}} P_{ijk}\qquad \cdots\cdots\cdots\cdots (1)$$

リレーション内の値（v_{ij_1}、v_{ij_2}、…、$v_{ij\alpha_{ij}}$）が、この他のものに対して、ℓ倍多く問合せによって参照されるとすると、

リレーション内の値　$v_{ijk}(k=1、2、\cdots、\alpha_{ij})$に対して、

$$P_{ijk}=P_{ij2}=\cdots\cdots=P_{ij\alpha_{ij}}=\ell\cdot\gamma$$

リレーション外の値　$v_{ijk}(k=\alpha_{ij}+1、\alpha_{ij}+2、\cdots、\beta_{ij})$に対して

$$P_{ij\alpha_{ij}+1}=P_{ij\alpha_{ij}+2}=\cdots\cdots=P_{ij\beta_{ij}}=\gamma$$

よって、$\sum_{k=1}^{\beta_{ij}} P_{ijk}=\sum_{k=1}^{\alpha_{ij}} P_{ijk}+\sum_{k=\alpha_{ij}+1}^{\beta_{ij}} P_{ijk}$

$$=\ell\gamma\left(\sum_{k=1}^{\alpha_{ij}}\right)+\gamma\left(\sum_{k=\alpha_{ij}+1}^{\beta_{ij}}\right)$$

$$=\ell\gamma\cdot\alpha_{ij}+\gamma(\beta_{ij}-\alpha_{ij})$$

$$=\gamma[(\ell-1)\alpha_{ij}+\beta_{ij}]=1$$

$$\therefore\quad \gamma=1/[(\ell-1)\alpha_{ij}+\beta_{ij}]\cdots\cdots(2)$$

(1)と(2)式より

$$m=\frac{C_i}{\alpha_{ij}}\cdot\sum_{k=1}^{\alpha_{ij}}\ell\gamma$$

$$=\frac{C_i}{\alpha_{ij}}\cdot\frac{\ell}{(\ell-1)\alpha_{ij}+\beta_{ij}}\cdot\alpha_{ij}=C_i\cdot\ell/[(\ell-1)\alpha_{ij}+\beta_{ij}]\cdots\cdots(3)$$

mは、ある任意の値v_{ijk}を参照する問合せが選択する組の平均数を表わしている。

よって、属性α_{ij}に対する選択度s_{ij}は

$$s_{ij}=m/C_i$$

$$=\ell/[(\ell-1)\alpha_{ij}+\beta_{ij}]\cdots\cdots(4)$$

表1　記　号　の　意　味

記　号	意　　味
R_i	i番目のリレーション
C_i	R_iの組数　（cardinality）
d_i	R_iの属性数　（degree）
a_{ij}	R_iのj番目の属性、ここで$j=1、2、\cdots、d_i$
α_{ij}	R_i内でα_{ij}がとるユニークな値の数、$j=1、2、\cdots、d_i$
A_{ij}	α_{ij}の定義域（domain）
β_{ij}	定義域A_{ij}内のユニークな値の数
v_{ijk}	A_{ij}内のk番目の値 ただ$v_{ijk}(k=1、\cdots、\alpha_{ij})$は$R_i$内に、$v_{ijk'}(k'=\alpha_{ij}+1、\cdots、\beta_{ij})$は$R_i$外である。
P_{ijk}	値v_{ijk}を問合せが参照される確率（$\sum_{k=1}^{\beta_{ij}} P_{ijk}=1$）
n_{ijk}	α_{ij}の値としてv_{ijk}をもつ組の数$n_{ijk}\geq 1$ for $k\geq 1、\cdots、\alpha_{ij}$ $n_{ijk}=①$ for $k=\alpha_{ij}+1、\cdots、\beta_{ij}$
ℓ	R_i内の値が、R_i外の値に対して、問合せが参照する度合
s_{ij}	属性α_{ij}の選択度　$\ell/[(\ell-1)\alpha_{ij}+\beta_{ij}]$ 例　$s_{ij}=\begin{cases}1/\beta_{ij} & \text{for } \ell=1\\ 1/\alpha_{ij} & \text{for } \ell=\infty\\ 2/(\alpha_{ij}+\beta_{ij}) & \text{for } \ell=2\\ 1/(2\beta_{ij}-\alpha_{ij}) & \text{for } \ell=1/2\\ 0 & \text{for } \ell=0\end{cases}$

例として、次のようなリレーションPERSON（NO，NAME，BIRTH－MONTH，BIRTH－YEAR）を考えてみよう。リレーションPERSONのエクステンションは、次のようである。

属性番号	11	12	13	14
PERSON	NO	NAME	BIRTH-MONTH	BIRTH-YEAR
1	805	TARO	Jan.	1955
2	1902	HANAKO	Feb.	1960
3	389	MOMOE	Dec.	1948
4	1145	TAKEKO	Jan.	1961
5	7	MIDORI	Aug.	1965
6	10	HIDEKI	Aug.	1955
7	519	JIRO	Nov.	1969
8	4	UME	Aug.	1948
9	9862	HARUKO	Dec.	1955

BIRTH－MONTHの定義域には、1月から12月までの12個のユニークな値が属している。このリレーションのインテグリティ条件として、12人の部のなかでBIRTH－YEARは1940年以降、1970年まで生まれたものだけが許されるとなると、BIRTH－YEARの定義域には、30コの値が属している。属性を、属性番号（11、12、13、14）で呼ぶことにすると、αijとBijの値は、次のようになる。

(ij)	11(NO)	12(NAME)	(BIRTH-13 MONTH)	(BIRTH-14 YEAR)
αij	9	9	5	6
βij	12	12	12	30

ℓの種々の値に対する選択度s_{ij}の値を表2にまとめる。

表2　ℓに対するs_{ij}の値

		NO	NAME	BIRTH-MONTH	BIRTH-YEAR
ℓ	s_{ij}	11	12	13	14
0	0	0	0	0	0
1/10	$1/(10\beta_{ij}-9\alpha_{ij})$	1/111 =0.00901	0.00901	1/75 =0.0133	1/246 =0.00407
1/2	$1/(2\beta_{ij}-\alpha_{ij})$	1/15 =0.0667	0.0667	1/19 =0.0526	1/54 =0.0185
1	$1/\beta_{ij}$	1/12 =0.0833	0.0833	1/12 =0.0833	1/30 =0.0333
2	$2/(\beta_{ij}+\alpha_{ij})$	2/21 =0.0952	0.0952	2/17 =0.118	1/18 =0.125
10	$10/(\beta_{ij}+9\alpha_{ij})$	10/93 =0.108	0.08	10/57 =0.175	10/84 =0.119
100	$100/(\beta_{ij}+99\alpha_{ij})$	100/903 =0.111	0.111	100/507 =0.197	100/524 =0.160
∞	$1/\alpha_{ij}$	1/9 =0.111	0.111	1/5 =0.2	1/6 =0.167

例えば、range　p　PERSON；

retrieve (p.name)　where p. birth—month = " Jan " ；

という問合せを考えてみよう。

属性 birth－month の選択度は、$\ell=2$の時、0.118であるので、条件を満足する組数の期待値は

は 9・0.118=1.06 となる。

又　　$\ell=\infty$の時、0.2であるので、条件を満足する組数の期待値

は 9・0.2=1.8 となる。

付記3.　3年間の研究開発経過

当研究開発は昭和52年度から3年計画で遂行された。本項では、当研究開発の一環として実施した海外及び国内出張先、成果として外部に発表した論文、報告書、とりまとめた内部資料等について紹介する。

国内出張（順不同）

北海道大学　工　学　部
東北大学　電気通信研究所
東北大学　工　学　部
名古屋大学　プラズマ研究所
京都大学　工　学　部
愛知県庁　電子計算課
国立民族学博物館
その他、メーカ講習会、学会研究会等

参加国際会議

第3回　Very Large Data Base（東京、1977－10）
第3回　Berkeley Workshop（サンフランシスコ、1978－8）
第4回　Very Large Data Base（西ベルリン、1978－9）
第4回　ICCC′78（京都、1978－9）　…　論文発表
第6回　Artificial Intelligence（東京、1979－8）

情報処理学会（論文発表）

第19回　全国大会（東京、1978－8）
第20回　全国大会（東京、1979－7）
第21回　全国大会（東京、1980－5）　（予定）

海外出張 — 訪問先と主要面接者 —

1 UC. Berkeley

Electronics Research Laboratory, College of Engineering,

University of California, Berkeley

所 在 地

Berkeley, California 94720, U. S. A.

Prof. M. Stonebraker

UC. Berkeley, Electronics Research Laboratory

Dr. E. Allman

UC. Berkeley, Electronics Research Laboratory

2 SRI スタンフォード研究所

Artificial Intelligence Center, Stanford Research

Institute, International

所 在 地

333 Ravenswood Avenue, Menlo Park, California 94025,

U. S. A.

Tel: 415-326-6200

Dr. D. Sagalowicz

SRI, International, Artificial Intelligence Center

3　CNRS　科学技術情報センター

Centre National de la Recherche Scientifique

Centre de Documentation Scientifique et Technique

Informascience

所　在　地

26, Rue Boyer - 75971 Paris Cedex 20, France

Tel:　797. 35. 59

Dr. Y. Salle　（女　史）

CNRS, Informascience, Chargés de mission

Dr. P. Buffet

CNRS, Informascience, Informatique

Dr. Y. Henry　（女　史）

CNRS, Informascience

4　IRIA　情報処理自動化研究所

Institut de Recherche d' Informatique et d' Automatique

所　在　地

Domaine de Voluceau - Rocquencourt

B. P. 105 - 78150 Le Chesnay - France

Tel:　954. 90. 20

Dr. W. Litwin

5 Univ. of Grenoble グルノーブル大学

Laboratoire d' Informatique

Universite' Scientifique et Medicale de Grenoble

所 在 地

B. P. 53

38041 Grenoble Cedex; France

Tel: (76) 54 81 45

Prof. M. Adiba

Laboratoire d'Informatique, Univ. of Grenoble

Prof. C. Delobel

Laboratoire d'Informatique, Univ. of Grenoble

6 GMD 西独数理データ処理公社

Gesellschaft für Mathematik und Datenverarbeitung mbh Bonn

所 在 地

PO BOX 1240, Schloss Birlinghoven,

D-5205 St. Augustin 1, West Germany

Tel: (0 22 41) 14 - 1

Dr. K. Voss

GMD, Chief, Chairman for EDP-systems Application

in National Administrations

Dr. S. Christiansen (女 史)

GMD, INGVER project

Dr. G. Wurch

GMD, INGVER project

Dr. H. Scholev

GMD, FIDAS Consultant Manager

7 GLC 大ロンドン市議会

Greater London Council, Central Computer Service

所 在 地

The County Hall, London SE1 7PB, U. K.

Dr. A. E. Mitson

GLC, Head of Computing Services

Dr. C. Doggrell

GLC, Computing Services

Dr. H. Kon （女 史）

GLC, Computing Services, IPS Manager

年　度　報　告　書

(a) 分散型リソース処理技術の研究開発（52−S001）、昭和53年3月

この報告書は、昭和52年度から3ケ年計画で発足した「分散型リソース処理技術の研究開発」の第一年度目の報告書であり、分散処理、データ共有方式の検討、日本語情報処理の3部より構成されている。

第Ⅰ部の分散処理は、分散処理の一般論を述べている。第1章では分散処理の出現の背景を述べ、分散処理の定義をおこなっている。第2章では、ネットワークユーザの分類をおこない、ユーザの視点の重要性を論じている。第3章においては、ネットワークで共有されるリソースを検討し、データが特に重要なリソースであることを指摘している。

第Ⅱ部では、分散処理におけるデータ共有方式について議論している。まず第1章では、ユーザの視点からみた分散処理技術に関して、第Ⅰ部で分類を試みた分散処理関係者の各層に対応したシステムをそれぞれ紹介している。第2章では、データを管理する強力な道具としてデータベース管理システム（ＤＢＭＳ）をとりあげ議論している。第3章では、ネットワーク技術とデータベース技術を有効に組み合わせて実現している分散型データベースについて議論している。第4章では、ネットワーク上に散在するリソースを統合していく上で重要な機能の一つであるネットワーククリアリングセンターについて議論している。特に、リソースの所在源情報管理の基本的技術としてＤＤ／Ｄに焦点をあて、ＤＢＭＳの結合形態やディレクトリの配置や管理方法について言及している。第5章では、異機種のコンピュータから構成されたコンピュータネットワークを利用するユーザが、まず直面する問題である制御言語の多様性を解決する技術について議論している。つまり、コンピュータの異機種を克服するために設定した標準制御言語のシステムを示している。第6章では、第5章までの検討に基づいて、異種コンピュータネットワークにおけるリソース統合問題を採り上げている。

(b) 分散型リソース処理技術の研究開発（53−S001）、昭和54年3月

この報告書は、昭和52年度から3ケ年計画で発足した「分散型リソース処理技術の研究開発」の第二年度目の報告書であり、分散型データベースシステム、日本語情報処理の2部より構成されている。

第Ⅰ部は分散型データベースシステムに関する研究開発についてとりまとめたものであり5つの章からなっており、それぞれ次のような内容について述べている。第1章では、当研究開発の背景とその目的について述べている。第2章では、分散及び異種性問題を解決するために設定した4層スキーマ構造に基づく分散型データベースシステムのアーキテクチャについて述べている。第3章では、第2章で示したシステムをサポートするネットワークデータディレクトリが保持する情報を生成する、クリアリング情報生成システムについて述べている。第4章では、海外調査に基づいて分散型リソース処理技術の技術動向について述べている。第5章では、当研究開発でとりまとめた論文・資料について紹介している。

(c) 分散型リソース処理技術の研究開発（54−S001）、昭和55年3月

本報告書。

(a) ICCC' 78 京都、１９７８年１０月

Takizawa, M., Hamanaka, E. and Ito, T.: Resource Integration and Data Sharing on Heterogeneous Resource Sharing System, ICCC'78, Kyoto, Sept. 1978, pp.253-258

There are two problems to discuss regarding the resource sharing system employing computer networks such as ARPANET and CYCLADES; resource heterogeneity problems accompanying the incorporation of various host computers, and distribution problems posed by resources combined through networks. In this paper, to solve the two above-mentioned problems, we classify existing approaches and then propose our approach. Our approach consists of two stages of conceptualization. First, the heterogeneity problem is removed by translating local models into common model. Secondly, the distribution problem is removed by integrating the common models for each site to the composed resource of interest to a network community. We apply this approach to the general-purpose distributed data base on networks.

(b) 情報処理学会欧文誌

Takizawa, M. and Hamanaka, E.: The Four-Schema Concept as the Gross Architecture of Distributed Databases and Heterogeneity Problems, Journal of Information Processing, Vol. 2, No. 3, 1979, pp.134-142

This paper presents the gross architecture of distributed database (DDB), which consists of four schemas and mappings between them. We call it the four-schema structure (FSS). In the FSS, the heterogeneity of each DBMS, i.e. the differences of data models and languages, distributed over networks is first removed (call it homogenization), and then the distribution of such homogenized DBMS, i.e. the information of where required data is located, is removed

(call it integration). Inverse mappings of them are respectively the translation of a common local query into DBMS queries, and the decomposition of a global query into local queries.

In this paper, focusing on heterogeneity problem, we can present the FSS concept and DDB architecture based on it, show the homogenization process, and discuss the query translation and also required information (heterogeneity information or HI).

(c) IFIP'80, 東京、1980年10月 (to appear)

Takizawa, M. and Hamanaka, E.: Query Translation in Distributed Databases, IFIP'80, Tokyo, Oct. 1980

In distributed databases (DDBs), query processing is a very important problem. A user's query based on a global view (call it a GCS) to all the data in the DDB has to be decomposed into local queries. These reference only relations in the local site. Then, at each site, the local query in a common language has to be translated into DMLs executable on the local DBMS. The former is called a query decomposition (QD) and the latter a query translation (QT). This paper focuses on the QT in DDB problems.

The translation problem of a QUEL query into DBTG DMLs is discussed in particular. The QT is composed of three main modules: Structure Transform (ST), Optimizer (OPT), and DML Generator (DMLG). The ST translates the structure of the query into the DBTG structure. The OPT generates the optimal access path. An algorithm called DFA which results in the least intermediates is also proposed. The DMLG generates the real DBTG DMLs from the access path. In this paper we try to clarify the translation process and required information.

(d) 投 稿 中

Takizawa, M.: Operational Query Decomposition Algorithm in Distributed Databases

There are two main problems in designing the distributed database systems (DDBSs): 1) heterogeneity problem, and 2) distribution problem. The heterogeneity problem is how to overcome the differencies of the data models and languages of database systems (DBSs). The distribution problem is how to integrate such homogenized DBSs into one logical and virtual database system. Our approach called a four-schema structure (FSS), first, solves the former and then the distribution problem.

This paper concentrates on the distribution problem. It contains three main parts. The first one is called an integration which is a process of integrating the homogenized DBSs (call them local conceptual schemas or LCSs) into one logical DBS (call it a global conceptual schema or GCS). The second is a query decomposition (QD) which is a process of decomposing a global query into a sequence of queries executable on each DBS. The last is an information required by the QD and removed in the integration, which represents the correspondence between the GCS and LCSs.

We propose the GCS definition language (GSDL) which is an extension of a relational calculus language QUEL, in the integration. In the QD, the algorithm for transmission of data which minimizes the required information and keeps it static based on the dynamic decisions is proposed. In the DI, we show its schema in a relational form.

(e) JIPDEC セミナー 1980年2月

Takizawa, M.: Distribution Problems in Distributed Databases Integration and Query Decomposition, JIPDEC Information Systems Seminar on Semantic Aspects of Databases, JIPDEC, Tokyo, Feb. 1980

情報処理学会全国大会における論文

(a) 浜中、滝沢、伊藤：

ＪＩＰＮＥＴにおけるリソース統合システム、昭和５３年度、１９回大会、２Ｉ－１

当財団は、昭和４８年度から４年間に渡り、リソースシェアリングコンピュータネットワークＪＩＰＮＥＴを開発した。ＪＩＰＮＥＴにより国産の３機種による異機種間のリソースシェアリングを実現した。当研究開発では、更に一歩進めて異機種コンピュータネットワークにおけるリソース統合問題を検討している。本論文では、ＪＩＰＮＥＴにおけるリソース統合システムとして、異種のＤＢＭＳから成る分散型データベースシステムを採り上げ、解決すべき問題点、システムの持つべき機能について検討を行う。

(b) 浜中、滝沢：

ＪＩＰＮＥＴにおけるリソース統合システム(2) — 分散型データベースシステム（ＪIPNET — ＤＤＢ）とネットワークデータディレクトリ（ＮＤＤ）、昭和５４年度、２０回大会、２Ｄ－７

当財団では、コンピュータネットワークＪＩＰＮＥＴを基本ファシリティとしたデータ主体のリソース統合システム、即ち分散型データベースシステム（ＪＩＰＮＥＴ－ＤＤＢ）の研究開発に取組んでいる。ＪＩＰＮＥＴ－ＤＤＢを構成するＤＢＭＳは異種（heterogeneous）であり、かつ地域的に分散していることを前提としている。本論文では、この様なＤＤＢを統一的に利用できるように設定した４層スキーマ構造とネットワークデータディレクトリ（ＮＤＤ）について述べる。

(c) 滝沢、浜中：

ＪＩＰＮＥＴにおけるリソース統合システム(3) — 分散型データベースシステム（ＪIPNET — ＤＤＢ）と問合せ処理、昭和５４年度、２０回大会、２Ｄ－８

ＪＩＰＮＥＴ－ＤＤＢにおける４層スキーマ構造（ＦＳＳ）概念と、同種化及び統合化において除去される分散情報（ＤＩ）と異種性情報（ＨＩ）とは２Ｄ－７において論じられた。本論文では、同種化と統合化の各々のプロセスである問合せ変換（ＱＴ）と問合せ分割（ＱＤ）とについて考える。ＱＤは、ＧＣＳに基づいた全体問合せ（ＧＱ）を、一連の各サイト単位のＬＣＳに基づいた局所問合せ（ＬＱ）に、ＤＩを用いて分割するプロセスである。ＱＴは、ＱＤされたＬＱを、各ＤＢＭＳで実行可能なＤＭＬに、ＨＩを用いて変換するプロセスである。ＧＱ及びＬＱは、関係計算言語ＱＵＥＬで記述されており、簡単にするために接合問合せとする。本論文ではＱＤ及びＱＴの概要を論じる。特にＱＴでは、ＤＢＴＧ　ＤＭＬへの変換について考える。

(d) 浜中、滝沢：

ＪＩＰＮＥＴにおけるリソース統合システム(4) — 分散型データベースシステム（ＪIPNET — ＤＤＢ）における問合せ変換と一考察、昭和５５年度、２１回大会、４Ｆ－５

分散型データベース（ＤＤＢ）の開発には分割型（top－down）と統合型（bottom－up）と呼ばれる２つのアプローチがあり、我々が目指すＤＤＢ（ＪＩＰＮＥＴ－ＤＤＢ）は後者の統合型である。このアプローチの採用理由は、既に巨大なデータリソースが存在している事実と、リソース

の最適配置機能を付加すれば容易に分割型のアプローチへも適用可能であるという２点に集約できる。本論文では、ＪＩＰＮＥＴ－ＤＤＢの全体アーキテクチャ、問合せ変換における構造変換、最適化、データ操作言語の生成、さらに実験を通して得られた一考察について論じる。

(e) 滝沢、浜中：

ＪＩＰＮＥＴにおけるリソース統合システム(5) — 問合せ分割、昭和５５年度、２１回大会、４Ｆ－６

分散型データベースシステムの全体概念スキーマ（ＧＣＳ）レベルでは、どこに何があるかという分散問題は除去されている。従ってＧＣＳに基づいた問合せ（ＧＱ）は、各サイト及び２つのサイト間で実行可能な一連の問合せ（ＬＱ）へ分割されねばならない。これを問合せ分割（ＱＤ）と呼ぶ。ＱＤは、１）参照されるＧＣＳリレーションを対応するＬＣＳリレーションへの分割（表現変換）と、２）サイト間にまたがったＬＱの処理（サイト間処理）との２つから成っている。1）については問合せ変形手法を用いて処理される。本報告では、１）の処理後のＧＱ（ＧＬＱ）のサイト間処理について論じる。

箱根プログラミングシンポジウムにおける論文

滝沢、浜中：

分散型データベース（ＪＩＰＮＥＴ－ＤＤＢ）における問合せ変換、情報処理学会、第２１回プログラミングシンポジウム、箱根、昭和５５年１月

コンピュータネットワークを介して、地理的に分散しているデータベース（以降DBと略す)を結合し統合的に利用するシステムは、分散型データベース（ＤＤＢ）と呼ばれるものである。ＤＤＢの開発には、大別して２つのアプローチがある。１つは、ＤＤＢ全体に、ある仮想的なＤＢを設定し、この中身をある目的関数のもとで各サイトに最適配置する方法である。これは、分割型（top－down）ＤＤＢと呼ばれ、ＳＤＤ－１〔ＲＯＴＨＪ　７７〕、ＩＮＧＲＥＳ〔ＳＴＯＮＭ　７７〕等がその例である。

もう１つのアプローチは、統合型（bottom－up）と呼ばれるものである。これは相異なる意味構造を持つ既存ＤＢを統合し、ＤＤＢ全体の統一視野をユーザに提供することを目的にしている。例として、ＰＯＬＹＰＨＭＥ〔ＡＤＩＢＭ　７８〕がある。

我々の目指すＤＤＢも後者のタイプである。既に巨大なデータリソースが存在している事実を無視できないことと、更にリソースの最適配置機能を付加することによって容易に前者のアプローチへも適用できることの２点がその理由である。

本論文では、我々のＤＤＢの全体アーキテクチャを述べるとともに、現在インプリメント中の問合せ変換について論じる。

日自振論文集

小関、浜中、滝沢、野村、小池：

分散型リソース処理技術の研究開発、日本自転車振興会研究開発論文集、昭和５５年３月

コンピュータ利用の高度化及び複雑化に伴い、ハードウエア、ソフトウエア、データ等のリソートの集中処理はほぼ限界に達してきている。

分散処理システムの出現によって大量のデータ及び複雑な処理が可能になった反面、新たに発生した問題、即ち異機種に分散しているデータ及びプログラムをいかにして効率よく、且つ容易に利用するかという問題の解決に迫られている。このような問題を解決するためには、分散処理システム内のリソース全体を総合的に把える必要がある。

また我国の情報処理の特徴の一つである日本語情報処理（漢字処理）について特に分散型リソース処理という観点で把えると、日本語情報のデータベースが重要なリソースとして期待されており、これを利用するためには日本語の入出力機能を備えた、且つ低コストで操作性に優れた日本語端末が必要である。

以上の要請に基づいて昭和５２年度より３ケ年計画で実施しているのが本研究開発であり、本論文はその実施内容についての概説である。

本論文は大きく二つの部分から成り立っている。１では異機種に分散したリソースを統合利用する技術（分散型データベースシステムの研究開発）について、２では安価で操作性に優れた日本語端末の開発とその技術（日本語端末の研究開発）について論じる。

ＪＩＰＤＥＣジャーナル　№３６、昭和５３年１２月

欧・米の分散型リソース処理技術

当研究開発の一貫として昭和５３年８月下旬から約３週間に渡り調査訪問した欧米４ケ国計７つの先進的研究機関（大学及び研究所）における調査概要について述べている。今回の調査は、分散型リソース処理技術及び評価技術とデータの収集を目的として行った。報告の最後に、実用化への技術的問題点について述べている。

内部発表会等

海外出張報告会　　（１９７８－１１）

プロジェクト中間成果報告会．（１９７９－１０）

研究会

データベースの意味論的側面に関する研究会（ＪＩＰＤＥＣ Information Systems Seminar on Semantic Aspects of Databases）（1980－2）

内部資料（順不同）（unpublished）

海外調査概要報告書

クリアリング情報生成システム （７９年３月）

異種性情報生成サブシステム

システム基本設計書／プログラム設計・仕様書

分散情報生成サブシステム

システム基本設計書／プログラム設計・仕様書

問合せ処理システム （７９年７月）

問合せ変換サブシステム

システム基本設計書

分散型データベース管理における normalized schema と distribution view

―既存分散型データベースシステムの検討 Jan. 1978

分散型データベースに対する概念化のアプローチ Feb. 1978

Formalization of Existing DBMS Schemas: The Syntactical Transformation of Existing Data Models into an E-R Model, June 1978

今後の方針について Nov. 1978

Our Investigations on DDBMS Dec. 1978

On the Integrity Constraints Dec. 1978

Query Translation Dec. 1978

Query Processing Jan. 1979

Query Processing in a Relational Database System Feb. 1979

Global Conceptual Schema (GCS) and Globel-Local Query Decomposition (GLQD) Feb. 1979

On Distributed Query Processing	April 1979
分散型リソース処理技術の研究開発プロジェクトの現状と次期プロジェクトの提案	May 1979
Query Translation in Distributed Databases	Sept. 1979

付記4. 略 語 表

本報告書で用いられる略語と英語名及び日本語名の対応表を以下に示す。
〔 〕内には、この用語について特に詳しく述べられている場所を示している。

付記4.1 本報告書内の略語表

A		
AA	application administrator [3]	アプリケーション管理者
AL	access link [5.5.2]	アクセスリンク
AT	access tree [5.4.3]	アクセス木
AU	access unit [5.5.1]	アクセス単位
	attribute	属 性
B		
BFA	breadth-first algorithm [5.4.1]	横型アクセス木 生成アルゴリズム
BLDBS	bibliography database system [4.6.3]	文献データベースシステム
	binary tree	2分木
C		
CC	clearing center	案内センタ
CC	control concurrency and consistency control [2]	同時実行及び一致性制御
CDBS	centralized DBS	集中型データベースシステム

CN	confluent node [5.4.1]	合流ノード
CNF	conjunctive normal form [5.1]	積正規形
	conceptual schema [1]	概念スキーマ
	conjunction	論 理 積
	connectivity [5.4.2]	結 合 度
	consistency [2]	一 致 性

D

DB	database	データベース
DBA	database administrator [1]	データベース管理者
DBM	database machine [1]	データベースマシン
DBS	database system	データベースシステム
DD	distribution description [6.2]	分散記述
DDBS	distributed database system	分散型データベース システム
DFA	depth-first algorithm [5.4.2]	縦型アクセス木 生成アルゴリズム
DGQ	decomposed GCS query [7.4]	分割されたＧＣＳ問合せ
DI	distribution information [6]	分散情報
DLDP	destination LDP [7.7]	目的ＬＤＰ
DML	data manipulation language	データ操作言語
DMLG	DML generator [5.5]	ＤＭＬ生成システム
DNF	disjunctive normal form [5.1]	和正規形
DQG	DBTG query graph [5.3]	ＤＢＴＧ問合せグラフ
	data model	データモデル

	destination relation (r) [7.7]	目的リレーション
	destination site (j) [7.7]	目的サイト
	disjunction	論 理 和
	domain	定義域（領域）
E		
EHS	explicitly hidden structure [5.3.2]	明隠れ構造
EI	external information [3]	外部情報
ES	entity-set [4.2.1]	事象集合
ESR	entity-set relation [4.2.1]	事象集合リレーション
EXS	external schema [3]	外部スキーマ
F		
FRAU	failure-return access unit [5.5.2]	失敗復帰アクセス単位
FSS	four-schema structure [3.1]	四層スキーマ構造
G		
GA	global administrator [3.1]	全体管理者
GCS	global conceptual schema [3.1]	全体概念スキーマ
GDP	global database processor [3.3]	全体データベースプロセッサ
GLQ	global LCS query [7.5]	全体ＬＣＳ問合せ

GLQG	global LCS query graph [7.6]	全体ＬＣＳ問合せグラフ
GQ	GCS query [8]	ＧＣＳ問合せ
GSDL	GCS definition language [6.2, 付記1]	ＧＣＳ定義用言語
	gross architecture [2, 3.1]	全体アーキテクチャ
H		
HI	heterogeneity information [4.5]	異種性情報
HIMLDP	heterogeneity information manage-	異種性情報管理局所
	ment local database processor [3.3]	データベースプロセッサ
HIPS	heterogeneity information process-	異種性情報生成システム
	ing system [8.1]	
HQD	horizontal query decomposition	水平問合せ分割
	[7.5]	
HS	hidden structure [5.3.2]	隠れ構造
I		
IHS	implicitly hidden structure [5.3.2]	暗隠れ構造
ILQP	initial local query processing [7.6]	初期局所問合せ処理
		内部スキーマ
	internal schema [1]	

J		
JL	join link [5.3.1]	結合リンク
JQG	join query graph [7.6]	結合問合せ図
	join	結　合
L		
LA	local administrator [3.1]	局所管理者
LCS	local conceptual schema [3.1]	局所概念スキーマ
LCT	logical transmission cost table [7.7]	論理転送コスト表
LDP	local database processor [3.3]	局所データベースプロセッサ
LIS	local internal schema [3.1]	局所内部スキーマ
LISD	LIS DML program [5]	LIS DML プログラム
LNAU	Last-appeared next access Unit [5.5.2]	最後に現れた次アクセス単位
LQ	LCS query [5]	LCS問合せ
M		
MDNF	modified disjunctive normal form [7.11]	変形和正規形

N

NA	network accessor [3.3]	ネットワークアクセスシステム
NAP	network access protocol	ネットワークアクセスプロトコル
NAU	next access unit [5.5.1]	次アクセス単位
NC	network community [3]	ネットワーク共同体
NDD	network data directory [3.2]	ネットワークデータディレクトリ
	null value [6.2]	未定義値
	n-ary tree	n分木

O

OAU	owner access unit [5.5.1]	親アクセス単位
OCA	expected number of occurrenced to be accessed [5.4.2]	アクセスされるオカーランスの期待値
OPT	optimizer [5.4]	最 適 化
	occurrence tree [5.4.2]	オカーランス木

P

PMDBS	project management database system [4.6.1]	プロジェクト管理データベースシステム
PRDBS	project database system [4.6.2]	プロジェクトデータベースシステム

Q

QD	query decomposition [7]	問合せ分割
QG	query graph [5.3]	問合せグラフ
QPI	query processing information [7.7]	問合せ処理情報
QT	query translation [5]	問合せ変換
QTP	query translation processor [5.6]	問合せ変換システム
	query	問合せ

R

RDBS	relational DBS	リレーショナルデータベースシステム
RQG	relational query graph [5.3.1]	リレーショナル問合せグラフ
RS	relationship-set [4.2.1]	関係性集合
RSR	relationship-set-relation [4.2.1]	関係性集合リレーション
	range	値　域
	record-type	レコード型
	relation	リレーション
	restriction	制　限
	restriction link [5.3.1]	制限リンク
	result attribute	結果属性
	result link [5.3.1]	結果リンク
	result relation	結果リレーション

S		
SLDP	source LDP [7.7]	ソースＬＤＰ
SRAU	success-return access unit [5.5.2]	成功復帰アクセス単位
ST	structure transform [5.3]	構造変換
	stage [7.7]	ステージ
	schema	スキーマ
	scheme	スキーム
	selectivity [5.4.2]	選択度
	semantic aspect [3]	意味論的側面
	set-type	セット型
	source relation (r') [7.7]	ソースリレーション
	source site (i) [7.7]	ソースサイト
	subtree selectivity [5.4.2]	部分木選択度
	survivability [3]	長寿命性
	syntactic aspect [3]	構文的側面
T		
TS	transmission scheduling [7.7]	転送スケジューリング
TSA	transmission scheduling algorithms [7.7]	転送スケジューリングアルゴリズム

V

VS	value set [4.2.1]	値 集 合

W

WS	working space [7.1]	作業領域
WSA	WS allocated [7.7]	ＷＳＡメッセージ
WSM	WS manager [7.1, 7.8]	ＷＳ管理システム

付記 4.2　データベース関係用語の和英対応例

AA, See application administrator		
abnormal termination	1	異常終了
abstraction	1	抽象化
access	1	アクセス
	2	呼出し
access constraints	1	= INTEGRITY
	2	= SECURITY
access control	1	呼出し制御
	2	アクセス制御
access method	1	アクセス法
	2	呼出し法
access path	1	アクセスパス
	2	呼出し経路
	3	アクセス経路
	4	: NAVIGATION
access strategy	1	アクセス戦略
accessing model	1	アクセスモデル
	2	呼出しモデル
addressing	1	番地付け
	2	番地指定
after image	1	直後像

aggregation		1	集 合 （体）
		2	データ （群）
anomaly		1	不整合
		2	: UPDATE
		3	: DELETION
		4	: INSERT
		5	: STORAGE
application program		1	応用プログラム
		2	適用業務プログラム
application administrator		1	応用管理者
architecture		1	アーキテクチャ
area		1	領 域
assertion		1	確認命題
association		1	連 関
		2	連 想
		3	= RELATIONSHIP
associative retrieval	ALPH	1	（値による）連想検索
atomic relation		1	原リレーション
attribute(s)		1	属 性
		2	= FIELD
		3	= DATA ITEM

authentication (security)	1	認　証
	2	確　証
authorization	1	認　可
	2	認　証
authorization mechanism	1	認証機構
automatic	1	自　動
axiomatization	1	合理化
B-tree	1	B氏木
	2	Bツリー
back-end	1	後　置　（型）
backman diagram	1	バックマンダイアグラム
	2	バックマン（線）図
backup	1	状態取得
	2	バックアップ
balanced tree	1	釣合木
	2	平衡木
base relation	1	基底関係
before image	1	直前像
binary operator	1	2項演算子
binary relation	1	2項関係
binary search	1	2分探索
binding	1	接　合

block		1	ブロック
		2	= PHYSICAL RECORD
bucket		1	バケツ
buffer		1	バッファ
calculation key	DBTG	1	計算キー
		2	CALCキー
calculation location mode	DBTG	1	計算配置方式
candidate key		1	キー候補
		2	候補キー
		3	準キー
canonicial form		1	正準形
cardinality		1	対応濃度
		2	基　数
		3	個　数
casual user		1	不特定利用者
		2	不定時利用者
		3	一般ユーザ
certification		1	検　証
chain		1	連　鎖
		2	鎖
		3	チェイン

check point		1	確認点
		2	チエックポイント
		3	検査点
child		1	子
child / twin pointers	IMS	1	子／兄弟ポインタ
child segment	IMS	1	子セグメント
class relation		1	群関係
clause		1	論理節
closure		1	閉 包

CODASYL, see the Conference on Data Systems Languages

code		1	符 号
		2	コード
collision (hash addressing)		1	(ハッシングの結果の)同値
		2	= SYNONYM
command		1	指 令
		2	コマンド
communication area		1	通信域
compaction		1	圧 縮
		2	= COMPRESSION
compatibility		1	互換性
		2	両立性

completeness		1	完全性
complexity		1	複雑さ
component		1	成　分
compound attribute		1	複合性
compound domain		1	複合定義域
compression		1	圧　縮
		2	= COMPACTION
computer naive user		1	電子計算機素人利用者
concatenated key		1	連想キー
concatenation		1	連　結
		2	結　合
conceptual level		1	概念レベル
conceptual model		1	概念モデル
conceptual record		1	概念レコード
conceptual schema		1	概念スキーマ
		2	概念図式
concurrency		1	同時実行性
		2	並行性
concurrency control		1	同時実行制御
		2	= SHARED CONTROL
concurrent run unit	DBTG	1	同時実行単位
conference on data systems languages			データシステムズ言語協議会

connect		1	接　続
connection(s)		1	（レコード間の）結合関係
consistency		1	整合性
		2	無矛盾性
		3	一貫性
constraints		1	条　件
		2	制　約
contents		1	（レコードの）内容
context		1	文　脈
		2	場
		3	前後関係
conversion		1	変　換
copy		1	複　写
correctness		1	正当性
		2	正確性
cross reference		1	相互参照
currency indicator	DBTG	1	現在指示子
cycle		1	サイクル
		2	輪
data		1	データ
data administration facility		1	データ管理者機能
data administrator		1	データ管理者

data aggregate		1	データ群
data bank		1	データバンク
		2	: DATABASE
data base		1	データベース
data base control system			データベース制御システム
data base key	DBTG	1	データベースキー
data base task group		1	データベース作業班
data compression		1	データ圧縮
data conversion		1	データ変換
		2	: DATA TRANSLATION
data definition		1	データ定義
data description		1	データ記述
data description language		1	データ記述言語
data dictionary		1	データデイクショナリ
		2	データ辞書
data dictionary / directory		1	データデイクショナリ／デイレクトリ
		2	データ辞書／簿
data directory		1	データデイレクトリ
		2	データ簿
data independence		1	データ独立（性）
data integrity		1	データ保全性
		2	データ一貫性

data item	1	データ項目
	2	= ATTRIBUTE
	3	= FIELD
data manipulation facility	1	データ操作機能
data manipulation language	1	データ操作言語
data materialization	1	データ具現
data migration	1	データ移動
data model	1	データモデル
	2	データ模型
data name	1	データ名
data set	1	データセット
	2	= FILE
	3	= RELATION
data sharing	1	データ共有
data storage description language	1	データ蓄積記述言語
data structure	1	データ構造
data structure diagram	1	データ構造図
data sublanguage	1	データ準言語
data submodel	1	データ準モデル
data translation	1	データ変換
	2	: DATA CONVERSION

data transmission	1	データ転送
database	1	データベース
database administrator	1	データベース管理者
database description	1	データベース記述
database machine	1	データベースマシン
	2	データベース機械
database management system	1	データベース管理システム
database manager	1	データベース管理
database procedure	1	データベース手続き

DB, see database / data base

DBA, see database administrator

DBCS, see database control system

DBMS, see database management system

DBTG, see data base task group

DD/D, see data dictionary / directory

DDB(S), see distributed database (system)

DDL, see data description language

deadlock	1	すくみ
	2	デッドロック
decoding	1	復　号
decomposition	1	分　割
	2	分　解
decryption	1	解　読

default		1	省略時
degree		1	次　数
		2	度　数
delete		1	削　除
delimiter		1	区切り記号
derived		1	導　出
		2	派　生
		3	誘　導
derived relation		1	派生関係
description entry		1	記述項
		2	= ENTRY
destaging		1	せり下げ
determinant		1	決定項
device independence		1	装置独立
device media control language		1	装置媒体制御言語
diagram		1	図
		2	図　式
difference	REL	1	差
direct location mode	DBTG	1	直接配置方式
directory		1	見出し簿
		2	ディレクトリ
		3	簿

disconnect		1	切り離し
disjunctive mapping		1	選言写像
distributed database (system)		1	分散(型) データベース(システム)
distribution independence		1	分散独立
DMCL, see device media control language			
domain		1	定義域
		2	領 域
		3	: SIMPLE
		4	: COMPOUND
DSDL, see data storage description language			
DSL, see data sublanguage			
dump		1	写 し
duplicate		1	複 写
dynamic set	DBTG	1	動的親子集合
elimination of duplicates		1	(結果の)重複排除
embedded multivalued dependency		1	潜在多値従属性
empty set	DBTG	1	空親子集合
encoding		1	符号化
encryption		1	暗号化
encryption, see scrambling			
end user		1	エンドユーザ
		2	末端利用者

end user facility task group	1	エンドユーザ機能作業班
entity	1	事　像
	2	主　体
	3	実　体
	4	主　像
	5	実　在
entity-relationship	1	事像ー関係性
entry	1	（表の）記入項目
	2	記述項
	3	（データの）投入
	4	（プログラムの）入口
	5	= DESCRIPTION ENTRY
equi-join	1	等（価）結合
	2	: NATURAL JOIN
	3	: GREATER-THAN JOIN
error recovery	1	誤り復帰
	2	誤り回復
	3	障害回復

EUFTG, see end user facility task group

exclusive control		1	排他制御
external level		1	外部レベル
external model		1	外部モデル
external record		1	外部レコード
external schema		1	外部スキーマ
facility		1	機　能
failure		1	故　障
fault		1	欠　陥
field		1	フィールド
		2	= ATTRIBUTE
		3	= DATA ITEM
file		1	ファイル
		2	= DATA SET
		3	= RELATION
find		1	呼出し
flat file		1	平坦ファイル
foreign key		1	外来キー
		2	外部キー
	IMS	3	= SYMBOLIC POINTER
formula		1	(論理)式
front end		1	前置(型)
full (functional) dependence		1	全(関数)従属
function		1	機　能

functional dependence	1	関数従属
functional dependency	1	関数従属性
general format	1	一般形式
general rules	1	一般規則
get	1	呼出し（ＧＥＴ）
granularity of locks	1	ロッキング単位
	1	顆粒錠
graphics-oriented language(s)	1	グラフィックス(型/向)言語
	2	写像（型／向）言語
greater-than join	1	比較結合
grouping	1	グループ化
guide / share	1	ガイドシェア
hashing	1	ハッシング（ハッシュ）
	2	ちらし
	3	= RANDOMIZING
height-balanced tree	1	釣合木
hi join	1	高結合
hierarchical model	1	階層（型）モデル
hierarchy	1	階層
host programming language	1	ホスト言語
	2	親言語

identifier		1	識別子
		2	一意名
		3	識別名
image function	ALPH	1	映像関数
		2	イメージ関数
		3	像関数
implement		1	実　働
		2	具　現
		3	作　成
implementation		1	実働化
		2	具現化
index		1	索　引
		2	インデックス
indicator		1	指示子
information algebra		1	情報代数
information bearing structure		1	情報搬送構造
information contents		1	情報内容
inquiry		1	質　問
		2	= INTERROGATION
insert		1	挿　入
		2	追　加
insertion anomaly		1	挿入不都合

install	1	設　置
instance	1	実現値
	2	= OCCURRENCE
integrated	1	結　合
	2	統　合
integrity (control)	1	完全性（制御）
	2	保全性（制御）
	3	統合性（制御）
intention	1	意　志
interface	1	インタフェース
interlock	1	= DEADLOCK
internal schema	1	内部スキーマ
interrogation	1	質　問
	2	= INQUIRY
inversion (inverted-)	1	逆引き
inversion relation	1	転置関係
inverted file	1	逆ファイル
	2	転置ファイル
	3	倒置ファイル
inverted list	1	逆リスト
	2	転置リスト
	3	倒置リスト

invisibility		1	不可視性
		2	不可視可
irreducible relation			
item		1	項 目
join		1	結 合
	S2K	2	= LINK
	ADAB	3	: COUPLING
	TOTA	4	: LINKAGE PATH
	DBTG	5	: SET
	IMS	6	: HIERARCHY
	IMS	7	: SYMBOLIC POINTER
	M204	8	: CROSS REFERENCE
		9	: EQUI-JOIN
		10	: NATURAL JOIN
		11	: GREATER-THAN JOIN
		12	: HI JOIN
		13	: LOW JOIN
journal		1	日 誌
		2	控
		3	= LOG
key		1	(識別の為の)キー
		2	(機密保護の)キー(鍵)

key sensitivity	IMS	1	キーの守備範囲
keyword		1	キーワード
		2	必要語
		3	キー語
		4	予約語
LDB, see logical database			
LDBR, see logical database record			
leaf		1	葉
level		1	レベル
		2	水 準
library		1	ライブラリ
		2	登録集
library function(s)		1	集合関数
		2	= AGGREGATE FUNCTION
link		1	（CHAINの）リンク
		2	継ぎ
link record	DBTG	1	リンクレコード
	IDS	2	= コネクタレコード
		3	結合レコード
links		1	結合子
location mode		1	配置方式
location mode data item	DBTG	1	配置方式用データ項目

lock		1	（機密の）錠
		2	（排他制御）施錠する
		3	ロック
lock(s)		1	錠
lock-name		1	錠 名
locked resource		1	機密資源
log		1	日 誌
		2	= JOURNAL
logical child	IMS	1	論理子
logical database	IMS	1	論理データベース
logical database record	IMS	1	論理データベースレコード
logical parent	IMS	1	論理親
logical record		1	論理レコード
logical twin	IMS	1	論理兄弟
maltivalued dependency		1	多値従属性
man-machine system		1	マンーマシンシステム
		2	人間ー機械系
mandatory	DBTG	1	永 久
		2	必 須
		3	永 続
manual	DBTG	1	手 動
mapping		1	写 像

mapping-oriented language(s)		1	写像(型)言語
mapping-type sublanguage		1	写像(型)準言語
master relation		1	主関係
		2	主要関係
measurement		1	計 測
member	DBTG	1	子
		2	メンバ
member record	DBTG	1	子 レコード
membership class	DBTG	1	格納(STORAGE) 切離し (REMOVE)
metadata		1	メタデータ
		2	データのデータ
model		1	モデル
		2	模 型
modification		1	修 正
N-ary		1	n 項
natural join		1	自然結合
navigation		1	巡 航
		2	: ACCESS PATH
nesting		1	入れ子
network		1	網
network model		1	網(型)モデル
normal form(s)		1	正規形

normalization		1	正規化
null (value)		1	空(値)
occurrence		1	実現値
		2	= INSTANCE
onion-layer language		1	NETWORK -> HIERARCHICAL -> RELATIONAL
operational data		1	運用データ
optimization		1	最適化
optional	DBTG	1	一 時
		2	任 意
ordering		1	(レコードの)順序付け
		2	(レコードの)順番
organization		1	(ファイルの)編成
overflow		1	オーバフロー
		2	あふれ
owner	DBTG	1	親
		2	オーナ
owner record		1	親レコード
owner-coupled set		1	= SET
paired segments	IMS	1	(親と子の)組セグメント
parametric user		1	パラメトリックユーザ
parent	IMS	1	親

PCB, see program communication block

PDB, see physical database

PDBR, see physical database record

phantom		1	仮　想
physical child	IMS	1	物理子
physical database	IMS	1	物理データベース
physical database record	IMS	1	物理データベースレコード
physical parent	IMS	1	物理親
physical record		1	物理レコード
		2	ブロック
physical twin	IMS	1	物理兄弟
piped mode	ALPH		
plex		1	網
		2	＝　ネットワーク
pointer		1	ポインタ
		2	つなぎ
populate		1	注入する
power set			
predicate calculus sublanguage		1	述語論理（型）準言語
primary index		1	主索引
primary key		1	主キー
		2	一次キー
prior		1	逆方向
privacy		1	プライバシー
		2	機密保護

privacy key		1	機密キー
privacy lock		1	機密錠
privacy lock procedure		1	機密錠手続き
privacy transformation, see scrambling			
program communication block	IMS	1	プログラム通信部
program specification block	IMS	1	プログラム定義部
projection		1	射　影
protection		1	（データの）保護
PSB, see program specification block			
qualification		1	修　飾
		2	限　定
query		1	問合せ
		2	質　問
		3	: INTERROGATION
		4	: RETRIEVAL
		5	: INQUIRY
query by example		1	例題による問合せ
query language		1	問合せ語
quiet point		1	休止点
randomizing		1	ランダム化
		2	= HASHING
range		1	範囲一致
		2	変　域

range key		1	範囲キー
range variable	ALPH		
realm		1	領 域
record		1	レコード
	IMS	2	= SEGMENT
	S2K	3	= DATA SET
	S2K	4	: REPEATING GROUP
	REL	5	: TUPLE
record type		1	レコード型
		2	記 録 型
recovery		1	回 復
recursive set		1	再帰親子集合
redundancy		1	冗長(性)
regular relation		1	正則関係
relation		1	関 係
		2	リレーション
		3	ファイル
		4	表
		5	: BASE
		6	: MASTER
		7	: CLASS
		8	: RELATIONSHIP

	9	: REGULAR
	10	: ENTITY
	11	: INVERSION
relation record	1	関係レコード
	2	= LINK RECORD
relational calculus	1	関係論理
relational completeness	1	関係完備
relational database	1	関係データベース
	2	リレーショナルデータベース
relational model	1	関係モデル
	2	リレーショナルモデル
relationally complete	1	関係完備
relationship relation	1	相互関係
relationship	1	関　連
remove	1	切離し
reorganization	1	再編成
repeating group	1	繰返し集団
requirement	1	要　求
reserved word	1	予約語
restart	1	再　開
restriction	1	制　限
	2	制　約
restructuring	1	再構成

retrieval		1	検　索
retrieval usage mode		1	検索使用モード
ring		1	リング
role		1	役　割
roll back		1	後退復帰
	IMS	2	= BACK OUT
roll forward		1	前進復帰
root		1	根
run unit	DBTG	1	実行単位
sabotage		1	サボタージュ
		2	ＮＯＴ＝怠業
scan		1	走　査
schema		1	スキーマ
		2	固有（データ）構造記述
		3	図　式
scrambling			
search		1	探　索
secondary index		1	二次索引
		2	副次索引
secondary key		1	二次キー
		2	副次キー
		3	: ALTERNATE KEY

security		1	安全性
		2	機密保護
		3	機　密
segment		1	（記憶装置の）区域
	IMS	2	セグメント
segment search argument	IMS	1	セグメント探索引数
selection		1	選　択
self-contained language		1	独立言語
		2	自立型言語
self-descriptive		1	自己記述的
semantic model		1	意味モデル
semantics		1	意味（論）
set	DBTG	1	親子集合
		2	= OWNER-COUPLED SET
		3	= COSET
		4	= FANSET
		5	= DATA-STRUCTURE SET
		6	= DATABASE SET
set membership		1	親子関係
set-theoretic operators		1	集合論的演算子
shared control		1	共用制御
		2	= CONCURRENCY CONTROL

shared data		1	共用データ
		2	共有データ
shared lock		1	(データの)共用制御
		2	: EXCLUSIVE LOCK
simple domain		1	単純定義域
		2	原始(不可分)定義域
		3	: NORMALIZED
singular set	DBTG	1	単一親子集合
		2	孤児親子集合
snapshot		1	一時的写し
source file		1	源泉ファイル

SSA, see segment search argument

SSTG, see sub-schema task group

staging		1	せり上げ
status code		1	状態符号
storage		1	記　憶
storage anomaly		1	記憶不整合
storage structure		1	記憶構造
store		1	格　納
structure		1	構　造
		2	構　成
sub-schema task group		1	サブスキーマ作業班

sublanguage	1	準言語
subschema	1	サブスキーマ
	2	個別（データ）構造記述
surrogate		
symbolic pointer		
synonym	1	同　義　（語）
syntax	1	構　文　（論）
syntax rules	1	構文規則
synthesis	1	合　成
table	1	表
target (list)	1	目　標　（部）
	2	目　的
	3	受取側
temporary area / realm	1	一時領域
terminal user	1	端末ユーザ
totally inverted	1	完全転置方式
transaction	1	トランザクション
	2	発　生
	3	処　理　（単位）
	4	業　務

transitive dependency	REL	1	推移従属
tree		1	木
		2	ツリー
tuning		1	性能向上
		2	調　整
tuple		1	組
		2	順　組
		3	: RECORD
tuple identifier		1	組識別子
twin pointer	IMS	1	兄弟ポインタ
twin segment	IMS	1	兄弟セグメント
type		1	型
unary relation			
unit		1	装　置
		2	: DEVICE
update		1	更　新
update anomaly		1	更新不整合
update lock		1	更新錠
update usage mode		1	更新使用モード
usage mode		1	使用モード
user interface		1	ユーザインタフェース
user language		1	利用者言語
user profile		1	ユーザプロファイル

user work(ing) area		1	利用者作業域
user's view		1	利用者視野
		2	ユーザのビュウ
UWA, see user work(ing) area			
validation		1	正当性確認
validity		1	正当性
value		1	値
verification/validation		1	検査／確認
		2	= V / V
view		1	ビュウ
		2	見 方
		3	視 野
		4	視 点
violative operation			
virtual domain		1	仮想域
volume		1	ボリューム
window		1	窓
window set	UDL		
work(ing) area		1	作業域
		2	作業領域

付記5. 関 連 文 献

[ABRIJ74]

Abrial, J. R., "Data Semantics," Proc. IFIP TC-2 Working Conf. on DBMS, Cargese, Corsia, France, April 1974, pp.1-60

[ADIBM76]

Adiba, M., Delobel, C. and Leonard, M., "A Unified Approach for Modelling Data in Logical Data Base Design," Proc. INIP TC-2 Working Conf. on Modelling in DBMS, Freudenstadt, Germany, Jan. 1976, pp.311-338

[ADIBM77A]

Adiba, M. and Delobel, C., "The Problem of the Cooperation Between Different DBMS," Proc. IFIP TC-2 Working Conf. on Architecture and Models in DBMS, Nice, France, Jan. 1977, pp.165-186

[ADIBM77B]

Adiba, M., "A Cooperation System for Heterogeneous Data Base Management Systems," SCH-I-023, IRIA, June 1977, 20p.

[ADIBM78A]

Adiba, M., Chupin, J. C., Demolombe, R., Gardarin, G. and Le Bihan, J., "Issues in Distributed Data Base Management Systems: A Technical Overview," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.89-110

[ADIBM78B]

Adiba, M. and Euzet, C., "A Distributed Data Base System Using Logical Relational Machines," Proc. 4th International Conf. on VLDB. Berlin, Sept. 1978, pp.450-461

[ADIBM78C]

Adiba, M., "Modelling Approach for Distributed Data Bases," Laboratoire D' Informatique, USMG, Grenoble, Jan. 1978

[ADIBM78D]

Adiba, M., "Un Modele Relationnel et une Architecture pour les Systemes de Bases de Donnees Reparties, Application au Projet Polypheme," Universite Scientifique et Medicale de Grenoble Institut National Polytechnique de Grenoble, Sept. 1978

[AGERT79]

Agerwala, T., "Putting Petri Nets to Work," IEEE Computer, Vol. 12, No. 12, Dec. 1979, pp.85-94

[AHOA 74]

Aho, A. V., Hopcroft, J. E. and Ullman, J. D., "The Design and Analysis of Computer Algorithms," Addison-Wesley, 1974

[AHOA 79A]

Aho, A. V., Sagiv, Y. and Ullman, J. D., "The Theory of Joins in Relational Databases," ACM TODS, Vol. 4, No. 3, Sept. 1979, pp.297-314

[AHOA 79B]

Aho, A. V., Beeri, C. and Ullman, J. D., "Efficient Optimization of a Class of Relational Expressions," ACM TODS, Vol. 4, No. 4, Dec. 1979, pp.435-464

[ALSBP76]

Alsberg, P. A. and Day, J. D., "A Principle for Resilient Sharing of Distributed Resources," Proc. the 2nd International Conf. on Software Engineering, San Francisco, California, Oct. 1976, pp.562-570

[ANDER75]

Anderson, R. H., "Advanced Intelligent Terminals as a User's Network Interface," IEEE Compcon '75 Conference, Sept. 1975, pp.25/4.1-6

[ANDER76A]

Anderson, R. H. and Gillogly, J. J., "Rand Intelligent Terminal Agent (RITA): Design Philosophy," R-1809-ARPA, Rand Corporation, Santa Monica, Feb. 1976.

[ANDER76B]

Anderson, R. H. and Gilogly, J. J., "Rand Intelligent Terminal Agent (RITA) as a Network Access Aid," AFIPS Conference Proceedings, Vol. 45, May 1976, pp.501-509

[ANDER77]

Anderson, R. H., Gallegos, M., Gillogly, J. J., Greenberg, R. and Villanueva, "RITA Reference Manual," R-1808-ARPA, Rand Corporation, Sept. 1977.

[ANDRE78]

Andre, E. and Decitre, P., "On Providing Distributed Application Programmers with Control over Synchronization," Proc. Computer Network Protocols, Liege, Belgium, Feb. 1978, pp.D1/1-D1/6

[ANSIX75]

ANSI/X3/SPARC DBMS Study Group, "Interim Report of the Study Group on Data Management Systems," Report 75-02-08, Feb. 1975

[ANSIX76]

ANSI/X3/SPARC DBMS Study Group, "Discussion of Future Directions of ANSI SPARC DBMS Study Group," Proc. 2nd Share Working Conf. on DBMS, Montreal, Canada, April 1976, pp.207-220

[ANTOF78]

Antonacci, F., Dell'Orco, P., Spadavecchia, V. N. and Turtur, A., "AQL: A Problem-Solving Query Language for Relational Data Bases," IBM J. Res. Develop., Vol.22, No. 5, Sept. 1978, pp.541-559

[ARMSW74]

Armstrong, W. W., "Dependency Structures of Data Base Relationships," Proc. IFIP Conference, Stockholm, Aug. 1974, pp.580-583

[ARORA78A]

Arora, A. K. and Carlson, C. R., "The Information Preserving Properties of Relational Database Transformations," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.352-359

[ARORS78B]

Arora, S. K. and Smith, K. C., "Testing Relational Views for Multivalued Dependencies using a Set Theoretic Approach," Proc. Compcon 78 Fall, Washington, D. C., Sept. 78, pp.389-393

[ASTRM75]

Astrahan, M. M. and Chamberlin, D. D., "Implementation of a Structured English Query Language," CACM, Vol. 18, No. 10, Oct. 1975, pp.580-588

[ASTRM76]

Astrahan, M. M., et al., "System R: Relational Approach to Database Management," ACM TODS, Vol. 1, No. 2, June 1976, pp.97-137

[ASTRM79A]

Astrahan, M. M., Blasgen, M. M., Chamberlin, D. D., Gray, J. N., King, W. F., et al., "System R: A Relational Data Base Management System," IEEE Computer, Vol. 12, No. 5, May 1979, pp.42-48

[ASTRM79B]

Astrahan, M. M., et al., "System R: A Relational Database Management System," IEEE COMPCON 79 Spring, S.F., CA., Feb. 1979, pp.72-76

[AVRUI76]

Avbrunin, I. L., "The Navy Laboratory Computer Network (NALCOM)," IEEE Trends and Application of Computer Networks, 1976, pp.1-5

[BABBE79]

Babb, E., "Implementing a Relational Database by means of Specialized Hardware," ACM TODS, Vol. 4, No. 1, Mar. 1979, pp.1-29

[BACHC77A]

Bachman, C. W., "Why Restrict the Modelling Capability of CODASYL Data Structure Sets?," AFIPS Conf. Proc., Vol. 46, May 1977, pp.69-75

[BACHC77B]

Bachman, C. W. and Dale, M., "The Role Concept in Data Models," Proc. 3rd International Conf. on VLDB, Tokyo, Oct. 1977, pp.464-476

[BACHC78A]

Bachman, C., "Commentary on the CODASYL Systems Committees Interim Report on Distributed Database Technology," AFIRS Conf. Proc., Vol. 47, May 1978, pp.919-921

[BACHC78B]

Bachman, C. W., "Provisional Model of Open System Architecture," Proc. 3rd Berkeley Workshop on Distributed Data Management and Computer Networks, San Francisco, Aug. 1978, pp.1-18

[BACHC78C]

Bachman, C. and Canepa, M., "The Session Control Layer of an Open System Interconnection," Proc. COMPCON78 Fall, Washington, D.C., Sept. 78, pp.150-156

[BACHC78D]

Bachman, C. W., "Domestic and International Standards Activities for Distributed Systems," Proc. COMPCON 78 Fall, Washington, D. C., Sept. 1978, pp.140-149

[BACKJ78]

Backus, J., "Can Programming be liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs," CACM, Vol. 21, No. 8, Aug. 1978, pp.613-641

[BADAD79]

Badal, D. Z., "Concurrency Control and Semantic Integrity Enforcement in Distributed Databases," INFOTECH State of the Art Report: Distributed Databases, Vol. 2, 1979, pp.15-27

[BANCF79]

Bancilhon, F., "Supporting View Updates in Relational Data Bases," Proc. of the IFIP TC-2 Working Conf. on Data Base Architecture, Venis, Italy, May 1979, pp.213-234

[BANEJ78A]

Banerjee, J. Hsiao, D. and Baum, R. I., "Concepts and Capabilities of a Database Computer," ACM TODS, Vol. 3, No. 4, Dec. 1978, pp. 347-384

[BANEJ78B]

Banerjee, J. and Hsiao, D. K., "A Methodology for Supporting Existing CODASYL Databases with New Database Machines," Proc. of the 1978 Annual Conf. (ACM78), Washington, D. C., Dec. 1978, pp.925-936

[BANEJ78C]

Banerjee, J. Hsiao, D. and Ng, F. K., "Data Netwo k- A Computer Network of General-Purpose Front-End Computers and Special-Purpose Back-End Database Machines," Proc. of Computer Netwo k Protocols, Liege, Belgium, Feb. 1978, D7-1 D6-12

[BANEJ79]

Banerjee, J. Hsiao, D. K. and Kannan, K., "DBC-A Database Computer for Very Large Databases," IEEE Trans. on Computers, Vol. C-28, No. 6, June 1979, pp.414-429

[BASIV79]

Basili, V. R. and Reiter, R. W. J., "An Investigation of Human Factors in Software Development," IEEE Computer, Vol. 12, No. 12, Dec. 1979, pp.21-38

[BAYER74]

Bayer, R., "Storage Characteristics and Methods for Searching and Addressing," Proc. IFIP Congress 74, Stockholm, Sweden, Aug. 1974, pp.440-444

[BAYER77]

Bayer, R. and Unterauer, K., "Prefix B-Trees," ACM TODS, Vol. 2, No. 1, March 1977, pp.11-26

[BECKL78]

Beck, L. L., "A Generalized Implementation Method for Relational Data Base Sublanguages," Proc. of the 2nd COMPSAC, Chicago, Ill., Nov. 1978, pp.452-457

[BEERC77]

Beeri, C., Fagin, R. and Haward, J. H., "A Complete Axiomatization for Functional and Multivalued Dependencies in Database Relations," Proc. ACM SIGMOD Conf., Toronto, Aug. 1977, pp.47-61

[BEERC78]

Beeri, C., Bernstein, P. A. and Goodman, N., "A Sophisticates Introduction to Database Normalization Theory," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978

[BEERC79]

Beeri, C. and Bernstein, P. A., "Computational Problems Related to the Design of Normal Form Relational Schemas," ACM TODS, Vol. 4, No. 1, Mar. 1979, pp.30-59

[BELFG75]

Belford, G. G., "Research in Network Data Management and Resource Sharing," CAC Document #161, JTSA Document #5508, Center for Advanced Computation, Univ. of Illinois at Urbana- Champaign, May 1975.

[BENOJ73]

Benoit, J. W., "Evolution of Network User Services - The Network Recource Manager," IEEE Computer Network: Trends and Applications, New York, 1973, pp.21-24

[BENOJ74]

Benoit, J. W., Graf-Webster, E., "REX: A Resource Location and Acquisition Service for the ARPA Computer Network," MTP-387, The MITRE Corporation, Jan. 1974, p.23

[BERND79]

Bernard, D., "Management Issues in Cooperative Computing," ACM Computing Surveys, Vol. 11, March 1979, pp.4-17

[BERNP75]

Bernstein, P. A., Swenson, J. R. and Tsichritzis, D., "A Unified Approach to Functional Dependencies and Relations," Proc. of ACM SIGMOD Workshop on Management of Data, SF., May 1975, pp.237-245

[BERNP76]

Bernstein, P. A., "Synthesizing Third Normal Form Relations from Functional Dependencies," ACM TODS, Vol. 1, No. 4, Dec. 1976, pp.277-298

[BERNP77A]

Bernstein, P. A., Goodman, N., Rothnie, J. B. and Papadimitriou, C. A., "Analysis of Serializability in SDD-1: A System for Distributed Databases (The Fully Redundant Case)," TR-CCA-77-05, Computer Corporation of America, June 1977, 74P.

[BERNP77B]

Bernstein, P. A., Shipman, D. W., Rothnie, J. B. and Goodman, N., "The Concurrency Control Mechanism of SDD-1: A System for Distributed Databases," Technical Report CCA-77-09, Computer Corporation of America, Cambridge, Dec. 1977

[BERNP78A]

Bernstein, P. A., Pothnie, J. B., Goodman, N. and Paradimitriou, C. A., "The Concurrency Control Mechanism of SDD-1: A System for Distributed Databases (The Fully Redundant Case)," IEEE Trans. on SE, Vol. SE-4, No. 3, May 1978, pp.154-168

[BERNP78B]

Bernstein, P. A. and Shipman, D. W., "A New Analysis of Concurrency Control in SDD-1: A System for Distributed Databases," Technical Report CCA-78-08, CCA., June 1978

[BERNP78C]

Bernstein, P. A. and Shipman, D. D., "A Formal Model of Concurrency Control Mechanisms for Database Systems," Proc. of the 3rd Berkeley Workshop on Distributed Data Management and Computer Networks, San Francisco, Aug. 1978, pp.189-205

[BERNP79A]

Bernstein, P. A. and Goodman, N., "Approaches to Concurrency Control in Distributed Data Base Systems," AFIPS Conf. Proc., Vol. 48, New York, June 1979, pp.813-820

[BERRP78]

Berra, P. B., "Recent Development in Data Base and Information Retrieval Hardware Architecture," Proc. of COMPSAC, Chicago, Ill., Nov. 1978, pp.698-703

[BERRP79]

Berra, P. B. and Oliver, E., "The Role of Associative Array Processors in Data Base Machine Architecture," IEEE Computer, Vol. 21, No. 3, March 1979, pp.53-61

[BILLH76]

Biller, H., Glatthaar, W. and Neuhold, E. J., "On the Semantics of Data Bases: The Semantics of Data Manipulation Languages," Proc. IFIP TC-2 Working Conf. on Modelling in Database Management Systems, Freudenstadt, Jan. 1976, pp.239-267

[BILLH77A]

Biller, H. and Neuhold, E. J., "Concepts for the Conceptual Schema," Proc. IFIP TC-2 Working Conf. on Architecture and Model in DBMS, Jan. 1977, pp.1-30

[BILLH77B]

Biller, H. and Neuhold, E. J., "An Example of a Correct Data Translation Using a High Level Data Definition Language," Universitat Stuttgart, Jan. 1977, 23P.

[BILLH78A]

Biller, H. and Neuhold, E. J., "Semantics of Data Bases: The Semantics of Data Models," Information Systems, Vol. 3, No. 1, 1978, pp.11-30

[BILLH78B]

Biller, H. and Neuhold, E. J., "Remarks on the Comments of B. Langefors on Our Paper: Semantics of Data Bases: The Semantics of Data Models," Information Systems, Vol. 3, No. 1, 1978, pp.35-36

[BILLH78C]

Biller, H., "On the Notion of Irreducible Relations," Univ. of Stullgart, 1980.

[BILLH79]

Biller, H., "On the Equivalence of Data Base Schemas a Semantic Approach to Data Translation," Information Systems, Vol. 4, No. 1, 1979, pp.35-48

[BIRSE76]

Birss, E. W., and Fry, J. P., "Generalized Software for Translating Data," AFIPS Conference Proceedings, May 1976, pp.889-897

[BLASM77]

Blasgen, M. W. and Eswaran, K. P., "Storage and Access in Relational Data Bases," IBM Systems Journal, Vol. 16, No. 4, 1977, pp.363-377

[BOBRD75]

Bobrow, D. G. and Collins, A., "Representation and Understanding," Academic Press, Inc., New York, 1975, p.427

[BONCR77A]

Bonczek, R. H. and Whinston, A.B., "A Generalized Mapping Language for Network Data Structure," Information Systems, Vol. 2, No. 4, 1977, pp.171-185

[BONCR77B]

Bonczek, R. H., Cash, J. I. and Whinston, A. B., "A Transformational Grammar- Based Query Processor for Access Control in a Planning System," ACM TODS, Vol. 2, No. 4, DEC. 1977, pp.326-338, pp.143-154

[BONCR79]

Bonczek, R. H. and Whinston, A. B., "The Integration of Network Data Base Management and Problem Resolution," Information Systems, Pergamon Press, Vol. 4, No. 2, 1979, pp.526-534

[BORKS78]

Borkin, S. A., "Data Model Equivalence," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978

[BOYCR75]

Boyce, R. F., Chamberlin, D. D., King, W. F. and Hammer, M. M., "Specifying Queries as Relational Expressions: The Square Data Sublanguage," CACM, Vol. 18, No. 11, Nov. 1975, pp.621-628, pp. 125-148

[BRACG76]

Bracchi, G., Paolini, P. and Pelagatti, G., "Binary Logical Associations in Data Modelling," Proc. IFIP TC-2 Working Conf. on Modelling in DBMS, June 1976,

[BRADJ78A]

Bradley, J., "Operations Data Bases," Proc. the 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.164-176

[BRADJ78B]

Bradley, J., "An Extended Owner-Coupled Set Data Model and Predicate Calculus for Database Management," ACM TODS, Vol. 3, No. 4, Dec. 1978, pp.385-416

[BRAYO79]

Bray, O. and Thurber, K. J., "What's Happening with Data Base Processors?" Datamation, Jan. 1979, pp.146-156

[BREIH79]

Breitwieser, H. and Kersten, U., "Transaction and Catalog Management of the Distributed File Management System DISCO," 5th VLDB Rio De Janeiro, Brazil, Oct. 1979, pp.340-350

[BREUB79]

Breutman, B., Falkenberg, E. and Maner, R., "CSL: A Language for Defining Conceptual Schemas," Proc. of the IFIP Working Conf. on Database Architecture, Venice, Italy, June 1979, pp.220-239

[BUBEJ76]

Bubenko, J. A., Jr. and Berild, S., "From Information Requirements to DBTG-Data Structures," Proc. Conf. on Data: Abstraction, Definition and Structure, Salt Lake City, March 1976, pp.73-85

[BUBEJ77A]

Bubenko, J. A., Jr., "The Temporal Dimension in Information Modelling," Proc. IFIP TC-2 Working Conf. on Architecture and Models in DBMS, Jan. 1977, pp.93-118

[BUBEJ77B]

Bubenko, J. A., Jr., "IAM: An Inference Abstract Modelling Approach to Design of Conceptual Schema," Proc. ACM SIGMOD International Conf. on Management of Data, Toronto, Canada, Aug. 1977, pp.62-74

[BUBEJ77C]

Bubenko, J. A., Jr., "Validity and Verification Aspects of Information Modelling," Proc. 3rd International Conf. on VLDB, Tokyo, Japan, Oct. 1977, pp.556-565

[BUFFP77]

Buffet, P., "Pascaline, A Multidisciplinary European Data Base," 1st International On-Line Information Meeting, London, Dec. 1977, pp.143-149

[BUNEP79]

Buneman, P. and Frankel, R. E., "FQL-A Functional Query Language (A Preliminary Report)," Proc. of the ACM SIGMOD International Conf. on Management of Data, Boston, MA., May 1979, pp.52-58

[BUNEO79]

Buneman, O. P. and Clemons, E. K., "Efficiently Monitoring Relational Databases," ACM TODS, Vol. 4, No. 3, Sept. 1979, pp. 368-382

[BURGJ77]

Burger, J. F., "Data Base Semantics in the EUFID System," Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, Berkeley, Calif., May 1977, pp.202-214

[BURKW76]

Burkhard, W. A., "Hashing and Trie Algorithms for Partial Match Retrieval," ACM TODS, Vol. 1, No. 2, June 1976, pp.175-187

[CADIJ75]

Cadiou, J. M., "On Semantic Issues in the Relational Model of Data," Proc. International Symposium on Mathematical Foundations of Computer Science, Gdansk, Roland, Sept. 1975, pp.23-38

[CADIJ79]

Cadiou, J. M. and Chang, C. L., "A Surface Schema Method for Generating English Sentences of Formal Queries in Relational Data Bases," ETL AI Symposium, Tokyo, Aug. 1979

[CANAR74]

Canady, R., Harrison, R. D., Ivie, E. L., Ryder, J. L. and Wehr, L. A., "A Back-end Computer for Data Base Management," CACM, Vol. 17, No. 10, Oct. 1974, pp.575-582

[CANNR78]

Canning, R. D., "Installing a Data Dictionary," EDP Analyzer, Vol. 16, Jan. 1978, 13P

[CARDA79]

Cardenas, A. F., "Data Base Management Systems," Allyn and Bacon, Inc., Boston, MA., 1979, pp.519

[CARLC76]

Carlson, C. R. and Kaplan, R. S., "A Generalized Access Path Model and Its Application to a Relational Data Base System," Proc. International Conf. on Management of Data, ACM-SIGMOD, June 1976, pp.143-154

[CASER72]

Casey, R. G., "Allocation of Copies of a File in an Information Network," AFIPS Conference Proceedings, SJCC, 1972, pp.617-621

[CASER73]

Casey, R. G., "Design of Tree Networks for Distributed Data," AFIPS Conference Proceedings, May 1973, pp.251-257

[CASHP74]

Cashim, P., "Datapac Standard Network Protocol," INWG General Note #77, Nov. 1974

[CCA77]

Computer Corporation of America, (CCA), "A Distributed Database Management System for Command and Control Applications-Semi-Annual Technical Report," Technical Report CCA-77-06, July 1977, p.121

[CEC 76]

Commission of the European Communities, "EURONET News Issue No. 2," August 1976

[CEC 77A]

Commission of the European Communities, "EURONET News Issue No. 7," July 1977

[CEC 77B]

Commission of the European Communities, "EURONET News Issue No. 8," October 1977

[CEC 77C]

Commission of the European Communities, "EURONET News Issue No. 9," December 1977

[CERCN77]

Cercone, N., "A Heuristic Morphological Analyzer for Natural Language Understanding Programs," Proc. of COMPSAC, Chicago, Nov. 1977, pp.676-682

[CHAMD74]

Chamberlin, D. D., "SEQUEL: A Structured English Query Language," Proc. 1974, ACM SIGMOD Working Conf. on Data Description, Access and Control, May 1974

[CHAMD75]

Chamberlin, D. D., J. N. and Traiger, J. L., "Views, Authorization, and Locking in a Relational Data Base System," AFIPS Conf. Proc., Vol. 44, May 1975, pp.425-430

[CHAMD76A]

Chambarlin, D. D., "Relational Database Management Systems and Control," Computing Surveys, Vol. 8, No. 1, March 1976, pp.43-66

[CHAMD76B]

Chamberlin, D. D. and R. F. Buyce, "SEQUEL: A Structured English Query Language," Proc. ACM-SIGMOD Workshop on Data Description, Access and Control, May 1976, pp.249-264

[CHAMD76C]

Chamberlin, D. D., Astrahan, M. M., et. al., "SEQUEL2: A Unified Approach to Data Definition, Manipulation and Control," IBM Journal of Research and Development, Vol. 20, No. 6, Nov. 1976

[CHAMG78]

Champine, G. A., "Four Approaches to a Data Base Computer," Datamation, Dec. 1978, pp.101-109

[CHAMG79A]

Champine, G. A., "Trends in Data Base Processor Architecture," IEEE COMPCON 79 Spring, S. F., Feb. 1979, pp.69-71

[CHAMG79B]

Champ, G. A., "Current Trends in Data Base Systems," IEEE COMPSAC 79, Spring, S.F., Feb. 1979, pp.177-180

[CHANA77]

Chandra, A. K. and Merlin, P. M., "Optimal Implementation of Conjunctive Queries in Relational Data Bases," Proc. of the 9th Annual ACM Symposium on Theory of Computation, Boulder. Colorado, May 1977, pp.77-89

[CHANC76]

Chang, C. L., "DEDUCE --- A Deductive Query Language for Relational Data Base," Pattern Recognition and Artificial Intelligence (Chen, C. H. Eds.), Academic Press, 1976, pp.108-134

[CHANC78]

Chang, C. L., "DEDUCE 2: Further Investigations of Deduction in Relational Data Bases," Logic and Data Bases (Gallaire, H. and Minker, J. Eds.) Plenam Press, 1978, pp.201-236

[CHANC79A]

Chang, C. L., "Resolution Plans in Theorem Proving," IJCAI-79, Tokyo, Japan, Aug. 1979, pp.143-148

[CHANH78]

Chang, H., "On Bubble Memory and Relational Data Base," Proc. of the 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.207-229

[CHANH78B]

Chang, H., "Interpretation of Query-By-Example Language in Bubble Hardware Language," Proc. of the 1978 Annual Conf. (ACM78), Washington, D.C., Dec. 1978, pp.919-924

[CHANS76A]

Chang, S. K., Obrien, M., Read, J., Borovec, R., Chang, W. H. and Ke, J. S., "Design Considerations of a Database System in a Clinical Network Environment," AFIPS Conf. Proc., Vol. 45, May 1976, pp.277-286

[CHANS76B]

Chang, S. K. and McCormick, B. H., "An Intelligent Coupler for Distributed Database Systems," Dept. of Information Engineering, Univ. of Illinois at Chicago Circle, Chicago, Ill. 60680, 1976

[CHANS78A]

Chang, S. K. and Cheng, W. H., "Database Skeleton and Its Application to Logical Database Synthesis," IEEE Trans. on Software Engineering, Vol. SE-4, No. 1, Jan. 1978, pp.18-30

[CHANS78B]

Chang, S. H. and Cheng, W. H., "Database Skeleton and Its Application to Fuzzy Query Translation," IEEE Trans. on Software Engineering, Vol. SE-4, No. 1, Jan. 1978, pp.31-44

[CHENP73]

Chen, P. P., "Optimal File Allocation in Multi-Level Storage Systems," AFIPS Conference Proceedings, May 1973, pp.277-282

[CHENP76]

Chen, P.P.S., "The Entity-Relationship Model-Toward a Unified View of Data," ACM TODS, Vol. 1, No. 1, March 1976, pp.9-36

[CHENP77]

Chen, P.P.S., "The Entity-Relationship Model-A Basis for the Enterprise View of Data," AFIPS Conf. Proc., Vol. 47, May 1977, pp.77-84

[CHENT78]

Chen, T. C., Lum, V. Y. and Tung, C., "The Rebound Sorter: An Efficient Sort Engine for Large Files," Proc. of the 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.312-318

[CHENW78]

Cheng, W. H., "Optimization Techniques in Designing Relational Database Systems," Ph. D. Dissertation, Univ. of Illinois at Chicago Circle, 1978

[CHUPJ76]

Chupin, J. C., Seguin, J. and Sergeant, G., "Distributed Applications on Heterogeneous Network," Proc. 3rd Annual Symposium on Computer Architecture, Florida, USA, Jan. 1976

[CHUW73]

Chu, W. W., "Optimal File Allocation in a Computer Network," Computer Communications Networks (N. Abramson and F. Juo, Eds.), Prentice Hall, 1973, pp.82-94

[CHUW74]

Chu, W. W. and Ohlmacher, G., "Avoiding Deadlock in Distributed Data Bases," Proc. of the ACM National Symposium, 1974

[CHUW75]

Chu, W. W. and Nahouraii, E., "File Directory Design Considerations for Distributed Database," Proc. International Conf. on VLDB, Farmingham, Massachusetts, 1975, pp.543-545

[CHUW76]

Chu, W. W., "Performance of File," Directory Systems for Data Bases in Star and Distributed Networks," AFIPS,Conf. Proc., Vol. 45, 1976, pp.577-587

[CHUW79]

Chu, W. W. and Hurley, D., "A Model for Optimal Query Processing for Distributed Data Bases," IEEE COMPCON 79 Spring, Feb. 1979, pp.116-122

[CHUW79B]

Chu, W. W., "Design Considerations of File Directory Systems for Distributed Databases," Infotech State of the Art Report: Distributed Databases, Vol. 2, 1979, pp.73-85

[CLARK77]

Clark, K. L. and Tarnlund, S. A., "A First Order Theory of Data and Programs," Information Processing 77, North-Holland, 1977, pp.939-944

[CLARK78]

Clark, J. D. and Hoffer, J. A., "A Procedure for the Determination of Attribute Access Probabilities," ACM SIGMOD International Conf. on Management of Data, Austin, Texas, May 1978, pp.110-117

[CLAYB77]

Claybrook, B. G., "A Facility for Defining and Manipulating Generalized Data Structures," ACM TODS, Vol. 2, No. 4, Dec. 1977, pp.370-406

[CLEME78]

Clemons, E. K., "An External Schema Facility to Support Data Base Update," Databases: Improving Usability and Responsiveness, Academic Press, 1978, pp.371-398

[CLEME79]

Clemons, E. K., "Design of a Prototype ANSI/SPARC Three-Schema Data Base System," AFIPS Conf. Proc., Vol. 48, New York, June 1979, pp.689-696

[CLIPW76]

Clipsham, W. W., Glave, E. E. and Narraway, M. L., "DATAPAC Network Overview," ICCC 1976, pp.131-136

[CODAS71]

Data Base Task Group, "Data Base Task Group to the CODASYL Programming Language Committee," ACM, April 1971

[CODAS73]

CODASYL, "CODASYL Data Description Language Journal of Development," June 1973, NBS Handbook 113, (1974)

[CODAS75]

"A Progress Report on the Activities of the CODASYL End User Facility Task Group," June 1975

[CODAS76]

CODASYL COBOL "Journal of Development 1976," Canadian Government 110-GP-1D, 1976

[CODAS78A]

The CODASYL System Committee, "Distributed Data Base Technology-An Interim Report of the CODASYL Systems Committee," AFIPS Conf. Proc., Vol. 47, May 1978, pp.909-917

[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

[CODDE71A]

Codd, E. F., "Further Normalization of the Relational Data Base Model," Courant Computer Science Symposium, 6, "Data Base Systems," New York City, May 1971.

[CODDE71B]

Codd, E. F., "Relational Completeness of Data Base Sublanguages," Courant Computer Science Symposium 6, "Data Base Systems," New York City, May 1971, pp.65-98

[CODDE74A]

Codd, E. F., "Recent Investigations into Relational Data Base Systems," Proc. IFIP Congress 1974, Aug. 1974, pp.1017-1021

[CODDE74B]

Codd, E. F., "Seven Steps to Rendezvous with the Casual User," Proc. IFIP Working Conf. on Data Base Management, 1974, pp.179-200

[CODDE79]

Codd, E. F., "Extending the Database Relational Model to Capture more Meaning," ACM TODS, Vol. 4, No. 4, Dec. 1979, pp.397-434

[COMBP75]

Comba, P. G., "Needed: Distributed Control," Proc. International Conf. on VLDB, 1975, pp.364-375

[COMED78]

Comer, D., "The Difficulty of Optimum Index Selection," ACM TODS, Vol. 3, No. 4, Dec. 1978, pp.440-445

[COMED79]

Comer, D., "The Ubiquitous B-tree," ACM Computing Surveys, Vol. 11, No. 2, June 1979, pp.121-137

[CUADC75]

Cuadra, C. A., "SDC Experiences with Large Data Bases," Journal of Chemical Information and Computer Sciences, Vol. 15, No. 1, 1975, pp.48-51

[CURTR76]

Curtice, R. M., "The Outlook for Data Base Management," Datamation, April 1976, pp.46-49

[DALEA76A]

Dale, A. G. and Lowenthal, E. I., "End-User Interfaces for Data Base Management Systems," Proc. of the 2nd Share Working Conf. on ABMS, Montreal, Canada, April 1976, pp.81-99.

[DALEA76B]

Dale, A. G. and Dale, N. B., "Schema and Occurrence Structure Transformations in Hierarchical Systems," Proc. International Conf. on Management of Data, Washington D. C., June 1976, pp.157-168

[DALEA77]

Dale, A. G. and Dale, N. B., "Main Schema-External Schema Interaction in Hierarchically Organized Data Bases," Proc. ACM SIGMOD International Conf. on the Management of Data, Tronto, Canada, Aug. 1977, pp.102-110

[DALEA79]

Dale, A. G., "Database Management Systems Development in the USSR," ACM Computing Surveys, Vol. 11, No. 3, Sept. 1979, pp.213-226

[DATEC77]

Date, C. J., "An Introduction to Data Base System," (Second Edition), Addison-Wesley, June 1977

[DATEC79]

Date, C. J., "Locking and Recovery in a Shared Database System: An Application Programming Tutorial," Proc. of the 5th VLDB, Rio De Janeiro, Brazil, Oct. 1979, pp.1-15

[DAVER78A]

Davenport, R. A., "Distributed or Centralized Data Base," Computer Journal, Vol. 21, No. 1, Feb. 1978, pp.7-14

[DAVER78B]

Davenport, R. A., "Distributed Database Technology- A Survey," Computer Networks, Vol. 2, No. 3, July 1978, pp.155-167

[DAVER78C]

Davenport, R. A., "Integrity in Distributed Database Systems," EUROCOMP 78, 1978, pp.751-773

[DAVER79]

Davenport, R. A., "Design of Distributed Database Systems," INFOTECH State of the Art Report: Distributed Databases, Vol. 2, 1979, pp.87-114

[DAYAU78]

Dayal, U. and Bernstein, P. A., "On the Updatability of Relational Views," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.368-377

[DEANV79]

De Antonellis, V., De Cindio, F., Antoni, G. D. and Manri, G., "Use of Bipartite Graphs as a Notation for Data Bases," Information Systems, Vol. 4, No. 2, 1979, pp.137-141

[DEBLJ78]

Deblasis, J. P. and Johnson, T. H., "Review of Data Base Administrators Functions from a Survey," Proc. of the International Conf. on Management of Data, Austin, Texas, May 1978, pp.101-109

[DEHEC76]

Deheneffe, C. and Hennebert, H., "NUL: A Navigational User's Language for a Network Structured Data Base," Proc. ACM SIGMOD International Conf. on Management of Data, Washington D. C., June 1976, pp.135-142

[DELIA79]

Deliyanni, A. and Kowalski, R. A., "Logic and Semantic Networks," CACM, Vol. 22, No. 3, Mar. 1979, pp.184-192

[DELOC73]

Delobel, C. and Casey, R. G., "Decomposition of a Data Base and the Theory of Boolean Switching Functions," IBM Research and Development, Vol. 17, No. 5, Sept. 1973, pp.374-386

[DELOC78A]

Delobel, C., "Data Base Theory and Modelling-Theoretical and Practical Aspects," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.112

[DELOC78B]

Delobel, C., "Normalization and Hierarchical Dependencies in the Relational Data Model," ACM TODS, Vol. 3, No. 3, Sept. 1978, pp.201-222

[DENND79]

Denning, D. E. and Denning, P. J., "Data Security," ACM Computing Survey, Vol. 11, No. 3, Sept. 1979

[DENNI79]

Dennis, J. B., "The Variety of Data Flow Computers," Proc. of the International Conf. on Distributed Computing Systems, Huntsville, Alabama, Oct. 1979, pp.430-439

[DENNP78]

Denning, P. J. and Slutz, D. R., "Generalized Working Sets for Segment Reference Strings," CACM, Vol. 21, No. 9, Sept. 1978, pp.750-759

[DEON74]

Deo, N., "Graph Theory with Applications to Engineering and Computer Science," Prentice-Hall, 1974, p.478

[DEPPM76]

Deppe, M. E. and Fry, J. P., "Distributed Databases: A Summary of Research," Computer Networks, Vol. 1, No. 1, Sept. 1976, pp.130-138

[DEWID76A]

De Witt, D. J., "DIRECT-A Multiprocessor Organization for Supporting Relational Database Management Systems," IEEE Trans. on Computers, Vol. C-28, No. 6, June 1979, pp.395-406

[DEWID79B]

De Witt, D. J., "Query Execution in Direct," Proc. of the ACM SIGMOD International Conf. on Management of Data, Boston, MA., May 1979, pp.13-22

[DIJKE78]

Dijkstra, E. W., Lamport, L., Martin, A. J., Scholten, C. S. and Steffens, E. F. M., "On-the-Fly Garbage Collection: An Exercise in Cooperation," CACM, Vol. 21, No. 11, Nov. 1978, pp.966-975

[DOLKD77]

Dolk, D. R. and Loomis, M. E., "A Methodology for the Design of Generalized Query Processors for CODASYL Data Bases," Proc. of COMPSAC, Chicago, NA., 1977, pp.260-266

[DORKE77]

Dorkenoo, E. S., Lemaitre, M. and Lemoine, M., "A Procedural Language for the Relational Data Base Management System "SYNTEX"," Proc. of IFIP Congress 77 (Information Processing 77), Toronto, Aug. 1977, pp.453-457

[DURCR75]

Durchholz, R., "Relation Representation by Tables and By Functions," Information Systems, Vol. 1, No. 3, 1975, pp.91-96

[DURCR77]

Durchholz, R., "Types and Related Concepts," Proc. International Computing Symposium 1977, Liege, Belgium, April 1977, pp.31-38.

[EGELJ75]

Egeland, J., "The SUNY Biomedical Communication Network: Bull. Med. Libr. Assoc. 63(2), April 1975, pp.189-194

[ELIZJ76]

Elizabeth, J., Feinler, J., Et Al., "ARPANET Resource Handbook," SRI, Network Information Center, Menlo. Park, Calif., Dec. 1976, 858p.

[ELLIC77]

Ellis, C. A., "A Robust Algorithm for Updating Duplicate Databases," Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Network, May 1977, pp.146-158

[ELMAR79]

El - Masri, R. and Wiederhold, G., "Data Model Integration Using the Structural Model," Proc. of the ACM. SIGMOD. International Conf. on Management of Data, Boston, MA., May 1979, pp.191-202

[EPSTR77]

Epstein, R., "A Tutorial on Ingres," Memorandum No. UCB/ERL M77 125, Electronics Research Lab., Univ. of California, Berkeley, Dec. 1977

[EPSTR78]

Epstein, R., Stonebraker, M. and Wong, E., "Distributed Query Processing in a Relational Data Base System," UCB/ERL M78/18, Electronic Research Lab., UC., Berkeley, April 1978

[ESROE73]

Esro/Eldo, "The Space Documentation Service a Progress Report," ESRO/ELDO Bulletin No. 23, Nov. 1973, p.8

[ESWAK75]

Eswaran, K. P. and Chamberlin, D. D., "Functional Specifications of a Subsystem for Data Base Integrity," Proc. of the VLDB, Fragingham, Sept. 1975, pp.48-68

[ESWAK76]

Eswaran, K. P., Gray, J.N., Lorie, R. A. and Traiger, I. L., "The Notions of Consistency and Predicate Locks in a Database System," CACM, Vol. 19, No. 11, Nov. 1976, pp.624-633

[EUSID77]

Eusidic, D., "Newsidic Quarterly No. 25," Autumn 1977, p.18

[FADOR75A]

Fadous, R., "Mathematical Foundations for Relational Data Bases," Ph. D. Dissertation, Department of Computer Science, Michigan State University, 1975

[FADOR75B]

Fadous, R. and Forsyth, J., "Finding Candidate Keys for Relational Data Bases," Proc. 1975, ACM SIGMOD International Conf. on the Management of Data, May 1975

[FAGIR77A]

Fagin, R., "Multivalued Dependencies and a New Normal Form for Relational Databases," ACM TODS, Vol. 2, No. 3, Sept. 1977, pp.262-278

[FAGIR77B]

Fagin, R., "Functional Dependencies in a Relational Database and Propositional Logic," IBM Journal of Research and Development, Vol. 21, No. 6, DEC. 1977, pp.534-544

[FAGIR77C]

Fagin, R., "The Decomposition Versus Synthetic Approach to Relational Database Design," Proc. 3rd International Conf. on VLDB, Tokyo, Oct. 1977, pp.441-446

[FAGIR78]

Fagin, R., "On an Authorization Mechanism," ACM TODS, Vol. 3, No. 3, Sept. 1978, pp.310-319

[FAGIR79]

Fagin, R., "Normal Forms and Relational Database Operators," Proc. of the ACM SIGMOD International Conf. on Management of Data, Boston, MA., May 1979, pp.153-160

[FALKE76A]

Falkenberg, E., "Concepts for Modelling Information," Proc. IFIP Working Conf. on Modelling in DBMS, 1976, pp.95-109

[FALKE76B]

Falkenberg, E., "Significations: The Key to Unify Data Base Management," Information Systems, Vol. 2, No. 1, 1977, pp.19-28

[FALKE77A]

Falkenberg, E., "Concepts for the Coexistence Approach to Data Base Management," Proc. IFIP TC-2 Working Conf. on Architecture and Model in DBMS, Jan. 1977, pp.39-50

[FALKE77B]

Falkenberg, E., "Coexistence and Translation of Data," Proc. of the 3rd International Conf. on VLDB, Tokyo, Japan, Oct. 1977, pp.314-317

[FALKE77C]

Falkenberg, E., "Concepts for the Coexistence Approach to Data Base Management," Proc. International Computing Symposium 1977, 1977, pp.39-50

[FALKF78]

Falkenberg, F. D., "Data Models: The Next Five Years," INFOTECH State of the Art, Database Technology: 2, 1978, pp.53-68

[FARRJ76]

Farrell, J., "The Datacomputer -- A Network Data Utility," Proc. Berkeley Workshop on Distributed Data Management and Networks, Berkeley, May 1976, pp.352-364

[FIKER77]

Fikes, R. and Hendrix, G., "A Network - Based Knowledge Representation and its Natural Deduction System," Proc. of the 5th IJCI, Cambridge, MA., Aug. 1977, pp.235-246

[FLORA78]

Flory, A. and Kouloumdjian, J., "A Model and a Method for Logical Data Base Design," Proc. of the 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.333-341

[FORDH78]

Fordick, H. C., Schantz, R. E. and Thomas, R. H., "Operating Systems for Computer Networks," IEEE Computer, Vol. 11, No. 1, Jan. 1978, pp.48-57

[FOUCO78]

Foucaut, O. and Rolland, C., "Concepts for Design of an Information System Conceptual Schema and Its Utilization in the REMORA Project," Proc. of the 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.342-350

[FRYJ76]

Fry, J. P. and Sibley, E. H., "Evolution of Data Base Management Systems," ACM Computing Surveys, Vol. 8, No. 1, March 1976, pp.7-42

[FRYJ78A]

Fry, J. P., Lowenthal, E., Shoshani, A., Birss, E., et Al., "An Assessment of the Technology for Data - and Program - Related Conversion," AFIPS Conf. Proc., Vol. 47, May 1978, pp.887-907

[FRYJ 78B]

Fry, J. P. and Teorey, T. J., "Design & Performance Tools Improving Database Usability & Responsiveness," Proc. of the International Conf. on Databases: Improving Usability and Responsiveness, Haifa, Israel, Aug. 1978, pp.151-189

[FURTA77]

Furtado, A. L. and Kershberg, L., "An Algebra of Quotient Relations," Proc. of ACM SIGMOD International Conf. on Management of Data, Toronto, Aug. 1977, pp.1-8

[FURTA78]

Furtado, A. L., "Formal Aspects of the Relational Model," Information Systems, Vol. 3, No. 2, 1978, pp.131-140

[FURTA79]

Furtado, A. L., Sevcik, K. C., Dos Santos, C. S., "Permitting Updates Through Views of Data Bases," Information Systems, Vol. 4, No. 4, 1979, pp.269-283

[FURUK77]

Furukawa, K., "A Deductive Question Answering System on Relational Data Bases," Proc. 5th International Joint Conf. on Artificial Intelligence, MIT, Cambridge, Aug. 1977, pp.59-66

[FUTOT78]

Futo, I., Darvas, F., and Szeredi, P., "The Application of Prolog to the Development of QA and DBM System," Logic and Data Bases (Gallaire, H. and Minker, J. Eds.), Plenum Press, 1978, pp.347-376

[GALLH78]

Gallaire, H., Minker, J., and Nicolas, J. M., "An Overview and Introduction to Logic and Data Bases," Logic and Data Bases, (Gallaire, H., and Minker, J. Eds.), Plenum Press, 1978, pp.3-30

[GARCH78]

Garcia - Molina, H., "Distributed Database Coupling," 3rd USA - Japan Computer Conf., Oct. 1978, pp.75-79

[GARDG76A]

Gardarin, G. and Spaccapietre, S., "Integrity of Data Bases: A General Lockout Algorithm with Deadlock Avoidance," Proc. IFIP TC-2 Working Conf. on Modelling in DBMS, Freudenstadt, Jan. 1976, pp.395-411

[GARDG76B]

Gardarin, G. and Le Biahn, J., "Proposal for a Common Description Model of Distributed Data Bases," MOD-I-008, IRIA, Nov. 1976

[GARDG77A]

Gardarin, G. and Le Bihan, J., "An Approach Towards a Virtual Data Base Protocol for Computer Networks," IRIA SCH-I-022, 1977

[GARDG77B]

Gardarin, G. and Lebeux, P., "Scheduling Algorithms for Avoiding Inconsistency in Large Databases," Proc. 3rd International Conf. on VLDB, Tokyo, Oct. 1977, pp.501-506

[GARDG79A]

Gardarin, G., "A Unified Architecture for Data and Message Management," AFIPS Conf. Proc., Vol. 48, New York, June 1979, pp.681-688

[GARDG79B]

Gardarin, G. and Jouve, M., "The Execution Kernel of a Distributed Data Base Management System," Proc. of the IFIP TC-2 Working Conf. on Data Base Architecture, Venice, May 1979, pp.3-21

[GARDG79C]

Gardarin, G. and Chu, W. W., "A Reliable Distributed Control Algorithm for Updating Replicated Databases," Proc. of the 6th Data Communication Symp., Pacific Grove, CA., Nov. 1979, pp.42-51

[GELEE79]

Gelenbe, E. and Sevcik, K., "Analysis of Update Synchronization for Multiple Copy Data Bases," IEEE Trans. on Computers, Vol. C-28, No. 10, Oct. 1979, pp.737-747

[GERRR75]

Gerritsen, R., "A Preliminary System for the Design of DBTG Data Structures," CACM, Vol. 18, No. 10, Oct. 1978, pp.551-557

[GHOSS77]

Ghosh, S. P., "Data Base Organization for Data Management," Academic Press, New York, 1977, p.376

[GRACH79]

Garcia - Molina, H., "Centralized Control Update Algorithms for Fully Redundant Distributed Databases," The Distributed Computing Systems, Huntsville, Arabama, Oct. 1979, pp.699-705

[GRADD75]

Gradwell, D. J. L., "Why Data Dictionaries?," Database Journal, Vol. 16, No. 2, pp.15-18

[GRAYJ75]

Gray, J. N., Lorie, R. A. and Putzolu, G. R., "Granualarity of Locks in a Shared Data Base," Proc. International Conf. on VLDB, 1975, pp.428-451

[GREED78]

Greenblatt, D. and Waxman, J., "A Study of Three Database Query Language," Proc. of the International Conf. on Databases: Improving Usability and Responsiveness, Haifa, Israel, Aug. 1978, pp.77-97

[GRIFN78]

Griffeth, N. D., "Nonprocedural Query Processing for Databases with Access Paths," Proc. of the ACM SIGMOD International Conf. on Management of Data, Austin, Texas, May 1978, pp.160-168

[GRUBD77]

Grubb, D. S. and Cotton, I. W., "Criteria for Evaluation of Data Communications Services," Computer Networks, Vol. 1, 1977, pp.325-340

[GUIDE75]

Guide, "Data Dictionary/Directory Requirements," Data Dictionary/ Directory Project Report, May 1975

[GUPTU79]

Gupta, U., "Bounds on Storage for Consecutive Retrieval," JACM, Vol. 26, No. 1, Jan. 1979, pp.28-36

[HABEA69]

Haberman, A. N., "Prevention of System Deadlocks," CACM, Vol. 12, No. 7, July 1969, pp.373-377

[HAERT78]

Haerder, T., "Implementing a Generalized Access Path Structure for a Relational Database System," ACM TODS, Vol. 3, No. 3, Sept. 1978, pp.285-298

[HAINJ74]

Hainant, J. L. and Lecharlier, R., "An Extensible Semantic Model for Data Base," Proc. IFIP Congress 1974, North-Holland, Amsterdam, 1974, pp.1022-1025

[HAINJ77]

Hainaut, J. L., "Some Tools for Data Independence in Multi-level Data Base Systems," Proc. IFIP Working Conf. on Architecture and Models in DBMS, Jan. 1977, pp.187-211

[HALLP76]

Hall, P., Owlett, J. and Todd, S., "Relations and Entities," in [NIIS76A] Jan. 1976, pp.201-220

[HAMAE78]

Hamanaka, E., Takizawa, M., and Ito, T., "Resource Integration System Using JIPNET," (Japanese) Proc. of the 19th Annual Convention of Information Processing Soc. Japan, Aug. 1978

[HHAMMM75]

Hammer, M. M. and McLeod, D. J., "Semantic Integrity in a Relational Data Base System," Proc. International Conf. on VLDB, Framingham, MA., Sept. 1975, pp.25-47

[HAMMM76A]

Hammer, M., "Data Abstractions for Data Bases," Proc. Conf. on Data: Abstraction, Definition and Structure, Salt Lake City, Utah, March 1976, pp.58-59

[HAMMM76B]

Hammer, M., "Index Selection in a Self-Adaptive Data Base Management System," Proc. of ACM SIGMOD International Conf. on the Management of Data, Washington, DC., June 1976, pp.1-8

[HAMMM76C]

Hammer, M. and Chan, A., "Acquisition and Utilization of Access Patterns in Relational Data Base Implementation," Pattern Recognition and Artificial Intelligence (Chen, C. H. Eds.), Academic Press, 1976, pp.292-313

[HAMMM78]

Hammer, M. and McLeod, D., "The Semantic Model: A Modelling Mechanism for Data Base Applications," Proc. of the ACM SIGMOD International Conf. on Management of Data, Austin, Texas, May 1978, pp.26-36

[HAMMM79]

Hammer, M. and Niamir, B., "A Heuristic Approach to Attribute Partitioning," Proc. of the ACM SIGMOD International Conf. on Management of Data, Boston, MA., May 1979, pp.93-101

[HARDW79]

Hardgrave, W. T., "Ambiguity in Processing Boolean Queries on TDMS Tree Structures: A Study of Four Different Philosophies," Proc. of the 5th VLDB, Rio De Janeiro, Brazil, Oct. 1979, pp.373-397

[HARTH76]

Hartson, H. R., "A Semantic Model for Data Base Protection Languages," Proc. 2nd International Conf. on VLDB, Sept. 1976, pp.27-42

[HAWTP79]

Hawthorn, P. and Stonebraker, M., "Performance Analysis of a Relational Data Base Management System," Proc. of the ACM SIGMOD International Conf. on Management of Data, Boston, MA., May 1979, pp.1-12

[HAYEP77]

Hayes, P. J., "On Semantics Nets, Frames and Associations," Proc. 5th International Joint Conf. on Artificial Intelligence, MIT, Cambridge, Aug. 1977, pp.99-107

[HEBAP78]

Hebalkar, P. G., "Application Specification for Distributed Data Base Systems," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.442-449

[HEGNS78]

Hegner, S. J. and Maulucci, R. A., "Set - Theoretic Foundations of Data - Structure Representation," Information Systems, Vol. 3, No. 3, 1978, pp.193-201

[HEIDG76]

Heidorn, G. E., "Automatic Programming Through Natural Language Dialogue: A Survey," IBM Journal of Research and Development, Vol. 20, No. 4, July 1976, pp.302-313

[HEIDG78]

Heidorn, G. E., "Natural Language Dialogue for Managing an On-Line Calendar," Proc. of the ACM 1978 Annual Conf. (ACM78), Washington, DC., Dec. 1978, pp.45-52

[HELBH77]

Helbig, H., "A New Method for Deductive Answer Finding in a Question - Answering System," Information Processing 77, Aug. 1977, pp.389-393

[HELDG75]

Held, G. D., Stonebraker, M. R. and Wong, E., "INGRES - A Relational Data Base System," AFIPS Conf. Proc., May 1976, pp.409-416

[HELDG78]

Held, G. and Stonebraker, M., "B - Trees Re-Examined," CACM, Vol. 21, No. 2, Feb. 1978, pp.139-143

[HENDG77A]

Hendrix, G. G., "LIFFER: A Natural Language Interface Facility," Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, Berkeley, May 1977, pp.196-201

[HENDG77B]

Hendrix, G. G., Sacerdoti, E. D., Sagalowicz, D. and Slocum, J., "Developing a Natural Language Interface to Complex Data," Proc. 3rd International Conf. on VLDB, Vol. 2, Tokyo, Oct. 1977, pp.37-58 (also Available in ACM TODS, Vol. 3, No. 2, June 1978, pp.105-147)

[HENDG77C]

Hendrix, G. G., "The LIFFER Manual: A Guide to Building Practical Natural Language Interfaces," Artificial Intelligence Center Technical Note 138, SRI International, Menlo Park, Calif., Feb. 1977, p.68

[HENDG77D]

Hendrix, G. G., "Human Engineering for Applied Natural Language Processing," Proc. 5th International Joint Conf. on Artificial Intelligence, MIT, Cambridge, Aug. 1977, pp.183-195

[HENDG78]

Hendrix, G. and Sacerdoti, E. D., "On Natural Language Question Answering System," CACM, Vol. 21, No. 12, DEE, 1978, pp.1085

[HEVNA78A]

Hevner, A. R. and Yao, S. B., "Query Processing on a Distributed Database," Proc. 3rd Berkeley Workshop on Distributed Data Management and Computer Networks, San Francisco, Calif., Aug. 1978, pp.91-107

[HEVNA78B]

Hevner, A. R. and Yao, S. B., "Optimization of Data Access in Distributed Systems," TR281, Computer Science Dept., Purdue Univ., July 1978

[HEVNA78C]

Hevner, A. R. and Yao, S. B., "Query Processing in a Distributed System," to appear in IEEE Trans. on Software Engineering, 1978

[HILLE78]

Hill, E., Jr., "A Comparative Study of Very Large Data Bases," Lecture Notes in Computer Science, 59, Springer - Verlag, 1978, p.140

[HOLLL79A]

Hollaar, L. A., "Text Retrieval Computers," IEEE Computer, Vol. 21, No. 3, March 1979, pp.40-50

[HOLLL79B]

Hollaar, L. A., "A Design for a List Merging Network," IEEE Trans. on Computers, Vol. C-28, No. 6, June 1979, pp.406-413

[HOLTA70]

Holt, A. F., Commonen, C., "Events and Conditions," Record of the Project MAC Conference on Networks, 1970, pp.3-52

[HOTAR77B]

Hotaka, R. and Tsubaki, M., "Self-Descriptive Relational Data Base," Proc. on 3rd Very Large Data Base, Tokyo, 1977, pp.415-426

[HOUSB77]

Housel, B. C., "Unified Approach to Program and Data Conversion," Proc. 3rd International Conf. on VLDB, Tokyo, Oct. 1977, pp.327-335

[HOUSB79]

Housel, B. C., Waddle, V. and Yao, S. B., "The Functional Dependency Model for Logical Database Design," Proc. of the 5th International Conf. on VLDB, Rio De Janeiro, Brazil, Oct. 1979, pp.194-208

[HSIAD79]

Hsiao, D. K., "Data Base Machines are Coming, Data Base Machines are Coming," IEEE Computer, Vol. 12, No. 3, March 1979, pp.7-9

[HSIAD79B]

Hsiao, D. K., Kerr, D. S. and Nee, C. J., "Database Access Control in the Presense of Context Dependent Protection Requirements," IEEE Trans. on Software Engineering, Vol. SE-S, No. 4, July 1979, pp.349-358

[HULTC77]

Hulten, C. and Soderlund, L., "A Simulation Model for Performance Analysis of Large Shared Data Bases," Proc. 3rd International Conf. on VLDB, Tokyo, Japan, Oct. 1977, pp.128-143

[HUMPS74]

Humpherey, S. M., "Searching the Medlars Citation File On-Line Using Elhill 2 and Stairs: A Comparison," Information Storage and Retrieval, Vol. 10, 1974, pp.321-329

[HUNTH79]

Hunt, H. B. and Rosenkrantz, D. J., "The Complexity of Testing Predicate Locks," Proc. of the ACM SIGMOD International Conf. on Management of Data, Boston, MA., May 1979, pp.127-133

[HWANK77]

Hwang, K. and Yao, S. B., "Optimal Batched Searching of Tree Structured Files in Multiprocessor Computer Systems," JACM, Vol. 24, No. 3, July 1977, pp.44-454

[ISO 77]

"Basic Architecture for End to End Protocols (French Contribution on End to End Protocols - Project 17)," ISO/TC97/SC6, Data Communications, Jan. 1977, p.8

[ISOT 78]

ISO/TC97/SC16, "Comments on Provisional Reference Model of Open Systems Architecture (Revision 1)," INWG General Note 175, Oct. 1978, p.27

[JEFFD76]

Jefferson, D. K., "The Role of the External Schema," Proc. 2nd Share Conf. on DBMS, Montreal, Canada, April 1976, pp.67-79

[KALIL78]

Kalinichenko, L. A., "Data Models Transformation Method Based on Axiomatic Data Model Extension," Proc. of the 4th VLDB, Berlin, Sept. 1978, pp.549-555

[KAMBY77]

Kambayashi, Y., Tanaka, K. and Yajima, S., "A Relational Data Language with Simplified Binary Relation Handling Capacity," Proc. 3rd International Conf. on VLDB, Tokyo, Oct. 1977, pp.338-350

[KAMEI78]

Kameny, I., Weiner, J., Crilley, M., Bruger, J., Gates, R. and Brill, D., "EUFID: The End User Friendly Interface to Data Management Systems," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.380-391

[KAPLS78]

Kaplan, S. J., "On the Difference Between Natural Language and High Level Query Language," Proc. of the 1978 Annual Conf. (ACM78), Washington, DC., Dec. 1978, pp.27-38

[KATZR79]

Katz, R. H., "Performance Enhancement for Relational Systems Through Query Compilation," AFIPS Conf. Proc., Vol. 48, New York, June 1979, pp.741-747

[KEGUA77]

Kegus, A. E., "EURONET Guideline: Standard Commands for Retrieval Systems," Final Report on a Study Carried Out for the Commission of the European Communities, DG XIII, Dec. 1977, p.66

[KELLC77]

Kellogg, C., Klahr, P. and Travis, L., "Deductive Methods for Large Data Bases," Proc. 5th International Joint Conf. on Artificial Intelligence, MIT, Cambridge, Aug. 1977, pp.203-209

[KENTW73]

Kent, W., "A Primer of Normal Forms," IBM Technical Report TR02. 600, IBM Res. Lab., San Jose, Dec. 1973

[KENTW76]

Kent, W., "New Criteria for the Conceptual Model," Proc. 2nd International Conf. on VLDB, Brussels, Belgium, Sept. 1976, pp.1-12

[KENTW78]

Kent, W., "Data and Reality," North-Holland, 1978

[KENTW79]

Kent, W., "Limitations of Record - Based Information Models," ACM TODS, Vol. 4, No. 1, March 1979, pp.107-131

[KERRD79]

Kerr, D. S., "Data Base Machines with Large Content - Addressable Blocks and Structural Information Processors," IEEE Computer, Vol, 21, No. 3, March 1979, pp.64-79

[KERSL76A]

Kershberg, L., Klug, A. and Tsichritzis, P., "A Taxonomy of Data Models," Computer Systems Research Group, Univ. of Toronto, Technical Report CSRG-70, May 1976 (also Available from IEEE Computer Society, R-77-47)

[KERSL76B]

Kershberg, L., Klug, A. and Tsichritzis, D., "A Taxonomy of Data Models," Proc. 2nd International Conf. on VLDB, Brussels, Belgium, Sept. 1976, pp.43-64

[KERSL76C]

Kershberg. L., Ozkarahan, E. A. and Pacheco, J. E. S., "A Synthetic English Query Language for a Relational Associative Processor" Proc. 2nd International Conf. on Software Engineering, San Francisco, Calif., Oct. 1976, pp.505-519

[KERSL76D]

Kershberg, L. and Pacheco, J. E. S., "A Functional Data Base Model," Computer Science Monograph, Pontifica Universidade Catiolica do Rio De Janeiro, Feb. 1976 (also Available as Technical Report 13, Dept. of Information Systems Management, Univ. of Maryland, 1976)

[KILGF75]

Frederick, G. Kilgour, "Computerized Library Networks," Second-UJCC, 1975, pp.166-171

[KIMBS75]

Kimbleton, S.R. and Schneider, G. M., "Computer Communication Networks," ACM Computing Surveys, Vol. 7, No. 3, Sept. 1975

[KIMBS78A]

Kimbleton, S. R., "Data Sharing Protocols: Structure, Requirement and Implementation," Proc. of IEEE COMPSAC78, Chicago, Ill., Nov. 1978, pp.270-276

[KIMBS78B]

Kimbleton, S. R., Wood, H. M. and Fitzgerald, M. L., "Network Operating System - An Implementation Approach," AFIPS Conf. Proc., Vol. 47, Anaheim, CA., June 1978, pp.773-782

[KIMBS79]

Kimbleton, S. R., Wang, P.S.C. and Fong, E. N., "XNDM: An Experimental Network Data Manager," NBS 1979

[KIMW 79]

Kim, W., "Relational Database Systems," ACM Computing Surveys, Vol. 11, No. 3, Sept. 1979, pp.185-211

[KINGP73]

King, P. F. and Collmeyer, A. J., "Database Sharing - An Efficient Mechanism for Supporting Concurrent Processes," AFIPS Conf. Proc., May 1973, pp.271-275

[KIRBJ77]

Kirby, J. and Kashyap, R. L., "An Approach for Communicating with Data Bases Using English Queries," Proc. of COMPSAC, Chicago, Nov. 1977, pp.650-656

[KIRSF76]

Kirshenbaum, F., "Panel Discussion on ANSI/X3/SPARC DBMS Study Group Report," The ANSI/SPARC DBMS Model, Proc. on 2nd Share Working Conf. on DBMS, 1976, pp.23-34

[KLEEW78]

Kleefstra, W. J., "Data Base Description with a Single Name Category Data Model," Proc. 4th International Conf. on VLDB, Berlin, West-Germany, Sept. 1978, pp.177-185

[KLUGA77]

Klug, A. and Tsichritzis, D., "Multiple View Support Within the ANSI/SPARC Framework," Proc. 3rd International Conf. on VLDB, Tokyo, Oct. 1977, pp.477-487

[KLUGW 78]

Kluge, W. E., "Data File Management in Shift-Register Memories," ACM TODS, Vol. 3, No. 2, June 1978, pp.159-177

[KNUTD73]

Knuth, D. E., "Sorting and Searching," The Art of Computer Programming, Addison-Wesley, Vol. 3, 1973

[KNUTD75]

Knuth, D. E., "Fundamental Algorithms," (2nd Edition), The Art of Computer Programming, Addison-Wesley, Vol. 1, 1975

[KOWAR74]

Kowalski, R. A., "Predicate Logic as Programming Languages," Information Processing 74, 1974, pp.569-574

[KOWAR75]

Kowalski, R., "A Proof Procedure Using Connection Graphs," JACM, Vol. 22, No. 4, Oct. 1975, pp.572-595

[KOWAR78]

Kowalski, R., "Logic and Data Description," Logic and Data Bases (Gallaire, E. H. and Minker, J. Eds.), Plenum Press, 1978, pp.77-103

[KOWAR79]

Kowalski, R., "Algorithm = Logic + Control," CACM, Vol. 22, No. 7, July 1979, pp.424-436

[KROND77]

Kroenke, D., "Database Processing," Scientific Research Associates, Inc. (SRA), 1977, p.408

[KUNGH79]

Kung, H. T. and Papadimitriou, C. H., "An Optimality Theory of Concurrency Control for Database," Proc. of the International Conf. on Management of Data, Boston, MA., May 1979, pp.116-126

[KUNIT77]

Kunii, T. L. and Kunii, H. S., "Design Criteria for Distributed Database Systems," Proc. 3rd International Conf. on VLDB, Tokyo, Oct. 1977, pp.93-104

[LACRM77A]

Lacroix, M. and Pirotte, A., "ILL: An English Structured Query Language for Relational Data Bases," Proc. IFIP TC-2 Working Conf. on Architecture and Models in Data Base Management Systems, Nice, France, Jan. 1977, pp.237-260

[LACRM77B]

Lacroix, M. and Pirotte, A., "Domain - Oriented Relational Language," Proc. 3rd International Conf. on VLDB, Tokyo, Oct. 1977, pp.370-378

[LAFUG79]

Lafue, G.M.E., "An Approach to Automatic Maintenance of Semantic Integrity in Large Design Data Bases," AFIPS Conf. Proc., Vol. 48, New York, June 1979, pp.713-732

[LAINH79]

Laine, H., Maanavilia, O. and Peltola, E., "Grammatical Data Base Model," Information Systems, Vol. 4, No. 4, 1979, pp.257-267

[LAMPL78]

Lamport, L., "Time, Clocks, and the Odering of Events in a Distributed System," CACM, Vol. 21, No. 7, July 1978, pp.558-565

[LANGB78]

Langefo, B., "Comments on a Paper by Biller and Neuhold," Information Systems, Vol. 3, 1978, pp.31-34

[LANGG79]

Langdon, G.G., Jr., "Database Machines: An Introduction," IEEE Trans. on Computers, Vol. C-28, No. 6, June 1979, pp.381-383

[LEER 78]

Lee, R. M. and Gerritsen, R., "Extended Semantics for Organization Hierarchies," Proc. of the ACM SIGMOD International Conf. on Management of Data, Austin, Texas, May-June 1978, pp.18-25

[LEFKH79]

Kefkouits, H. C., et al., "A Status Report on the Activities of the CODASYL End User Facilities Committee (EUFC)," ACM SIGMOD Record, Vol. 10, No. 2&3, Aug. 1979

[LEHMH78]

Lehmann, H., "Interpretation of Natural Language in an Information System," IBM J. Res. Develop., No. 22, No. 5, Sept. 1978, pp.560-572

[LEILH78]

Leilich, H. O., Stiege, G. and Zeidier, H. C., "A Search Processor for Data Base Management Systems," Proc. the 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.280-287

[LELAG78A]

Le Lann, G., "Algorithms for Distributed Data - Sharing Systems which Use Tickets," Proc. the 3rd Berkeley Workshop on Distributed Data Management and Computer Networks, San Francisco, Calif., Aug. 1978,pp.259-272

[LELAG78B]

Le Lann, G., "PSEUDO - Dynamic Resource Allocation in Distributed Databases," Proc. ICCC78, Kyoto, Sept. 1978, pp.245-251

[LEVEW76]

Leveit, W. J. M., "Formal Grammars and the Natural Language User: A Review," Topics in Artificial Intelligence, Springer-Verley, 1976

[LEVIK75A]

Levin, K. D. and Morgan, H.L., "Dynamic File Assignment in Computer Networks Under Varying Access Request Patterns," Decision Sciences Department, Univ. of Pennsylvania, Philadelphia, April 1975, p.21

[LEVIK75B]

Levin, K. D. and Morgan, H. L., "Optimizing Distributed Data Bases - A Framework for Research," AFIPS Conf. Proceedings, May 1975, pp.473-478

[LEWIE77]

Lewis, E. A., Sekino, L. C. and Ting, P.D., "A Canonical Representation for the Relational Schema and Logical Data Independence," Proc. of the 1st COMPSAC, Chicago, Ill., Nov. 1977, pp.276-280

[LICHL78]

Lichten, L., Fernandez, E. B., "Interaction with Databases Through Procedural Languages," Proc. of the 1978 Annual Conf. (ACM78), Washington, DC., Dec. 1978, pp.937-945

[LIENT78B]

Lien, T. E., "Consistency, Concurrency and Crash Recovery," Proc. of the ACM SIGMOD Austin, Texas, May 1978, pp.9-14

[LIENY77]

Lien, Y. E., "Design and Implementation of a Relational Database on a Minicomputer," Proc. ACM Annual Conf., Seattle, Oct. 1977, pp.16-22

[LIENY78]

Lien, Y. E. and Ying, J. H., "Design of a Distributed Entity - Relationship Database System," Proc. of IEEE COMPSAC78, Chicago, Ill., Nov. 1978, pp.277-282

[LIENY79]

Lien, Y. E., "Multivalued Dependencies with Null Values in Relational Data Bases," 5th VLDB, Rio De Janeiro, Oct. 1979, pp.61-66

[LINC 76]

Lin, C. S., Smith, D. C. P. and Smith, J. M., "The Design of a Rotating Associative Memory for Relational Database Applications," ACM TODS, Vol. 1, No. 1, March 1976, pp.53-65

[LINW 79]

Lin, W. C., Lee, R. C. T. and Du, H. C., "Common Properties of Some Multiattribute File System," IEEE Trans. on Software Engineering, Vol. SE-5, No. 2, March 1979, pp.160-174

[LIOUJ77]

Liou, J. H. and Yao, S. B., "Multi-Dimensional Clustering for Data Base Organization," Information Systems, Vol. 2, No. 4, 1977, pp.187-198

[LIPOG78]

Lipovski, G. J., "Architectural Features of CASSM: A Context Addressed Segment Sequential Memory," Proc. of the ACM Sigarch, April 1978, pp.31-38

[LIS 76]

Lockheed Information Systems, "DIACOG Information Retrieval Service, Specializing in Computer Searching of Data Bases in Acience, Technology/Engineering, Social Sciences and Business/ Economics," May, 1976

[LIUJ 76]

Liu, J.W.S., "Algorithms for Parsing Search Queries in Systems with Inverted File Organization," ACM TODS, Vol. 1, No. 4, Dec. 1976, pp.299-316

[LIUZR78]

Liuzzi, R. A. and Berra, P. B., "A Data Base Machine Design and Evaluation Facility," Proc. of COMPSAC, Chicago, Ill., Nov. 1978, pp.716-721

[LOCKP77]

Lockemann, P. C. and Neuhold, E. J., (eds.) "Systems for Large Data Bases," (Proc. the 2nd International Conf. on VLDB, Brussels, Sept. 1976), North-Holland, Amsterdam, 1977

[LOCKP79]

Lockemann, P. C., Mayr, H. C., Weil, W. H. and Wohilleber, W. H., "Data Abstructions for Database Systems," ACM TODS, Vol. 4, No. 1, March 1979, pp.60-75

[LORIR79]

Lorie, R. A. and Nilson, J. F., "An Access Specification Language for a Relational Data Base System," IBM J. Res. Develop., Vol. 23, No. 3, May 1979, pp.286-298

[LOZIE79]

Lozinskii, E. L., "On Query - Answering in Relational Data Bases," AFIPS Conf. Proc., Vol. 48, New York, June 1979, pp.717-720

[LUCCF78]

Luccio, F. and Pagli, L., "Power Tree," CACM, Vol. 21, No. 11, Nov.1978, pp.441-947

[MADNS75]

Madnick, S. E., "Infoplex - Hierarchical Decomposition of a Large Information Management System Using a Microprocessor Complex," NCC, Vol. 44, 1975, pp.581-586

[MADNS79]

Madnick, S. E., "The Infoplex Database Computer: Concepts and Definitions," IEEE COMPCON 79 Spring, S.F., Feb. 1979

[MAHMS76]

Mahmoud, S. and Riordon, J. S., "Optimal Allocation of Resources in Distributed Information Networks," ACM TODS, Vol. 1, No. 1, March 1976, pp.66-78

[MAHMS78]

Mahmoud, S. A., Riordon, J. S. and Toth, K. C., "Design of a Distributed Data Base File Manager for a Mini-computer Network," IEEE COMPSAC 77, Chicao, Ill., Nov. 1977, pp.822-828

[MAHMS79]

Mahmoud, S. A., Riordon, J. S. and Toth, K. C., "Distributed Database Partitioning and Query Processing," IFIP TC-2 Working Conf. on Data Base Architecture, Venice, May 1979, pp.35-59

[MAIED79]

Maier, D. Mendelzon, A. O. and Sagiv, Y., "Testing Implications of Data Dependencies," ACM TODS, Vol. 4, No. 4, Dec. 1979, pp.435-454

[MANNE77]

Manning, E. G. and Peebles, R. W., "A Homogeneous Network for Data - Sharing Communications," Computer Networks, Vol, 1, 1977, pp.211-229

[MANOF78]

Manola, F., "A Review of the 1978 CODASYL Database Specifications," Proc. the 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.232-242

[MARCR69]

Marcus, R. S., Kugel, P. and Kusik, R. L., "An Experimental Computer - Stored, Augumented Catalog of Professional Literature," AFIPS Conf. Proceedings, Vol. 34, pp.461-473

[MARCR75]

Marcus, R. S., "Network Access for the Information Retrieval Application," IEEE Intercon Conf. Record, Apr. 1975, pp.(25/4), 1-7

[MARCR76]

Marcus, R. S. and Reintjes, J. F., "The Networking of Interactive Bibliographic Retrieval Systems," Massachusetts Institute of Technology, Electronic Systems Lab., Report ESL-R-656, March 1976, NTIS Order No. PB 252 407, pp.164

[MARCR77]

Marcus, R. S. and Reintjes, J. F., "Computer Interfaces for User Access to Heterogeneous Information - Retrieval Systems," Massachusetts Laboratory, Report ESL-R-739, April 1977, p.84

[MARIT75]

Maril, T. and Stern, D., "The Datacomputer - A Network Data Utility," AFIPS Conf., May 1975, pp.389-395

[MARTW78]

Martin, W. A., "Some Comments on EQS, A Near Term Natural Language Data Base Query System," Proc. of the 1978 Annual Conf. (ACM78), Washington, D. C., Dec. 1978, pp.156-172

[MARYF78]

Maryanski, F. J., "A Survey of Developments in Distributed Data Base Management Systems," IEEE Computer, Vol. 11, No. 2, Feb. 1978, pp.28-38

[MARYF79]

Maryanski, F. J., Norsworthy, K. E., Norsworthy, K. A., Ratliff, J. R., "A System Architecture for Distributed Data Base Management," IEEE COMPCON 79 Spring, S.F., Feb. 1979, pp.163-167

[MASUY79]

Masunaga, Y., "On a Semantic Aspect of View Updates in a Relational Database System," Research Institute of Electrical Communication, Tohoku University, Sendai, Japan, 1979

[MCFAM75]

McFarland, M. E., "The National Referral Center," Special Libraries, Mar. 1975, pp.126-132

[MCGEW76]

McGee, W. C., "On User Criteria for Data Model Evaluation," ACM TODS, Vol. 1, No. 4, Dec. 1976, pp.370-387

[MCLED77]

McLeod, D. J., "High Level Definition of Abstract Domains in a Relational Data Base System," Computer Languages, (Pergamon Press), Vol. 2, 1977, pp.61-73

[[MCQUJ77]

McQuillan, J. M. and Walden, D. C., "The ARPA Network Design Decisions," Computer Networks, Vol. 1, No. 5, Aug. 1977, pp.243-289

[MELDM78]

Meldman, M. McLeod, D. J. and Squire, M., "RISS: A Relational Data Base Management System for Minicomputers," Van Nostrand Reinhold Company, 1978, pp.113

[MENDA79]

Mendelzon, A. O., "Generalized Mutual Dependencies and the Decomposition of Data Base Relations," Proc. of the 5th International Conf. on VLDB, Rio De Janeiro, Brazil, Oct. 1979, pp.75-82

[MENDA79A]

Mendelzon, A. O., "On Axiomatizing Multivalued Dependencies in Relational Databases," JACM,Vol. 26, No. 1, Japan, 1979, pp.37-44

[MENDA79B]

Mendelzon, A. O., "Generalized Mutual Dependencies and the Decomposition of Data Base Relations," Proc. of the 5th International Conf. on VLDB, Rio De Janeiro, Brazil, Oct. 1979, pp.75-82

[MERLP74]

Merlin, P. M., "A Study of Recoverability of Computing Systems," Ph. D. Dissertation, Dept. of Information and Computer Science, University of Calif., Irvine, 1974, p.165

[MERLP76]

Merlin, P. M. and Farber, D. J., "Recoverability of Communication Protocols - Implications of a Theoretical Study," IEEE Transactions on Communications, Sept. 1976, pp.1036-1043

[MERRT78]

Merrett, T. H., "The Extended Relational Algebra, A Basis for Query Languages," Proc. of the International Conf. on Databases: Improving Usability and Responsiveness, Haifa, Israel, Aug. 1978, pp.99-128

[MICHA76]

Michaels, A. S., Mittman, B. and Carlson, C. R., "A Comparison of the Relational and CODASYL Approaches to Data Base Management," ACM Computing Surveys, Vol. 8, No. 1, March 1976, pp.125, 151

[MILLL75]

Miller, L. A., "Naive Programmer Problems with Specification of Transfer-of-Control," Proc. NCC75, Vol. 44, 1975, pp.653-657

[MINKJ75]

Minker, J., "Performing Inferences Over Relational Data Bases," Proc. of the ACM SIGMOD International Conf. on the Management of Data, San Jose, CA., May 1975, pp.79-91

[MINKJ78]

Minker, J., "Binary Relations: Matrices and Inference Developments," Information Systems, Vol. 3, No. 1, 1978, pp.37-47

[MINOT78]

Minoura, T., "Maximally Concurrent Transaction Processing," Proc. the 3rd Berkeley Workshop on Distributed Data Management and Computer Networks, San Francisco, Aug. 1978, pp.206-214

[MINSN76]

Minsky, N., "Files with Semantics," Proc. ACM SIGMOD International Conf. on Management of Data, Washington, D.C., June 1976, pp.65-73

[MOHAC79A]

Mohan, C., "An Analysis of the Design of SDD-1: A System for Distributed Data Bases," Technical Report SDBEG-11, GAL-E-075, April 1975

[MOHAC79B]

Mohan, C. and Yeh, R. T., "Distributed Data Base Systems - A Framework for Data Base Design," Technical Report SDBEG-10, GAL-E-074

[MOHAC79C]

Mohan, C., "Data Base Design in the Distributed Environment," Working Paper WP-7902, GAL-E-076, May 1979

[MOHAC79D]

Mohan, C., "Distributed Data Base Management Progress, Problems, Some Proposals and Future Directions," WP-7802 (GAL-E-078) Software and Data Base Engineering Group, Dept. of Computer Sciences, Univ. of Texas at Austin, Austin, Texas 78712, May 1979

[MORGH77]

Morgan, H. L. and Levin, K. D., "Optimal Program and Data Locations in Computer Networks," CACM, Vol. 20, No. 5, May, 1977, pp.315-322

[MORRP77]

Morris, P. and Sagalowicz, D., "Managing Network Access to a Distributed Database," Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Network, May 1977, pp.58-67

[MOULP76]

Moulin, P., Randon, J., Teboul, M., et al., "Conceptual Model as a Data Base Design Tool," Proc. IFIP TC-2 Working Conf. on Modelling in DBMS, Jan. 1976, pp.221-238

[MUKHA79A]

Mukhopadhyay, A. and Hurson, A., "An Associative Search Language for Data Management," AFIPS Conf. Proc., Vol. 48, New York, pp.727-732

[MUKHA79B]

Mukhopadhyay, A., "Hardware Algorithms for Nonnumeric Computation," IEEE Trans. on Computers, Vol. C-28, No. 6, June 1979, pp.384-394

[MUNZR78A]

Munz, R., "System Architectures for Managing Distributed Databases," GI-Fachtagung "Datenbanken in Rechnernetzen mit Kleinrechhern," Karlsruhe, April 1978

[MUNZR78B]

Munz, R., "Realization, Synchronization and Restart of Update Transactions in a Distributed Database System," VDN-Report 8/78, Technische Universitat, Berlink Aug. 1978

[MUNZR79]

Munz, R., "Gross Architecture of the Distributed Database System VDN," Proc. of the IFIP Working Conf. on Database Architecture, Venice, May 1979, pp.23-34

[MYLOJ76]

Mylopoulos, J., Borgida, A., Cohen, P., Roussopoulou, N., Tsotsos, J. and Wong, H., "Torus: A Step Towards Bridging the Gap Between Data Bases and the Casual User," Information Systems, Vol. 2, No. 2, 1976, pp.49-64

[NAHOE76]

Nahouraii, E., Brooks, L. O. and Cardenas, A. F., "An Approach to Data Communication Between Different Generalized Data Base Management Systems," Proc. 2nd International Conf. on VLDB, Sept. 1976, pp.117-142

[NATIJ78]

Nations, J. and Su,S. Y. W. "Some DML Instruction Sequences for Application Program Analysis and Conversion," Proc. of the ACM SIGMOD International Conf. on Management of Data, Austin, Texas, May-June 1978, pp.120-131

[NAVAS76]

Navathe, S. B., "Restructuring for Large Database: Three Levels of Abstraction," ACM TODS, Vol. 1, No. 2, June 1976, pp.138-158

[NAVAS78]

Navathe, S. B. and Schkolnick, M., "View Representation in Logical Database Design," ACM SIGMOD International Conf. on Management of Data, Austin, May 1978, pp.144-156

[NAVAS79]

Navathe, S. B. and Leme, J., "On the Implementation of a Conceptual Schema Model within a Three-Level DBMS Architecture," AFIPS Conf. Proc., Vol. 48, New York, June 1979, pp.697-708

[NENAD78]

Nenasce, D. A. and Muntz, R. R., "Locking and Deadlock Detection in Distributed Database," Proc. the 3rd Berkeley Workshop on Distributed Data Management and Computer Networks, San Francisco, Aug. 1978, pp.215-232

[NEUHE77A]

Neuhold, E. J. and Biller, H., "POREL: A Distributed Data Base on an Inhomogoneous Computer Network," Proc. 3rd International Conf. on VLDB, Tokyo, Japan, Oct. 1977, pp.380-395

[NEUHE77B]

Neuhold, E. J. and Biller, H., "Distributed Data Bases on a Network of Minicomputer," Bases de Donnees Reparties, Institut de Programmation, Univ. of Paris VI, April 1977, pp.110-138

[NEUHE77C]

Neuhold, E. J., "The Design of Distributed Data Bases," Proc. the 2nd Hungarian Computer Science Conference, Budapest, Hungary, June 1977, pp.673-696

[NICOJ78A]

Nicolas, J. M., "Mutual Dependencies and Some Results on Undecomposable Relations," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.360-367

[NICOJ78B]

Nicolas, J. M., "First Order Logic Formalization for Functional, Multivalued and Mutual Dependencies," Proc. of the ACM SIGMOD International Conf. on Management of Data, Austin, Texas, May 1978, pp.40-46

[NICOJ78C]

Nicolas, J. M. and Gallaire, H., "Data Base: Theory Vs. Interpretation," Logic and Data Bases (Gallaire, H. and Minker, J. Eds.), Plenum Press, 1978, pp.33-54

[NICOJ78D]

Nicolas, J. M. and Yazdanian, K., "Integrity Checking in Deductive Data Bases," Logic and Data Bases (Gallaire, E. H. and Minker, J.), Plenum Press, 1978, pp.325-343

[NIEVJ74]

Nievergelt, J., "Binary Search Trees and File Organization," ACM Computing Surveys, Vol. 6, No. 3, Sept. 1974, pp.195-207

[NIH 76]

National Institute of Health "Communication in the Service of American Health ... A Bicentennial Report from the National Lib. of Medicine," Development of Health, Education, and Welfare, Public Health Service, 1976, p.198

[NIJEA78]

Nijenhuis, A. and Wilf, H. S., "Combinatorial Algorithms," (2nd Ed.), Academic Press, New York, 1978

[NIJSG76A]

Nijssen, G. (Eds.), "Modelling in Database Management Systems (Proceedings of the IFIP TC-2 Working Conference in Freudenstadt, Jan. 1976)," North-Holland, 1976

[NIJSG76B]

Nijssen, G., "A Gross Architecture for the Next Generation Database Management System," in [NIJS76A], pp. 1-24

[NIJSG77A]

Nijssen, G. (Eds), "Architecture and Model in Database Management Systems (Proc. of the IFIP TC-2 Working Conf. Jan. 1977)," North-Holland, 1977

[NIJSG77B]

Nijssen, G., "Current Issues in Conceptual Schema Concepts," in [NIJS77A], Jan. 1977, pp.31-66

[NIJSG77C]

Nijssen, G., "A Gross Architecture for the Next Generation DBMS," Information Processing 77 (Proc. IFIP Conf.), 1977, pp.327-335

[NIJSG77D]

Nijssen, G., "The Next Five Years in Data Base Technology," Infotech State of the Art Report, Data Base Technology, Vol. 2, 1978, pp.213-256

[NUTTJ72]

Nutt, G. J., "Evaluation Nets for Computer System Performance Analysis," AFIPS Conference Proceedings FJCC, 1972, pp.279-286

[OLLET78]

Olle, T. W., "The CODASYL Approach to Data Base Management," John Willey & Sons, 1978, p.287

[ORGAE72]

Organik, E. J., "The Multics System: An Examination of Its Structure," MIT Press, 1972

[OSBOS79]

Osborn, S. L., "Towards a Universal Relation Interface," 5th VLDB, Rio De Janeiro, Oct. 1979, pp.52-60

[OSMAI79]

Osman, I. M., "Updating Defined Relations," AFIPS Conf. Proc., Vol. 48, New York, June 1979, pp.733-740

[OTTMT78]

Ottmann, T., Six, H. W. and Wood, D., "Right Brother Trees," CACM, Vol. 21, No. 9, Sept. 1978, pp.769-776

[OWLEJ77]

Owlett, J., "Defferring and Defining in Databases," Proc. IFIP Working Conf. on Architecture and Models in Data Base Management Systems, Jan. 1977, pp.277-291

[OZKAE77]

Ozkarahan, E. A., Schuster, S. A. and Sev K, K, C., "Performance Evaluation of a Relational Associative Processor," ACM TODS, Vol. 2, No. 2, June 1977, pp.175-195

[OZKAE78]

Ozkarahan, E. A. and Oflazer, K., "Microprocessor Based on Modular Database Processors," Proc. of the 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.300-311

[PAIKI79]

Paik, I. and Delobel, C., "A Strategy for Optimizing the Distributed Query Processing," Proc. of the 1st International Conf. on Distributed Computing Systems, Huntsville, Alabama, Oct. 1979, pp.686-698

[PAOLP77]

Paolini, P. and Pelagatti, G., "Formal Definition of Mappings in a Data Base," Proc. ACM SIGMOD International Conf. on Management of Data, Toronto, Canada, Aug. 1977, pp.40-46

[PAOLP78]

Paolini, P., "An Alternative Structure for Data Base Management Systems," Proc. the 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.243-254

[PAPAC79]

Paradimitrious, C. H., "The Serializability of Concurrent Database Updates," JACM, Vol. 26, No. 4, Oct. 1979, pp.631-653

[PARDR78]

Pardó, R., Liu, M. T. and Babic, G., "An N-Process Communication Protocol for Distributed Processing," Proc. of Computer Network Protocols, Liege, Belgium, Feb. 1978, D7-1, D7-10

[PARKD79]

Parker, D. S. and Delobel, C., "Algorithmic Applications for a New Result on Multivalued Dependencies," 5th VLDB, Rio De Janeiro, Oct. 1979, pp.67-74

[PAXTW77]

Paxton, W. H., "A Framework for Speech Understanding," Artificial Intelligence Center, Technical Note 142, SRI, June 1977, p.280

[PEELR78]

Peeldes, R. and Manning, E., "System Architecture for Distributed Data Management," Computer, Jan. 1978, pp.40-47

[PELAG78A]

Pelagatti, G., Paolini, P. and Bracchi, "Mapping External View to a Common Data Model," Information Systems, Vol. 3, No. 2, 1978, pp.141-151

[PELAG78B]

Pelagatti, G. and Schreiber, F. A., "Comparison of Different Access Strategies in a Distributed Databases," Databases: Improving Usability and Responsiveness (Shneiderman, B. Ed.), Academic Press, 1978, pp.399-410

[PELAG79A]

Pelagatti, G. and Schreiber, F. A., "Evaluation of Transmission Requirements in Distributed Database Access," Proc. of the ACM SIGMOD International Conf. on Management of Data, Boston, MA., May 1979, pp.102-108

[PELAG79B]

Pelagatti, G. and Schreiber, F. A., "A Model of an Access Strategy in a Distributed Database," Proc. of the IFIP Working Conf. on Data Base Architecture, Venice, May 1979, pp.55-71

[PETEJ77]

Peterson, J. J., "Petri Nets," ACM Computing Survey, Vol. 9, No. 3, Sept. 1977, pp.223-252

[PETRS76]

Petrick, S. R., "On Natural Language Based Computer Systems," IBM Journal of Research and Development, Vol. 20, No. 4, July 1976, pp.314-325

[PIESD79]

Ries, D. R. and Stonebraker, M. R., "Locking Granularity Revised," ACM TODS, Vol. 4, No. 2, June 1979, pp.210-227

[PIROA77]

Pirotte, A., "The Entity - Association Model: An Information - Oriented Data Base Model," Proc. International Computing Symposium 1977, Liege, Belgium, April 1977, pp.581-597

[PIROA78A]

Pirotte, A., "Linguistic Aspects of High-Level Relational Languages," INFOTECH State of the Art Report, Database Technology:2, 1978, pp.271-300

[PIROA78B]

Pirotte, A., "High Level Data Base Query Language," Logic and Data Bases (Gallaire, H. and Minker, J. Eds.), Plenum Press, 1978, pp.409-436

[PLAGB72]

Plagman, B. K. and Altshuler, G. P., "A Data Dictionary/ Directory System within the Context of an Integrated Corporate Data Base," AFIPS, FJCC, Vol. 40, 1972, pp.1133-1140

[PLATW76]

Plath, W. J., "Request: A Natural Language Question - Answering System," IBM Journal of Research and Development, Vol. 20, No. 4, July 1976, pp.326-335

[POPEG79]

Popek, G. J. and Kline, C. S., "Encryption and Secure Computer Networks," ACM Computing Surveys, Vol. 11, No. 4, Dec. 1979, pp.331-395

[POUZL75]

Pouzin, L., "The CYCLADES Network - Present State and Development Trends," RES505.2, July 1975, pp.11

[POUZL77]

Pouzin, L., "Packet Networks - Issues and Choices," Proc. IFIP Congress 77, Toronto, Aug. 1977, pp.515-521

[PRENC77B]

Prenner, C. J., "A Uniform Notation for Expressing Queries," Memorandom No. UCB/ERL MM77/60, Electronics Research Lab., University of Calif., Berkeley, Calif., Sept. 1977

[PRENC78A]

Prenner, C. J. and Rowe, L. A., "Programming Languages for Relational Data Base Systems," AFIPS Conf. Proc., Vol. 47, May 1978, pp.849-855

[PYKET74A]

Pyke, T. N., Jr., "Network Access Techniques: Some Recent Development," Proc. 3rd Annual Texas Conference, Oct. 1974, pp.2-2-1 - 2-2-3

[PYKET74B]

Pyke, T. N., Jr., "Network Access Techniques: ome Recent Developments," Proc. Third Texas Conference on Computing System, Nov. 7-8, 1974

[RAMAC79]

Ramamoorthy, C. V. and Wah, B. W.,"Data Management in Distributed Data Bases," AFIPS Conf. Proc.,Vol.48, New York, June 1979, pp.667-680

[RAMAC79B]

Ramamoorthy, C. V. and Wah, B. W., "The Placements of Relations on a Distributed Relational Data Base," Proc. of the 1st Distributed Computing System, Huntsville, Oct. 1979, pp.642-650

[RAPPM79]

Rappolt, M. S., Jr., "Distributed Database - The User View," INFOTECH State of the Art Report: Distributed Databases, Vol. 2, 1979, pp.257-278

[RAVEN77]

Raver, N. and Hubbard, G. U., "Automated Logical Data Base Design: Concepts and Applications," IBM Systems Journal, Vol. 16, No. 3, 1977, pp.287-312

[REINJ69]

Reintjes, J. E., "System Characteristics of Intrex," AFIPS Conf. Proceedings, Vol. 34, 1969, pp.457-459

[REISS79]

Reiss, S. P., "Security in Databases: A Combinatorial Study," JACM, Vol. 26, No. 1, Jan. 1979, pp.45-57

[RIESD77]

Ries, D. R. and Stonebraker, M., "Effects of Locking Granularity in a Database Management System," ACM TODS, Vol. 2, No. 3, Sept. 1977, pp.233-246

[RIESD78]

Ries, D. and Epstein, R., "Evaluation of Distributed Criteria for Distributed Data Base Systems," Memorandum No. UCB/ERL M78/22, Electronics Research Lab., UC. Berkeley, May 1978

[RISSJZZ]

Rissan, J., "Independent Components of Relations," ACM TODS, Vol. 2, No. 4, Dec. 1977, pp.317-325

[RITCD74]

Ritchie, D. M. and Thompson, K., "The Unix Time-Sharing System," CACM, Vol. 12, No. 7, July 1974, pp.365-375

[ROB EL70]

Roberts, L. G. and Wessler, B. D., "Computer Network Development to Achieve Resource Sharing," AFIPS Conf. Proc. (FJCC), 1970, pp.543-549

[ROBEL77]

Roberts, L. G., "Packet Network Design - The Third Generation," Information Processing 77, North-Holland, 1977, pp.541-546

[ROSEA68]

Rosenfeld, A., "An Introduction to Algebraic Structures," Holden-Day, San Francisco, Calif., 1968

[ROSED77]

Rosenkrantz, D. J., Stearns, R. E. and Lewis, P. M., "A System Level Concurrency Control for Distributed Database Systems," Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, Berkeley, May 1977, pp.132-145

[ROSED78]

Rosenkrantz, D. J., Stearns, R. E. and Lewis II, P.M., "System Level Concurrency Control for Distributed Database Systems," ACM TODS, Vol. 3, No. 2, June 1978, pp.178-198

[ROSER74]

Rosenthal, R. and Watkins,S.,"Automated Access to Network Resources: A Network Access Machine," Computer-Networks - Trends and Applications, NBS - ICST and IEEE Computer Society, May 1974

[ROSER75]

Rosenthal, R., "Accessing On-Line Network Resources with a Network Access Machine," IEEE Intercon Conf. Record, April 1975, pp.25/3:1-4

[ROSER76A]

Rosenthal, R., "Network Access Techniques - A Review," AFIPS Conference Proceedings, Vol. 45, May 1976, pp.495-499

[ROSER76B]

Rosenthal, R., "A Review of Network Access Techniques with a Case Study: The Network Access Machine," NBS Technical Note 917, NBS, July 1976, p.29

[ROTHJ74]

Rothnie, J. B., "An Approach to Implementing a Relational Data Management System," Proc. ACM SIGMOD Workshop on Data Description, Access and Control, 1974, pp.277-294

[ROTHJ75]

Rothnie, J. B., "Evaluating Inter-Entry Retrieval Expressions in a Relational Data Base Management System," AFIPS Conf. Proceedings, May 1975, pp.417-423

[ROTHJ77A]

Rothnie, J. B., Coodman, N. and Bernstein, P. A., "The Redundant Update Methodology of SDD-1: A System for Distributed Data Bases (The Fully Redundant Case)," CCA Technical Report CCA-77-02, Computer Corporation of America, June 1977, p.70

[ROTHJ77B]

Rothnie, J. B. and Goodman, N., "An Overview of the Preliminary Design of SDD-1: A System for Distributed Databases," Proc. of the 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, UC. Berkeley, California, May 1977, pp.39-57

[ROTHJ77C]

Rothnie, J. B. and Goodman, N., "A Survey of Research and Development in Distributed Database Management," Proc. 3rd International Conf. on VLDB, Tokyo, Oct. 1977, pp.48-62

[ROUSN75]

Roussopoulos, N. and Mylopoulos, J., "Using Semantic Networks for Data Base Management," Proc. International Conf. on VLDB, Oct. 1975, pp.144-172

[ROUSN79]

Roussopoulos, N., "CSDL: A Conceptual Schema Definition Language for the Design of Data Base Application," IEEE Transactions on Software Engineering, Vol. SE-5, No. 5, Sept. 1979, pp.481-496

[ROWEL79]

Rowe, L. A. and Shoens, K. A., "Data Abstraction Views and Updates in RIGEL," Proc. of the ACM SIGMOD International Conf. on Management of Data, Boston, MA., May 1979, pp.71-81

[RUBYJ78]

Ruby, J. B. and Carnick, A. G., "An Approach to Providing a Relational Query Processor for a Limited CODASYL Data Base Management System," 3rd USA-Japan Computer Conference, Oct. 1978, pp.139-144

[SAGAD77]

Sagalowicz, D., "IDA: An Intelligent Data Access Program," Proc. 3rd International Conf. on VLDB, Tokyo, Oct. 1977, pp.293-302

[SAGEN78]

Sager, N., "Natural Language Information Formatting: The Automatic Conversion of Texts to a Structured Data Base," Advances in Computers, Vol. 17, Academic Press, 1978

[SAGIY78]

Sagiv, Y. and Yannakakis, M., "Equivalence Among Relational Expressions with the Union and Difference Operations," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.535-548

[SAKAH78B]

Sakai, H., "On the Optimization of An Entity - Relationship Model," Proc. 3rd USA-Japan Computer Conference, Oct. 1978, pp.145-149

[SCANR74]

Scantlebury, R. A. and Wilkinson, P. T., "The National Physical Laboratory Data Communication Network," Proceedings of the 2nd ICCC, Stockholm, Aug. 1974, p.223

[SCHAR73]

Schank, R. C. and Colby, K. M., "Computer Models of Thought and Languages," W. H. Freeman and Company, San Francisco, 1973

[SCHAR76]

Schantz, R. E. and Millstein, R. E., "The FOREMAN: Proving the Program Execution Environment for National Software Works," BBN Report, No. 3266, March 1976

[SCHEA78]

Scherr, A. L., "Distributed Data Processing," IBM System Journal, Vol. 17, No. 4, 1978, pp.324-343

[SCHEK77]

Shenk, K. L. and Pinkert, J. R., "An Algorithm for Servicing Multi-Relational Queries," Proc. ACM SIGMOD Conf. on Management of Data, Toronto, Canada, Aug. 1977, pp.10-19

[SCHKM77]

Schkolinik, M., "A Clustering Algorithm for Hierarchical Structures," ACM TODS, Vol. 2, No. 1, March 1977, pp.27-44

[SCHKM78]

Schkolnick, M., "A Survey of Practical Database Design Methodology and Techniques," Proc. of the 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.474-487

[SCHMA75A]

Schmid, H. A., "On the Semantics of the Relational Data Model," Proc. International Conf. on Management of Data, ACM SIGMOD, San Jose, May 1975, pp.211-223

[SCHMH75B]

Schmid, H. A. and Bernstein, P. A., "A Multi-Level Architecture for Relational Data Base Systems," Proc. International Conf. on VLDB, Fragingham, MA., Sept. 1975, pp.202-226

[SCHMH77]

Schmid, H. A., "An Analysis of Some Constructs for Conceptual Models," Proc. IFIP TC-2 Working Conf. on Architecture and Model in DBMS, Jan. 1977, pp.119-148

[SCHMJ77]

Schmidt, J. W., "Some High Level Language Constructs for Data of Type Relation," ACM TODS, Vol. 2, No. 3, Sept. 1977, pp.247-261

[SCHMJ78]

Schmidt, J., "Type Concept for Database Definition," Databases: Improving Usability and Responsiveness (B. Schneiderman ed.), Academic Press, 1978, pp.215-244

[SCHUG77]

Schussel, G., "The Role of the Data Dictionary," Datamation, June 1977, pp.129-142

[SCHUS78]

Schuster, S. A., Nguyen, H. B., Ozkarahan, E. A. and Smith, K. C., "RAP.2 - An Associative Processor for Data Bases," Proc. of ACM SIGARCH, April 1978, pp.52-59

[SCHUS79]

Schuster, S. A., Nguyen, H. B., Ozkarahan, E. A. and Smith, K. C., "RAP.2 - An Associative Processor for Databases and Its Applications," IEEE Trans. on Computers, Vol. C-28, No. 6, June 1979, pp.446-458

[SDDT 77]

The Stored - Data Definition and Translation Task Group of the CODASYL Systems Committee, "Stored - Data Description and Data Translation: A Model and Language," Information Systems, Vol. 2, No. 3, 1977, pp.95-148

[SELIP79]

Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A. and Price, T. G., "Access Path Selection in a Relational Database Management System," Proc. of the ACM SIGMOD, Boston, May 1979, pp.23-34

[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

[SENKM75A]

Senko, M. E., "The DDL in the Context of a Multilevel Structured Description: DIAM 2 with the FORAL," Proc. IFIP TC2 Working Conf. on Data Base Description, 1975, pp.239-257

[SENKM75B]

Senko, M. E., "Specification of Stored Data Structures and Desired Output Results in DIAM 2 with FORAL," Proc. International Conf. on VLDB, 1975, pp.557-571

[SENKM76A]

Senko, M. E., "DIAM as a Detailed Example of the ANSI SPARC Architecture," Proc. IFIP Working Conf. on Modelling in DBMS, 1976, pp.73-94

[SENKM76B]

Senko, M. E., "DIAM 2: The Binary Infological Level and Its Data Base Language - FORAL," Proc. Conf. on Data: Abstraction, Definition and Structure, Salt Lake City, Utah, March 1976, pp.121-140

[SENKM76C]

Senko, M. E. and Altman, E. B., "DIAM 2 and Levels of Abstraction, The Physical Device Level: A General Model for Access Methods," Proc. 2nd International Conf. on VLDB, Brussels, Sept. 1976, pp.79-94

[SENKM77A]

Senko, M. E., "Conceptual Schemas, Abstract Data Structures, Enterprise Descriptions," Proc. International Computing Symposium, 1977, pp.85-102

[SENKM77B]

Senko, M. E., "Data Structures and Data Accessing in Data Base Systems: Past, Present, Future," IBM Systems Journal, Vol. 16, No. 3, 1977, pp.208-257

[SENKM77C]

Senko, M., "FORAL LP-Making Pointed Queries with a Light Per," Proc. of the IFIP Congress 77 (Information Processing 77), Toronto, Aug. 1977, pp.635-640

[SENKM78A]

Senko, M. E., "FORAL LP: Design and Implementation," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.255-267

[SENKM78B]

Senko, M. E., "Classification: A Basic Tool for Data Bases and Information Systems," INFOTECH State of the Art Report, Database Technology: 2, 1978, pp.301-315

[SEVED77]

Severance, D. G. and Carlis, J. V., "A Practical Approach to Selecting Record Access Paths," ACM Computing Surveys, Vol. 9, No. 4, Dec. 1977, pp.259-272

[SHAR 76]

Sharman, G. C. H., "A Constructive Definition of Third Normal Form," Proc. ACM SIGMOD International Conf. on Management of Data, Washington, D. C., June 1976, pp.91-99

[SHARG77]

Sharman, G. C. H., "Update-by-Dialogue: Interactive Approach to Database Modification," Toronto, Canada, Aug. 1977, pp.21-29

[SHARG78]

Sharman, G. C. H. and Winterbotton, N., "The Data Dictionary Facilities of NDB," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.186-197

[SHAWA74]

Shaw, A. C., "The Logical Design of Operating Systems," Prentice-Hall, Inc., 1974, p.306

[SHEMJ72]

Shemer, J. E. and Collmeyer, A. J., "Database Sharing: A Study of Interference, Roadblock and Deadlock," Proc. ACM SIGFIDET, May 1972, pp.147-163

[SHIPD79]

Shipman, D., "The Functional Data Model and the Data Language Duplex," Proc. of the ACM SIGMOD International Conf. on Management of Data, Boston, MA., May 1979, pp.59

[SHNEB78]

Shneiderman, B., "Improving the Human Factor Aspect of Database Interactions," ACM TODS, Vol. 3, No. 4, Dec. 1978, pp.417-439

[SHNEB79]

Shneiderman, B., "Human Factors Experiments in Designing Interactive Systems," IEEE Computer, Vol. 12, No. 12, Dec. 1979, pp.9-19

[SHOSA75]

Shoshani, A., "A Logical Level Approach to Data Base Conversion," Proc. SIGMOD Conf., San Jose, California, May 1975

[SHUN 75]

Shu, N. C., Housel, B. C. and Lum, V. Y., "CONVERT: A High Level Translation Definition Language for Data Conversion," CACM, Vol. 18, No. 10, Oct. 1978, pp.557-567

[SHUN 77]

Shu, N. C., Housel, B. C., Taylor, R. W., Ghosh, S. P. and Lum, V. Y., "EXPRESS: A Data Extraction, Processing and Restructuring System," ACM TODS, Vol. 2, No. 2, June 1977, pp.134-174

[SIBLE73]

Sibley, E. H. and Taylor, R. W., "A Data Definition and Mapping Language," CACM, Vol. 16, No. 12, Dec. 1973, pp.750-759

[SIBLE76]

Sibley, E.H., "The Development of Database Technology," ACM Computer Surveys, Vol. 8, No. 1, March 1976, pp.1-5

[SIBLE77]

Sibley, E. H. and Kershberg, L., "Data Architecture and Data Model Considerations," AFIPS Conf. Proc.,Vol. 46, May 1977, pp.85-96

[SIBUM78]

Shibuya, M., Fujisaki, T. and Takao, Y., "Noun-Phrase Model and Natural Query Language," IBM J. Res. Develop., Vol. 22, No. 5, Sept. 1978, pp.541-559

[SIKLL78]

Siklossy, L., "Impertinent Question-Answering Systems: Justification and Theory," Proc. of the 1978 Annual Conf.(CACM 78), Washington, DC, Dec. 1978, pp.39-44

[SIMMR73]

Simmons, R. F., "Semantic Networks: Their Computation and Use for Understanding English Sentences," Computer Models of Thought and Language (Edited by Schank, R. G. and Golby, K. M.), W. F. Freeman and Company, San Francisco, 1973, pp.63-113

[SIMMR77]

Simmons, R. F. and Chester, D., "Inferences in Quantified Semantic Networks," Proc. of the 5th IJCAI, Cambridge, MA., Aug. 1977, pp.267-273

[SMALA79]

Small, D. L. and Chu, W. W., "A Distributed Data Base Architecture for Data Processing in a Dynamic Environment," IEEE COMPCOM 79 Spring, S. F., Feb. 1979, pp.123-127

[SMITD79]

Smith, D. C. P. and Smith, J. M., "Relational Data Base Machines," IEEE Computer, Vol. 12, No. 3, March 1979, pp.28-37

[SMITJ75]

Smith, J. M., "Optimizing the Performance of a Relational Algebra Database Interface," CACM. Vol. 18, No. 10, Oct. 1975, pp.568-576

[SMITJ77A]

Smith, T. J. and Smith, D. C. P., "Database Abstractions: Aggregation," CACM, Vol. 20, No. 6, June 1977, pp.405-413

[SMITJ77B]

Smith, J. M. and Smith, D. C. P., "Database Abstractions: Aggregation and Generation," ACM TODS, Vol. 2, No. 2, June 1977, pp.105-133

[SMITJ78]

Smith, J. M., "A Normal Form for Abstract SYNTAX," Proc. the 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.156-162

[SOFTA77A]

"ADABAS Guide"

[SOFTA77B]

"ADABAS Reference Manual"

[SOFTA77C]

"ADABAS Utilities Manual"

[SOFTA77D]

"ADASCRIPT Manual"

[SOFTA77E]

"ADAMINT Reference Manual"

[SOFTA77F]

"ADAWRITER Reference Manual"

[SOFTA77G]

"Programmer's Introduction to ADABAS"

[SOREP77]

Sorenson, P. G., "Picasso - An Aid to an End-User Facility," Proc. ACM SIGMOD International Conf. on the Management of Data, Toronto, Canada, Aug. 1977, pp.30-39

[SOWAJ76]

Sowa, J. F., "Conceptual Graphs for a Data Base Interface," IBM Journal of Research and Development, Vol. 20, No. 4, July 1976, pp.336-357

[STEET75]

Steel, T. B., "Data Base Standardization - A Status Report," Proc. ACM SIGMOD 1975 Conf.,San Jose, May 1975, pp.65-78

[STEET76]

Steel, T. B., Jr., "Formalization of Conceptual Model," Proc. 2nd Share Working Conf. on DBMS, Montreal, April 1976, pp.181-206

[STONM73]

Stonebraker, M., "High Level Integrity Assurance in Relational Data Base Management Systems," ERL Report ERL-M473, Electronic Research Lab. College of Engineering, UC, Berkeley, Aug. 1973, p.40

[STONM75]

Stonebraker, M., "Implementation of Integrity Constraints and Views by Query Modification," Proc. International Conf. on Management of Data, May 1975, pp.65-78

[STONM76]

Stonebraker, M., Wong, E., Kreps, P. and Held, G., "The Design and Implementation of Ingres," ACM Transactions on Database Systems, Vol. 1, No. 3, Sept. 1976, pp.189-222

[STONM77B]

Stonebraker, M. and Rowe, L. A., "Observations on Data Manipulation Language and Their Embedding in General Purpose Programming Language," Proc. 3rd International Conf. on VLDB, Tokyo, Japan, Oct. 1977, pp.128-143

[STONM77A]

Stonebraker, M. and Neuhold, E., "A Distributed Data Base Version of Ingres," Proc. the 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, UC. Berkeley, Calif., May 1977, pp.19-36

[STONM78A]

Stonebraker, M., "A Distributed Data Base Machine," Memorandum Research Lab., University of California, Berkeley, Calif. 94720, May 1978, p.20

[STONM78B]

Stonebraker, M., "Concurrency Control and Consistency of Multiple Copies of Data in Distributed Ingres," Proc. the 3rd Berkeley Workshop on Distributed Data Management and Computer Networks, San Francisco, Calif., Aug. 1978, pp.235-258

[STONM79]

Stonebraker, M., "MUFFIN: A Distributed Data Base Machine," Proc. of the 1st Distributed Computer System, Huntsville, Oct. 1979, pp.459-469

[SUMMP75]

Summit, R. K., "Lockheed Experience in Processing Large Data Base for Its Commercial Information Retrieval Service," Journal of Chemical Information and Computer Sciences, Vol. 15, No. 1, 1-42, pp.4042

[SUNDB78]

Sundgren, B., "Data Base Design in Theory and Practice," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.3-16

[SUS 77]

Su, S. Y. W. and Liu, B. J., "A Methodology of Application Program Analysis and Conversion Based on Database Semantics," Proc. ACM SIGMOD International Conf. on Management of Data, Toronto, Aug. 1977, pp.75-87

[SUS 78A]

Su, S. Y. W. and Eman, A., "CASDAL: CASSM's Data Language," ACM TODS, Vol. 3, No. 1, March 1978, pp.57-91

[SUS 78B]

Su, S. Y. W. and Lupkiewicz, S., "MICRONET: A Microcomputer Network System for Managing Distributed Relational Databases," Proc. of the 4th VLDB, West-Berlin, Sept. 1978, pp.288-298

[SUS 79A]

Su, S. Y. W., "Cellular-Logic-Devices: Concepts and Applications," IEEE Computer, Vol. 12, No. 3, March 1979, pp.11-25

[SUS 79B]

Su, S. Y. W., Nguyen, L. H., Eman, A. and Lipouski, G. T., "The Architectural Features and Implementation Techniques of then Multicell CASSM," IEEE Trans. on Computers, Vol. C-28, No. 6, June 1979, pp.430-445

[TABAK78]

Tabata, K., Matsumoto, H. and Ohno, Y., "A DBMS for a Composite Type of Distributed Data Base," Proc. 3rd USA-Japan Computer Conf., Oct. 1978, pp.187-191

[TAKIM78]

Takizawa, M., Hamanaka, E. and Ito, T., "Resecure Integration and Data Sharing on Heterogeneous Resource Sharing System," Proc. ICCC '78, Kyoto, Japan, Sept. 1978, pp.253-258

[TAKIM79A]

Takizawa, M. and Hamanaka, E., "The Four-Schema Concept as the Gross-Architecture of Distributed Databases and Heterogeneity Problems," Journal of Information Processing, IPSJ, Vol. 2, No. 3, Nov. 1979, pp.134-142

[TAKIM79A]

Takizawa, M., "Query Translation in Distributed Databases," JIPDEC TR 79/04 JIPDEC, Oct. 1979 (To Appear in Proc. of the IFIP Congress 80, Tokyo, Oct. 1980)

[TAKIM80A]

Takizawa, M., "Distribution Problems in Distributed Database-Integration and Query Decomposition," Proc. of the JIPDEC Information Systems Seminar on Semantic Aspects of Databases (also Available from JIPDEC TR 80/01),Tokyo, Japan, Feb. 1980

[TAKIM80B]

Takizawa, M., "Operational Query Decomposition Algorithm in Distributed Databases," JIPDEC TR 80/02 JIPDEC, March 1980

[TANAK79]

Tanaka, K., Kambayashi, Y. and Yajima, S., "Organization of Quasi-Consecutive Retrieval Files," Information Systems, Vol. 4, No. 1, 1979, pp.23-34

[TANAY76]

Tanaka, Y., Miyashita, K., Koyama, S., Miyamoto, E. and Tsuda, T., "HARPS (Hokkaido University Array Processor System): A New Hierarchical Array Processor System," Proc. of the 2nd Symposium on Micro Architecture, 1976, pp.91-98

[TANAY77]

Tanaka, Y. and Tsuda, T., "Decomposition and Composition of a Relational Data Base," Proc. of the 3rd International Conf. on VLDB, Tokyo, Japan, Oct. 1977, pp.454-461

[TANAY79A]

Tanaka, Y., "Logical Design of a 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

[TANAY79B]

Tanaka, Y. and Tsuda, T., "Firmware System for Parallel Processing on a Multiprocessor HARPS," IMMM 79, Geneva, June 1979

[TAYLR76]

Taylor, R. W. and Frank, R. L., "CODASYL Data-Base Management System," ACM Computing Surveys, Vol. 8, No. 1, March 1976, pp.67-103

[TEORT77]

Teorey, T. J., "The Database Design Evaluator, Part 1: Overview," Working Paper DE 7.2-1 Data Translation Project, Graduate School of Business Administration, The Univ. of Michiga, Ann Arbor, Michigan, July 1977, p.35

[TEORT78]

Teorey, T. J. and Oberlander, L. B., "Network Database Evaluation Using Analytical Modelling," AFIPS Conf. Proc., Vol. 47, May 1978, pp.833-842

[THACC79]

Thacker, C. P., McCreight, E. M., Lampson, B. W., Sproull, R. F. and Boggs, D. R., "ALTO: A Personal Computer," CSL-79-11, XEROX Palo Alto Research Center, Aug. 1979

[THOMD75]

Thomas, D. R., "Data-Manager - A Free Standing Data Dictionary System," Database Journal, Vol. 16, No. 2, pp.14-17

[THOMR73A]

Thomas, R. H., "A Resource Sharing Executive for the ARPANET," AFIPS Conf. Proc., May 1973, pp.155-163

[THOMR73B]

Thomas, R. H., "A Resource Sharing Executive for ARPANET," AFIPS Conf. Proc., Vol. 42, May 1973, pp.155-163

[THOMR76]

Thomas, R. H., "A Solution to the Update Problem for Multiple Copy Data Bases which Used Distributed Control," BBN Report No. 3340, BBN, July 1976, p.50

[THOMR79]

Thomas, R. H., "A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases," ACM TODS, Vol. 4, No. 2, June 1979, pp.180-209

[TOANN79]

Toan, N. G., "A Unified Method for Query Decomposition and Shared Information Updating in Distributed Systems," Proc. of the 1st International Conf. on Distributed Computing Systems, Huntsville, Alabama, Oct. 1979, pp.679-685

[TODDS76]

Todd, S. J. P., "The Peterlee Relational Test Vehicle - A System Overview," IBM Systems Journal, Vol. 15, No. 4, 1976, pp.285-318

[TODDS77]

Todd, S., "Automatic Constraint Maintenance and Updating Define Relations, Proc. IFIP 77, 1977, pp.145-148

[TOTHK78]

Toth, K. C., Mahmoud, S. A., Riordon, J. S. and Sherif, O., "The ADD System: An Architecture for Distributed Databases," Proc. 4th International Conf. on VLDB, Berlin, Sept. 1978, pp.462-471

[TSICD75]

Tsichritzis, D. C., "A Network Framework for Relation Implementation," Proc. IFIP TC-2 Working Conf. on Data Base Description, Wepion, Belgium, Jan. 1975, pp.269-285

[TSICD76B]

Tsichritzis, D., "LSL: A Link and Selector Language," Proc. ACM SIGMOD International Conf. on Management of Data, Washington, DC, June 1976, pp.123-133

[TSICD76D]

Tsichritzis, D. and Lochousky, F., "Views on Data," Proc. the 2nd Share Working Conf. on Data Base Management Systems, Montreal, Canada, Apr. 1976, pp.51-65

[TSICD77]

Tsichritzis, D. C. and Lochovsky, F. H., "Data Base Management Systems," Computer Science and Applied Mathematics, A Series of Monographs and Textbooks, Univ. of Maryland, 1977, p.388

[TSICD78]

Tsichritzis, D. and Klug, A. (Eds.), "The ANSI/X3/ SPARC DBMS Framework," Report of the Study Group on Data Base Management Systems, March 1978, p.56 (Also available from Information Systems, Vol. 13, No. 3, pp.173-191, 1978)

[TSUBM79]

Tsubaki, M. and Hotaka, R., "Distributed Multi-Database Environment with a Supervisory Data Dictionary Database," Proc. of the International Conf. on Entity-Relationship Approach to System Analysis and Design, Los Angeles, Calif., Dec. 1979

[UHROP73]

Uhrowcz: K, P. P., "Data Dictionary/Directories," IBM, Systems Journal, Vol. 12, No. 4, 1973, pp.332-350

[UNGEE79]

Unger, E. A., McBride, R. A., Slonim, J. and Maryanski, F. J., "Design for Integration of a DBMS into a Network Environment," 6th Data Communication Symp.,Pacific Grove, CA., Nov. 1979, pp.26-34

[UNGEH77]

Ungerer, H., "EURONET: A New Comprehensive Information Utility for the European User," 1st International On-Line Information Meeting, London, Dec. 1977, pp.203-214

[VALLO76]

Vallarino, O., "On the Use of Bit Maps for Multiple Key Retrieval," Proc. of Conf. on Data: Abstraction, Definition and Structure, Salt Lake City, Utah, March 1976, pp.108-114

[VANDE77]

Vandijck, E., "Towards a more Familiar Relational Retrieval Language," Information System, Vol. 2, No. 4, 1977, pp.159-169

[VANDE78]

Vandijck, E., "An Overview of Current Relational Data Base Query Languages," Data Base Technology, INFOTECH State of the Art Report, Vol. 2, 1978, pp.423-441,

[VASSY79]

Vassiliou, Y., "Null Values in Data Base Management a Denotational Semantics Approach," Proc. of the ACM SIGMOD International Conf. on Management of Data, Boston, MA., May 1979, pp.162-169

[VERTH78]

Verhofstad, J. S. M., "Recovery Techniques for Database System," ACM Computing Survey, Vol. 10, No. 2, June 1978, pp.167-195

[WAGNR73]

Wagner, R. E., "Indexing Design Considerations," IBM Systems Journal, Vol. 12, No. 4, 1973, pp.351-367

[WAHB 80]

Wah, B. W. and Yao, S. B., "Dialog - A Distributed Processor Organization for Database Machines," Purdue Univ., West Lafayatte, IN 47907, 1980

[WALTD77]

Waltz, D. L. and Goodman, B. A., "Writing a Natural Language Data Base System," Proc. of the 5th IJCAI, Cambridge, MA, Aug. 1977, pp.144-150

[WALTD78]

Waltz, D. L., "An English Language Question-Answering System for a Large Relational Database," CACM, Vol. 21, No. 1, July 1978, pp.526-539

[WASSA79]

Wasserman, A. I. and Prenner, C. J., "Toward a Unified View of the Data Base Management, Programming Language and Operating Systems - A Tutorial," Information Systems, Vol. 4, No. 2,1979, pp.119-126

[WATKS78]

Watkins, S. W. and Kimbleton, S. R., "Network Access Technology - A Perspective," AFIPS Conf. Proc., Vol. 47, May 1978, pp.495-503

[WEDEH74]

Wedekind, H., "On the Selection of Access Paths in a Data Base System," Proc. of IFIP TC-2 Working Conf. on Data Base Management, 1974, pp.385-397

[WEINJ77]

Weiner, J. L., "Deriving Data Base Specifications from User Queries," Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, Berkeley, Calif., May 1977, pp.182-195

[WELDJ79]

Weldon, J. L., "The Practice of the Data Base Administration," AFIPS Conf. Proc., Vol. 48, New York, June 1979, pp.709-712

[WEYLS75]

Weyl, S., "An Interlisp Relational Data Base System," Technical Report 11, SRI, Nov. 1975, p.35

[WHITS75]

Colin Whitby-Strevens, "Current Research in Computer Networks," ACM, Vol. 6, No. 2

[WIEDG77]

Wiederhold, G., "Database Design," McGraw-Hill, Inc., p.658

[WIEDG78]

Wiederhold, Gio., "Management of Semantic Information for Databases," Proc. of the 3rd USA-Japan Computer Conf., S. F., Oct. 1978, pp.192-197

[WILLM77]

Williams, M. E., "On-Line Retrieval - Today and Tomorrow," 1st International On-Line Information Meeting, London, Dec. 1977, pp.1-15

[WINOT75]

Winograd, T., "Frame Representations and the Declarative-Procedural Controversy Representation and Understanding (Bobrou and Collins (Eds.)), Academic Press, 1975, pp.185-210

[WINOT76]

Winograd, T., "Understanding Natural Language," Academic Press, New York, 1976, p.195

[WIRTJ79]

Wirth, T. F., "ASTROL - An Associative Structure-Oriented Language," AFIPS Conf. Proc., Vol. 48, New York, June 1979, pp.721-725

[WONGE76]

Wong, E. and Youssefi, K., "Decomposition - A Strategy for Query Processing," ACM TODS, Vol. 1, No. 3, Sept. 1976, pp.223-241

[WONGE77]

Wong, E., "Retrieving Dispersed Data from SDD-1: A System for Distributed Databases," Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, Berkeley, May 1977, pp.217-235

[WOODC79]

Wood, C., Summers, R. C. and Fernandez, E. B., "Authorization in Multilevel Database Models," Information Systems, Pergamon Press, Vol. 4, No. 2, 1979, pp.155-161

[WOODH79]

Wood, H. M. and Kimbleton, S. R., "Access Control Mechanisms for a Network Operating System," AFIPS Conf. Proc., Vol. 48, New York, June 1978, pp.821-829

[WOODW75]

Woods, W. A., "What's in a Link: Foundations for Semantic Networks," Representation and Understanding (Bobrow, D. G. and Collins, A.), Academic Press, 1975, pp.35-82

[WOODW77]

Woods, W. A., "Shortfall and Density Scoring Strategies for Speech Understanding Control," Proc. of the 5th IJCAI, Cambridge, MA, Aug. 1977, pp.18-26

[WOODW78]

Woods, W. A., "Semantics and Quantification in Natural Language Question Answering," Advances in Computers, Vol. 17, Academic Press, 1978, pp.1-87

[YANGC78]

Yang, C. S. and Salton, G., "Best-Match Querying in General Database Systems - A Language Approach," Proc. of COMPSAC, Chicago, Ill., Nov. 1978, pp.458-463

[YAOS 77A]

Yao, S. B., "Approximating Block Accesses in Database Organization," CACM, Vol. 20, No. 4, April 1977, pp.260-261

[YAOS 77B]

Yao, S. B., "An Attribute Based Model for Database Access Cost Analysis," ACM TODS, Vol. 2, No. 1, March 1977, pp.45-67

[YAOS 78A]

Yao, S. B., Bernstein, P. A., Goodman, N., Schuster, S. A., Shipman, D. W. and Smith, D. C. P., "Database System," IEEE Computer, Vol. 11, No. 9, Sept. 1978, pp.46-60

[YAOS 78B]

Yao, S. B., "Optimization of Query Evaluation Algorithms," (Unpublished) March 1978

[YAOS 78D]

Yao, S. B. and Dejong, D., "Evaluation of Database Access Paths," Proc. of the ACM SIGMOD International Conf. on Management of Data, Austin, Texas, May 1978, pp.66-77

[YAOS 79A]

Yao, S. B., "Optimization of Query Evaluation Algorithms," ACM TODS, Vol. 4, No. 2, June 1979, pp.133-155

[YAOS 79B]

Yao, S. B., "Database Storage Structure Design," Proc. of the 13th Computer Science Symposium on Database Engineering, Amagi, Nov. 1979, pp.2-1-2-22

[YEHR 78A]

Yeh, R. T., Roussopoulos, N. and Chang, P., "Data Base Design - An Approach and Some Issues," INFOTECH State of the Art Report, "Data Base Technology," VD. 2, pp.443-477

[YEHR 78B]

Yeh, R. T. and Chandy, K. M., "On the Design of Elementary Distributed Systems," Proc. of the 3rd Berkeley Workshop on Distributed Data Management and Computer Networks, S. F., Calif., Aug. 1978, pp.289-321

[YEHR 78C]

Yeh, R. T., Chang, P. and Mohan, C., "A Multi-level Approach to Data Base Design," IEEE COMPSAC 78, Chicago, Ill., Nov. 1978, pp.370-375

[YORMB76]

Yormark, B., "The ANSI/X3/SPARC SGDBMS Architecture," Proc. 2nd Share Working Conf. on DBMS, Montreal, Canada, April 1976, pp.1-21

[YOUSK77]

Youssefi, K., Whyte, N., Ubell, M., Ries, D., Hawttaorn, P., Epstein, B., Berman, R. and Allman, E., "INGRES Reference Manual - Version 6.1", Memo. ERL-M529, Electronics Research Lab., UC, Berkeley, Oct. 1977

[YOUSK78]

Youssefi, K. and Wong, E., "Query Processing in a Relational Database Management System," UCB/ERL M78/17, Electronics Research Lab., UC, Berkeley, March 1978, p.28

[YOUSK79]

Youssefi, K. and Doing, E., "Query Processing in a Relational Database Management System," Proc. of the 5th VLDB, Rio De Janeiro, Brazil, Oct. 1979, pp.409-417

[YUC 78]

Yu, C. T., Luk, W. S. and Siu, M. K., "On the Estimation of the Number of Desired Records with Report to a Given Query," ACM TODS, Vol. 3, No. 1, Mar. 1978, pp.41-56

[ZADEL75]

Zadeh, L. A., Fu, K. S., Tanaka, K. and Shimura, M. (Eds.), "Fuzzy Sets and Their Applications to Cognitive and Decision Processes," Academic Press, New York, 1975

[ZANIC76]

Zaniolo, C. A., "Analysis and Design of Relational Schemata for Data Base Systems," Ph. D. Dissertation, Computer Science Dept., Univ. of California, Los Angeles, Calif., 1976

[ZANIC77]

Zaniolo, C., "Relational Views in a Data Base System Support for Queries," Proc. of COMPSAC, Chicago, Nov. 1977, pp.267-275

[ZANIC79A]

Zaniolo, C., "Design of Relational View Over Network Schemas," Proc. of the ACM SIGMOD International Conf. on Management of Data, Boston, MA., May 1979, pp.179-190

[ZANIC79B]

Zaniolo, C., "Multimodel External Schemas for CODASYL Data Base Management Systems," Proc. of the IFIP TC-2 Working Conf. on Data Base Architecture, Venis, Italy, May 1979, pp.171-190

[ZIMMH77]

Zimmermann, H., "The Cyclades Experience-Results and Impacts," Proc. IFIP Congress 77, Toronto, Aug. 1977, pp.465-470

[ZLOOM75A]

Zloof, M. M., "Query by Example," AFIPS Conf. Proceedings, May 1975, pp.431-437

[ZLOOM77B]

Zloof, M. M., "Query-by-Example: A Data Base Language," IBM Systems Journal, Vol. 16, No. 4, 1977, pp.324-343

[ZLOOM77D]

Zloof, M. M. and De Jong, S. P., "The System for Business Automation (SBA): Programming Language," CACM, Vol. 20, No. 6, June 1977, pp.385-396

[ZLOOM78]

Zloof, M. M., "Design Aspects of the Query-by-Example Data Base Management Language," Databases: Improving Usability and Responsiveness, Academic Press, 1978, pp.29-56

[ZLOOM79]

Zloof, M. M., "A Language for Office and Business Automation," Proc. of the 13th IBM Computer Science Symposium on Database Engineering, Amagi, Japan, Nov. 1979, pp.V1-V28

付記6. 英 文 資 料

Distribution Problems in Distributed Database Systems:

Integration and Query Decomposition

Abstract

There are two main problems in designing the distributed database systems (DDBSs): 1) heterogeneity problem and 2) distribution problem . The heterogeneity problem is how to overcome the differencies of the data models and languages of database systems (DBSs). The distribution problem is how to integrate such homogenized DBSs into one logical and virtual database system. Our approach called a four-schema structure (FSS), first, solves the former and then the distribution problem.

This paper concentrates on the distribution problem. It contains three main parts. The first one is called an integration which is a process of integrating the homogenized DBSs (call them local conceptual schemas or LCSs) into one logical DBS (call it a global conceptual schema or GCS). The second is a query decomposition (QD) which is a process of decomposing a global query into a sequence of queries executable on each DBS. The last is an information required by the QD and removed in the integration, which represents the correspondence between the GCS and LCSs.

We propose the GCS definition language (GSDL) which is an extension of a relational calculus language QUEL in the integration. In the QD, the algorithm for transmission of data which minimizes the required information and keeps it static based on the dynamic decisions is proposed . In the DI, we show its schema in a relational form.

Contents

1. Introduction

Distributed database systems which aim at cooperating existing database system through computer networks are called bottom-up distributed databases (abbreviated DDBSs). There are two problems in the bottom-up design of the DDBS:

1) how to solve the heterogeneities of database systems which compose the DDBS, and

2) how to integrate to one logical and virtual database system from database systems distributed on the network, whose databases have different logical meanings.

We call the former problem heterogeneity problem and the latter distribution problem [TAKIM78]. The database systems are composed of several layers [TSICD78] and each layer provides one data model and one language based on the model.

Database systems connected by the network can communicate with each other through one of the layers. Therefore, they are assumed to be a black box which provides one schema and language based on one data model.

There are two aspects in considering the heterogeneity of the database system. One is characterized by a data model and language provided by the database system. It is called the syntactic aspect. It is also a tool of describing and accessing the database. The other is called the semantic aspect of the database system, which represents the meaning of stored database. The heterogeneity of the DBS is defined as the differencies in these two aspects. The heterogeneity problems are to solve the differencies of the syntactic aspects of the database systems and the distribution ones are to overcome the differencies of the semantic aspects.

We have already discussed the former problem in [TAKIM79a,b]. So, we would like to concentrate on the distribution problems in this paper.

2. FourSchema Structure (FSS) Approach to Designing DDBS.

Let us consider the design of the bottom-up type DDBS. As mentioned in Ch. 1, the heterogeneity of the DBS means that they differ from each other in their semantic and/or syntactic aspects. Table 2.1 shows the heterogeneity relationships between them. The class 1 represents the case that both aspects of them are equivalent. In this case, it is easy to integrate them. In reality, only the directory representing the distribution of them is required. The class 2 is more complicated. Compared with the class 1, they differ in their syntactic aspects. In addition to the directory, the translation mechanism of languages and models, i.e. homogenization, is required. We can take EURONET as the examples. The DDBSs classified into the class 1 or 2 are called special-purpose type ones, because the semantics of databases are the same, and are dedicated to one application, e.g. document retrievals.

In the class 3, although the database systems have the same syntactic aspect, their semantic aspects are different. In this case, only the distribution problems have to be solved. That is, such different semantics have to be integrated into one semantic aspect. Designing the DDBS of the class 4 is the most difficult, because both syntactic and semantic aspects are different. Both integration and homogenization problems have to be solved. The DDBSs classified into the class 3 or 4 are called general-purpose type ones, because we can define new meaning of data suited for various applications in terms of existing

database systems. We are trying to design and implement the DDBS of the class 4.

Since the semantic aspect of the database system is independent of the syntactic one, the heterogeneity and distribution problems can be considered independently. As seen in Table 2.1, the bottom-up design of the general-purpose DDBS requires the integration function in addition to the homogenization function. Hence, we should, first, homogenize data models and languages, and then integrate the semantic aspects. In our approach to designing the DDBS, therefore, first of all, we describe the schema of each database system in terms of a common data model and provide a common access language for users. After that, we integrate such homogenized DBSs into a logical database system, whose meaning is of interest to a specific network community. Our approach is called a four-schema structure (abbreviated FSS) [TAKIM78, 79a].
The FSS as shown in Fig. 2.1 consists of the following components:

1) four schema layers: local internal schema (LIS), local conceptual schema (LCS), global conceptual schema (GCS), and external schema (EXS),
2) mappings among layers: i) integration (LCSs to GCS), ii) query decomposition (GCS to LCSs), iii) homogenization (LCS to LIS), iv) query translation (LCS to LIS), and
3) required information: network data directory (NDD) consisting of distribution information (DI), and heterogeneity information (HI).

Four schemas are discussed in 2.1, and the NDD is touched on in 2.2.

2.1 Four Schemas [TAKIM78, 79a]

Let us explain briefly four schemas of the FSS in this section. The schema of the lowest layer is called a local internal schema (LIS). It is a description of data which can be used under a distributed database system's situation. It may correspond to the schema or subschema level of the existing DBMS. It also may correspond to the conceptual or external schema of the ANSI/X3/SPARC[TSICD78]. The LIS is described in terms of one of the existing data models: relational, DBTG, and IMS models. At the LIS level, one access language based on the model is also provided. The heterogeneity of the LIS is defined as the differences of the data models and access languages.

The next schema layer is a local conceptual schema (LCS) layer. It is generated by describing the LIS in terms of a common model by a local administrator (LA) of each site. At the LCS level, a common language based on the common model is also provided. We adopt an E-R model [CHENP76] as the common model and QUEL [HELDG75, YOUSK77] as the common access language. The reason for adopting the E-R model is that it provides the simplest means of describing the concepts of the entities and relationships among them, and it is easy to correspond it to the relational model [CODDE70]. In such a way, the LCS corresponds to the level at which the syntactic aspects of the DBS, i.e. the data model and language, are homogenized.

The designing of the LCS layer from the LIS layer is called homogenization, in which the heterogeneity of each LIS is removed as heterogeneity information (HI). The HI represents the correspondence between the LCS and the LIS. In turn, the mapping of the LCS layer to the LIS layer is called a query translation (QT)[TAKIM79b, 80a], in which a query based on the LCS is translated into an executable sequence of the access languages based on the LIS. The process of the QT based

on the graph representation is discussed in [TAKIM79b, 80a]. Since the QT is an inverse mapping of the homogenization, it utilizes the heterogeneity information (HI) removed in the homogenization.

A global conceptual schema (GCS) is a description of the meaning of data which is of interest to a specific network community on the basis of the local conceptual schemas (LCSs) distributed over the network. It is described by a global administrator (GA) through negotiating with the local administrators (LAs), taking into consideration the requirements of the community. This process is similar to the requirement engineering. The GCS is expressed in terms of the E-R model like the LCS by means of a GCS definition language (call it a GSDL). A set of GSDL statements to define a GCS is called a distribution description (DD) for the GCS. This design process in which the GCS is integrated from the LCSs is called an integration. Also, the correspondence information between them are removed as a distribution information (DI). At the GCS level, users can access the DDBS without being any conscious of where the required data is located and what type of DBS manages it.

The last schema layer is an external schema (EXS) layer. It is a description of data required by a specific application of the community in terms of the data model suited for it. It is also described as a subset of the GCS by an application administrator (AA) taking into consideration the usability and security of data. The EXS is equivalent to the external schema of the ANSI/X3/SPARC, because at the GCS level the DDBS is virtualized to be a large database system. In this sense, the EXS can be discussed independently of the DDBS.

Therefore, we would not discuss it in this paper. The design process in which the EXS is generated from the GCS is called a specialization and its inverse process a generalization. The correspondence information between the EXS and GCS is called an external information (EI).

2.2 Network Data Directory (NDD)

The information required by the mapping between the LIS and LCS, heterogeneity information (HI), the LCSs and GCS, and the GCS and EXS, are respectively called the distribution information (DI), and external information (EI). They are correspondence information between respective schema layers. They are totally called a network data directory (NDD).

We have to discuss the NDD with respect to its content and distribution form. Concerning the contents, the HI and DI are managed in a relational form. The HI is described in [TAKIM79a]. The schema of the DI is discussed in Ch. 3.

In the DDBS, it is an important problem how to distribute the NDD, because the performance and control schemes depend on it. In my opinion, the DDBS functions, i.e. query translation (QT) and decomposition (QD), and the information, i.e. DI, required by them have to exist at the same site. Since the query decomposition system exists at every site in our architecture, from the access efficiency viewpoint, the DI should be distributed fully redundantly to each site. The more statistics the DI contains, the more complete and efficient decision the QD can do. However, supporting the more statistics result in the more redundancies and storage overheads. In order to reduce the overhead for controlling the consistency of and concurrent access to redundant copies, it should be as small and static as possible.

The HI should be maintained at the corresponding site non-redundantly, in order not to propagate changes of database system to the DDBS level.

2.3 Design and Implementation

The reason for adopting the E-R model as a common model is that it provodes the concepts of the entities and relationships among them in a simple manner, and can be well corresponded to the relational model. One of the problems of the relational model is said to be its lack of these concepts, i.e. its semantic problems. However, we believe that the major advantage of the relational model is its simplicity of description of and access to data. The more semantics the data model provides, the more complex the description and access become.

From the above observations, we employ the E-R model as a means of designing the DDBS because of its descrivability of semantics and the relational model as a means of describing and accessing data because of its simplicity. Hence, the LCSs and the GCS are described in a relational form. Such descriptions are called relational descriptions (RDs) of these schemas. Users can see and access the DDBS through the RD of the GCS and each database system through the RD of the LCS, based on the E-R model description of these schemas.

2.4 Unified Architecture

The distributed database system (DDBS) is based on the advances of techniques of the DBMS[TSCID78] and computer networks. The report [ANSIX75] of the ANSI/SPARC on DBMS has made clear three layers of the DBMS, i.e. internal, conceptual, and external schemas. The ISO/TC97/SC16 [BACHC78] and ANSI/SPARC on Distributed Systems have advanced the standardizations of protocols and shown the layers of protocols. Since the DDBS technique is a conjunction of both techniques, the problem for designing the DDBS is to make clear through which layer of protocols and at which level of the DBMS layers database systems communicate with each other via the network.

Our DDBS architecture is shown in Fig. 2.2, which aims at unifying both DBMS and communication architectures. So, we call it a unified architecture. The notation in Fig. 2.2 is based on the [ANSIX75]. [ANSIX75].

The upper half of this figure shows the bottom-up designing process. The local administrator (LA) sees the local internal schema (LIS) of his site through the display interface (interface 9) of the LIS processor and gives the local conceptual schema (LCS) definition to the homogenizer through interface 7. The homogenizer generates the LCS and the heterogeneity information (HI), and stores them in the NDD. The global administrator (GA) sees the LCSs distributed over the network through their display interface 6, and defines the GCS by a GCS definition language, i.e. GSDL, (interface 1) to the integrator. The integrator takes its definition, generates the GCS and the distribution information (DI), and stores them in the NDD. The application administrator (AA) sees the GCS through the display interface 3 and defines the external schema (EXS) to the EXS

processor. The EXS processor generates the EXS and the external information (EI), and stores them in the NDD.

On the other hand, Fig. 2.3 shows the top-down designing process. In this case, first, the GA defines the GCS and the objective functions based on the system and application requirements. The decomposer takes the GCS and decides the optimal alloçation of data to the sites. The information (DI) of correspondence between the GCS and LCSs is generated and stored in the NDD. That is to say, it gives the LCS to each site. The local administrator (LA) takes the LCS generated by the decomposer and defines the LIS. The heterogenizer generates the LIS from the LCS and the HI, i.e. the correspondence information between the LCS and LIS, and stores them in the NDD. As seen in Fig. 2.2 and 2.3, the role of the GA and LAs are different in the bottom-up and top-down designing.

The down half of Fig. 2.2 shows access to the DDBS. This part is also the same as the top-down designing. The EXS query based on the EXS is issued via the interface 21 and is translated to the GCS query based on the GCS, which is written in QUEL as stated in 2.1, by the EXS/GCS transform. The transform utilizes the EI in the NDD. The GCS/LCS transform, i.e. QD, takes the GCS query based on the GCS, and decomposes it into a sequence of the LCS queries.

The sequence includes the synchronization of executions of each LCS query, transmissions of data between sites, and error recovery. This requires a protocol to communicate with the QD and QTs.

The LCS/LIS transform, i.e. QT, takes the LCS query issued by the QD via the network, and translates it into a sequence of the LIS DMLs executable on the DBS.

3. Integration and Distribution Information

Under the assumption that each database system in the DDBS has been homogenized already, let us discuss the integration of the local conceptual schema (LCS) layer into the global conceptual schema (GCS) layer and the distribution information (DI) in this chapter. The global administrator (GA) defines the GCS by means of the language, GSDL, on the basis of the LCSs distributed over the network. Such a definition of the GCS, i.e. the set of the GSDL statements, is called a distribution description (DD). The DD represents not only the definition of the GCS but also the correspondence between the GCS and LCSs. This correspondence information is a main part of distribution information (DI).

At first, semantic relationships between the LCSs are discussed in 3.1. The DD is discussed in 3.2. The DI is touched on in 3.3.

3.1 Semantic Links

In order to integrate more than one database system, any semantic relationship has to exist among them. To represent such semantic relationships, we introduce the concept of semantic links. The semantic link can exist between two relations if both relations share the same domain. The semantic link also represents the set-theoretical relationship between two subsets of both relations [see Fig. 3.1].

In Fig. 3.1, A_1 and B_1 are subsets of relations A and B, respectively, and have the same scheme. The relationship A_1 EQ A_2 indicates that two relations A_1 and A_2 have the same occurrence. $A_1 \subseteq A_2$ shows that every occurrence of A_1 is included in A_2. A_1 $\cap$ A_2 represents that both relations

A_1 and A_2 have a set of tuples in common. $A_1 \cup A_2$ shows that A_1 and A_2 have the same scheme but do not have any tuple in common.

Fig. 3.2 shows the semantic relationship between three relations, i.e. PROJECTs at site 1 and 2, and PROJ at site 3. Fig. 3.3 shows semantic link based on Fig. 3.2.

3.2 Distribution Description (DD)

A set of the GSDL statements for defining a GCS is called a distribution description (DD) of the GCS. In reality, the GCS is derived from the RDs of the LCSs. It is also similar to the view definition [STONM76] of the relational model. In designing the relational database, starting from one universal relation, it is vertically decomposed into relations in a normal form. Hence, views which reference base relations are defined in terms of relational operations, i.e. projections, restrictions, and joins. But in the bottom-up type DDBS, relations at each site are not ones decomposed from one universal relation, but ones which have existed already at the site. This means that the union operation is required in addition to joins as multi-relation operations. This is similar to the relations horizontally decomposed from the relation.

Consequently, the following functions are required to make up the distribution description (DD):

1) naming GCS relations and attributes,

2) corresponding the GCS attributes to the arithmetic expressions over the LCS attributes,

3) relational oprations: joins, projections, restrictions, and unions, and

4) translations of units and representations of data (e.g. minutes to seconds, character to integer).

The required data can be defined completely and non-procedurally in the relational calculus language. Hence, QUEL is adopted as the common access language. It consists largely of two parts, target-list and qualification. The target list represents how the result attributes are expressed in terms of attributes of the relations referenced by the query. It plays a role of not only projection but also naming of the result attributes. The qualification represents a condition which the result relation has to satisfy in a formula including joins and restrictions. This means we can only vertically join relations. As mentioned before, the definition of the GCS reqiures the union of relations, i.e. horizontal composition of relations. But we cannot express the unions in QUEL. Hence, we extend QUEL so as to represent the union. Such an extended QUEL is called a GSDL.

Let us describe the GSDL. It is composed of three kinds of statements, say, drange, define, and drop statements. The drange statement is used to define tuple variables against the LCS relations and is written in a following form.

drange ($x_1,\ldots,x_m$) ($X_1:s_1,\ldots,X_m:s_m$) ;

each x_i,(for i=1,...,m), stands for tuple variable which ranges over the LCS relation X_i at the site s_i. Here, each X_i and each s_i need not be necessarily distinct respectively. For example,let

us suppose there exist three relations EMP, DEPT, SAL at sites 1, 3, 1, respectively. The following statement defines three tuple variables e, d, s against relations EMP, DEPT, SAL, respectively.

drange (e, d, s) (EMP:1, DEPT:3, SAL:1) ;

The define statement is used to define a GCS relation in terms of the LCS relations in a following form.

define < type > < gcs-rel-name > (< gcs-att-list >)
< sub-def > {: < sub-def >} ;

< gcs-rel-name > is a name of the GCS relation to be defined. < type > indicates whether this GCS relation is an ESR or RSR. < gcs-att-list > is a list of the GCS attributes which compose the GCS relation. In it, the unit of the attribute is also able to be defined. < sub-def > is called a subdefinition of the GCS relation and is in a following form.

< sub-def > ::= (< taget-list >) where < qual >

The target list and qualification are the same as QUEL. Like QUEL < sub-def > represents conventional vertical joins. The list of subdefinitions divided by colon means that the GCS relation to be defined is the union of the results each of which is derived from each subdefinition. In order to take the union of the relations, their schemes have to be the same. To do that, by the target lists of subdefinitions, the schemes of their result relations can be the same.

The drop statement plays a role of removing the GCS relation defined already. Its form is as follows.

drop <gcs-rel-name> ;

Let us take Fig. 3.3 as an example. Three LCS relations are related by means of semantic links as shown in Fig. 3.2. Fig. 3.4 shows the GCS relation of type ESR, PROJECT (pno, pname, manager, budget, loc), defined over three LCS relations, PROJECT at site 1, PROJECT at site 2, and PROJ at site 4.

3.3 Distribution Information (DI)

The distribution information (DI) represents the correspondence information between the elements of the GCS and the LCSs. The DI is initiated in the integration, and generated from the heterogeneity information (HIs) of sites and the distribution description (DD) defined by the CA.

1) schema information relation (SIR) ,

2) correspondence information relation (CIR), and

3) location information relation (LIR).

The schemes of these relations and inter-relationships between them are shown in Fig. 3.5. The SIR is the description of the schemes of the GCS relations.

The CIR represents the correspondence between the GCS relation and the LCS relations. The attribute, subdefinition, of the CIR takes relational clculas expressions defined in the DD as values.

The LIR represents where each LCS relation is located. In the LIR, the performance information of the LCS relation and attribute such as its cardinality, size, and width, can be also defined. Such information are also stored in the HI of the correponding LCS. That is, it results in the redundancy. We think that whether such performance information is kept redundantly in both DI and HI depends on the query processing strategy. In my opinion, the location information which represent the locations of the LCS relations have to be maintained at every site in order to gain reasonable performance. Hence, it is desirable for the DI not to include the large performance information. The other reason is that it is relatively dynamic, time-varying compared with the schema information in the SIR. If such dynamic information are strongly redundantly distributed, it becomes difficult to control the concurrency access to and consistency of them.

4. Query Decomposition (QD)

The QD decomposes a query based on the GCS (call it a GCS query or GQ) into a sequence of queries based on the LCSs (call them LCS queries or LQs). It is comosed of two subparts: translation of referenced GCS relations and processing of inter-site queries. Query modification technique[STONM76] can be adopted for doing the first.

To process inter-site query, either one relation has to be transmitted into the other via a network. Since the network causes a bottleneck of the DDBS due to its restricted capacity, the transmission order of relations affects on the DDBS performance. It is closely related to a way of designing the DDBS. In a case of top-down designing, data can be allocated to sites so that query processings are localized. Furthermore, sizes of intermediate results can be estimated more easily. However, in a bottom-up design case, sine data have existed already independently of applications' requirements, more inter-site processings may be required and it is difficult to estimate sizes of intermediates.

4.1 Basic Assumptions

On considering the QD, we make the following assumptions on the network:

1) It is a site-to-site type, bu not a broadcast one.

2) It is always lightly loaded. Therefore, there is no need for considering queueing delay.

3) Its communication cost depends on a distance, and its measure is time. A logical cost, LC_{ij}, is defined as a delay time for transmitting a packet from site i to j. It is also proportional to the number of hops between them.

4) Local processing costs are neglectable compared with communication

costs.

We also make the following assumptions on processings at sites:
1) Each site has a working space (WS) which is managed in a relational form. Relations transmitted from the other sites and results of joins are stored in the WS.
2) Each site has two kinds of logical processors: a global database processor (GDP) and local database processors(LDPs)[TAKIM79a]. The GDP plays a role of the QD and mamagement of the HI at each site. Especially,the GDP to which user states a GQ is called a coordinate GDP (CGDP) that is a centralized controller for processing the QD. The LDP is responsible for the QT and management of the HI and WS, and exists against a DBS. LDPs which support data required by the QD are cooperated under the CGDP's control.
3) Let us consider an inter-site join of relations r' at site i and r at j. When r' is transmitted form i to j, r' and site i are called a source relation and site, respectively, and r and site j a destination relation and site, respectively. A pair of a transmission of r' to j, and a join of r' and r at j is called a stage. It is a basic unit of the inter-site query processing. LDP_k stands for the LDP at site k.

4.2 Objectives

The purpose of the QD is to generate an optimal sequence of stages. The objectives of the QD which have been taken up[HEVNA78] are to minimize the communication cost and the response time. The network causes a bottleneck of the DDBS due to its restricted capability. On the other hand, each site has some processing capacities. Therefore,

in order to achieve these objectives, it is necessary to reduce the network traffic for query processing and process queries in parallel at multiple sites. Works which have been done so far[CHUW 79,HEVNA78, EPSTR78,WONGE77] aim at achieving the objectives. Their characteristic is that strategies for transmitting relations are determined by estimating sizes of intermediates in an off-line manner. This estimation is based on statistics on relations such as selectivities and cardinalities. The selectivity[HEVNA78,SELIP79] of an attribute is defined as the ratio of the attribute cardinality to the cardinality of the relation to which it belongs. This definition can be true under an assumption that values of the attribute are even distributed. But we think that there are still problems whether actual distribution of values follows well this assumption. In order to make the selectivity more precise, [CHUW 79] proposes a method such that selectivities of well-used values are accumulated in the directory each time the values are accessed. However, it is noted that the more precise statistics on selectivities we have, the more storages are required.

We argue the following points. First, the performance information like selectivities and cardinalities have a dynamic property compared with the schema information. Secondly, the QD and its required information, i.e. the DI, have to exist at the same site for the efficiency purpose. It implies that every site has to equip a full copy of the DI. If each site provides such dynamic information strongly redundantly, the overhead for not only storing them but also controlling consistency of and concurrent access to them becomes serious and enormous. Besides the objectives as stated above, therefore, we would like to add one objective, i.e. to keep the information

required by the QD as small and static as possible.

We can summarize these discussions as follows. First, if every site has the facility of the QD, the DI should not include the performance information like selectivities. Secondly, we cannot obtain necessary and sufficient information to decide strategies in an off-line manner. We think, therefore, that it is a good candidate algorithm for the QD to decide strategies operationally and keep the DI static and small.

4.3 Strategies

Our strategies for processing inter-site queries are as follows. First, the CGDP decides cosequent stages dynamically based on the statistics of result relations of the preceding stages. This results in small and static DI.

Secondly, the parts of the query that reference only one site are processed locally at the site before the inter-site parts are processed. Because the local processings are neglectable in cost and results in the reduction of sizes of relations to be transmitted.

Thirdly, if two relations at different sites are to be joined, the smaller one is transmitted to the larger through a path with minimum transmission cost. We do not consider strategies such that two relations at different sites are transmitted to the other site and joined there.

Lastly, even if all the stages issued by the CGDP do not complete, if there exists a relation not being processed and a path with transmission cost less than some threshold value, then it can be transmitted through the path. By that, more than one stage can be processed in parallel.

Our QD for executing these strategies is composed of four main modules: query normalization (QN), query modification (QM), initial local query processing (ILQP), and transmission scheduling (TS). The first two modules translate a GCS query into queries referencing corresponding LCS relations by means of query modification technique [STONM76]. The last two are concerned with the processing of inter-site queries.

4.4 Query Normalization (QN)

A qualification part of a GQ is generally written in an arbitrary boolean combination form of comparison predicates. First, it is normalized in a disjunctive normal form (DNF). Next, the normalized GQ is further decomposed into a set of queries each of which has each disjunct as its qualification. Such a query is called a decomposed GQ (DGQ). This process is called a horizontal query decomposition. Each DGQ can be executed independently. The result of the original GQ can be obtained by taking the union of results of the DGQs.

Then, for each DGQ, by looking up the DI, the semantic correctness of the target-list and qualification is checked. Finally, its tree representations are constructed.

The QN brings in the easiness of logical handling of relational queries. But, this does not mean that it results in the optimal processing of the queries. We think that, in order to adopt the query modification method, it is necessary to normalize queries in the DNF. There are still problems how to enhance the performance to process "or" like " x.a = y.a or y.a = z.a".

4.5 Query Modification (QM)

The DGQ is translated into queries referencing corresponding LCS relations by means of query modification[STONM76]. That is, first, GCS attributes in the DGQ are replaced by expressions defined over LCS attributes, which are stored in the DI. Then, qualifications in definitions of the referenced GCS relations are conjuncted with the DGQ qualification. The resultant query is called a global LCS query (GLQ). A GCS definition includes generally more than one subdefinition[see 3.1]. Hence, a GLQ is created with respect to each combination of subdefinitions each of which belongs to the definition of each GCS relation referenced by the DGQ.

Let us consider the GCS relation PROJECT in Fig. 3.4 and a following query: "find names and managers of the projects which exist at JIPDEC and whose budgets are more than 15,000,000 yen." It is also written in QUEL as follows:

```
range    pr    PROJECT;
DGQ: retrieve into R ( pr.pname, pr.manager )
     where pr.loc = "JIPDEC" and pr.budget ≥ 15000000;
```

It is in a DNF. By looking up the DI for the GCS relation PROJECT, we can find its definition as shown in Fig.3.4. From the 1st and 2nd subdefinitions marked(1) and (2), respectively, the following GLQs are gotten:

```
     range ( p1, p2, p )( PROJECT:1, PROJECT:2, PROJ:1 );
GLQ1: retrieve into R1 ( p.pname, manager=p1.leader )
     where p.loc = "JIPDEC" and p1.budget ≥ 15000000 and
     (1)   p.pno = p1.pno and p.pname = p1.pname and p.eyar = p1.eyar;

GLQ2: retrieve into R2 ( p.pname, manager=p2.leader )
     where p.loc = "JIPDEC" and p2.budget ≥ 15000000 and
     (2)   p.pno = p2.pno and p.pname = p2.pname and p.eyar = p2.eyar;
```

The result of the DGQ, R, is the union of R1 and R2.

4.6 Initial Local Query Processing (ILQP)

The GLQ references the LCS relations at different sites. Next problem is how to process such an inter-site query.

The GLQP can be divided into two parts. One references only one site, and the other multiple sites. A conjunction of both is an original GLQ's qualification. The former parts have to be processed, first, closedly at one site, because it results in reduction of relations to be transmitted and its cost is assumed to be neglectable. We call such a local processing an initial local query processing (ILQP).

The ILQP is composed of the following functions:

1) to make a query graph[TAKIM79b] of the GLQ, that is called a GLQ graph (GLQG)[see Fig. 4.1],

2) to classify nodes in the GLQG into groups each of which consists of the nodes at the same site and connected by join-links at the site,

3) to generate LCS queries (LQs) each of which corresponds to each group, and send them to the corresponding sites.

For example, let us consider the GLQG in Fig. 4.1. The boxes represent tuple variables. The links between nodes are join-links. The arrowed links are result-links that represent result attributes. The remaining links are restriction links. The dotted circles indicate the groups in which all nodes are locally connected.

The LQs generated in such a manner is written in QUEL. Its target list has to contain attributes for inter-site joining, along with the result attributes specified in the original GLQ. The result relations of the LQs are stored in the WS.

Fig.4.2a shows the query graph resulted by the ILQP. As seen in this figure, it contains only inter-site joins. Hence, it is called a join query graph (JQG).

4.7 Transmission Scheduling (TS)

We consider the generation of a transmission scheduling (TS), i.e. an optimal sequence of stages, from the JQG in this section. Let r' and r be a source relation at site i and a destination relation at j, respectively. The stage consists of two subparts: a transmission of r' from i to j, and a join of r' and r at j and storing of the result as r. Hence, let r':i → j, c(r':i → j), and r':i → j:r be such a transmission, its cost, and such a stage, respectively.

Our algorithm for generating the TS is operational but not static. This means that the CGDP decides consequent stages based on monitored information on results of preceding stages. To monitor such results, each GDP manages two kinds of directories along with the DI, i.e. logical transmission cost table (LCT) and query processing information (QPI). In the QPI, the performance information of intermediate results are stored in two relations: QPI/REL (site-no, rel-no, cardinality, width) and QPI/ATT (site-no, rel-no, att-no, width). They maintain the performance information on relations and their attributes produced by stages, respectively. Such information are carried back to the CGDP by ACKs of stages from destination LDPs.

Each LCT entry, LC_{ij}, shows the communication cost between sites i and j. Here, c(r:i → j) is $|r|*LC_{ij}$, where |r| stands for the size of a relation r.

A primitive unit of our algorithm is composed of following parts: decision of next stage, reduction of the JQG, and update of the QPI. Let us suppose that all the nodes in the JQG are marked FREE.

Suppose that a stage r':i → j:r is selected as next one. The CGDP modifies the JQG. First, it marks r' SOURCE and r DEST in the JQG. Then, join-links except one between r' and r, each of which corre-

sponds to a join-link incident on r', are attached to r. If r' has a result-link, it is also attached to r'. In relation to such modification of the JQG, the QPI/ATT is updated so as to meet the new scheme of r.

Let us consider the JQG in Fig.4.2a. Suppose that a stage, R4:4 → 3: R3, is selected. Then, the JQG is reduced to one in Fig.4.2b. A join-link, j_4', corresponding to j_4 is attached to R3. Thus, j_9 is a conjunction of j_4' and j_5. A result-link, o_2', corresponding to o_2 is also attached to R3.

The CGDP sends a transmission (T) command to site i and a join (J) command to j[see Fig.4.7]. The T is in a following form:

T(stage-no, s-site-no, s-rel-no, d-site-no, d-rel-no).

On receiving it, the LDP_i transmits r' to j, only if the LDP_j is ready for receiving. The J command, J (stage-no, s-site-no, s-rel-no, d-site-no, d-rel-no, target-list, qual, s-rel-size), means that

retrieve into d-rel-no (target-list) where qual ;

The s-rel-size is a size of r', which is maintained in the QPI. By it, the LDP_j can allocate the WS for receiving r'. The target-list includes join-attributes of r' with respect to its adjacent nodes except r and join-attributes of r with respect to its adjacent nodes except r' along with result-attributes of r and r'. On receiving it, if the WS is available for receiving r', the LDP_j sends WSA to the LDP_i and waits for r'. If r' is received, it joins r' and r, stores its result in the WS, and sends ACK which also carries the information of the result to the CGDP. The sequence of the executions of Js has to be the same as the CGDP's sending order.

On receipt of an ACK for a stage, r':i → j:r, the CGDP trys to reduce the JQG. First, it removes r' and its related join-links from the JQG. Then, the QPI relations are updated using information in the ACK.

That is, all the tuples concerning r' are deleted from these relations and the cardinality of r in the QPI/REL is updated by new value.

Next, we shall decide next stage. A stage, r':i → j:r, that satisfies the following conditions is selected:

1) r' is marked FREE,
2) r is adjacent to r' in the JQG,
3) r is marked either FREE or DEST, and
4) c(r':i → j)is not only the minimum in the JQG but also less than some threshold value (THV).

The 1st condition ensures that r' is not being executed. The 2nd condition guarantees that there exists a join referencing r' and r. The 3rd one ensures that, even if r is a destination of the other stages that have not completed yet, r' can be sent to r. It means that transmissions of more than one stage can be overlapped in a palallel manner. Since transmission costs are overwhelming, we think these overlappings are effective.

The last condition plays a role of protecting relations of larger size from being transmitted when relations of smaller size are currently being executed. If nodes with transmission cost < THV are not found, the CGDP waits for completions of stages being executed. If all nodes are marked FREE and no nodes satisfying this condition can be found, the THV value is reset using the QPI. We would like to determine its value through simulation studies. They are under investigation.

An ACK from a destination LDP to the CGDP, ACK (stage-no, d-site-no, d-rel-no, cardinality), carries the cardinality of the result relation of the stage. The scheme of the result is managed in the QPI by the CGDP.

The detailed description of our algorithm for the GDP is shown in Fig.4.3 and for the LDP in Fig.4.4.

4.8 An Example of the TS

Let us consider Fig.4.2. Suppose that all the ILQPs have finished, i.e. all nodes are marked FREE, and the LCT and sizes of relations are given in Figs. 4.5 and 4.6, respectively. For simplicity, let each LCT entry LC_{ij} be a minimum hop number between i and j. Let the THV value be 500. Let ST_k and ACK_k be the k-th stage and its ACK, respectively. The communication costs with respect to join-links are calculated as follows:

j2:	c(R1:1 → 2) = c(R2:2 → 1) = 500*2 = 1000	
j4:	c(R4:4 → 1) = 100*5 = 500	|R4| < |R1|
j5:	c(R3:3 → 1) = 200*2 = 400	|R3| < |R1|
*j6:	c(R4:4 → 3) = 100*1 = 100 < 500	|R4| < |R3|
j8:	c(R3:3 → 4) = 200*1 = 200	|R3| < |R5|

Hence, R4:4 → 3:R3 is selected as an ST_1 and T is sent to 4 and J to 3. R4 is marked SOURCE and R3 DEST. The JQG is modified as shown in Fig.4.2a.

As an ST_2, R5:4 → 3:R3 is selected[see Fig.4.2b], because R3 and R5 are marked DEST and FREE, respectively, and c(R5:4 → 3) = 300 < 500 that is also the minimum. Here, R4 and R5 are transmitted in parallel.

Then, let us try to decide an ST_3. Here, only R_1 and R_2 are marked FREE. Costs for possible transmissions are as follows:

j2:	c(R2:2 → 1) = c(R1:1 → 2) = 500*2 = 1000 > 500
j9:	c(R1:1 → 3) = 500*3 = 1500 > 500.

Hence, no satisfactory stage can be found. Since R3 is not marked FREE, we wait for ACK_1 and ACK_2. On receipt of the ACK_1, R4 is deleted from the JQG and QPI, and ACK_2 is waited for. On receipt of ACK_2, R5 is deleted and R3 becomes FREE. Suppose the size of R3 is 200. Since c(R3:3 → 1) = 600 > 500, the THV value is reset, i.e. THV ← (1500 +1000 +500)/3 = 1000.

So, ST_3 is R3:3 → 1:R1 and executed[see Fig.4.2c]. R3 is marked SOURCE and R1 DEST.

Since R2 is FREE and c(R2:2 → 1) = 1000, R2:2 → 1:R1 is selected as ST_4[see Fig.4.2d], and R2 is transmitted to R1. When both stages complete, the JQG is reduced to one node graph[see Fig.4.2e]. Since it is a final result, it is transmitted to the CGDP.

Fig.4.2e summarizes this example. The horizontal axis shows time.

4.9 The Architectures of the GDP and LDPs

Fig.4.7 shows the architectures of the GDP and LDPs. User's query is stated to the CGDP. The QN and QM take it and translate it into GLQs using the DI. The ILQP creates LQs from the GLQ, issues them to corresponding LDPs, and creates the JQG. The TS issues T and J commands for executions of stages generated from the JQG and controls their executions monitoring their intermediate results.

An LDP exists for one DBS. The LDP is composed of two main modules. The one is called a QT[TAKIM79b]. It translates the LQ written in QUEL into an executable sequence, e.g. DBTG DMLs, executes it, and stores the result as a relation in the WS. The other is a WS manager (WSM). It is composed of four submodules, WS, JOIN, TRANS, and REC. The WS is a storage for storing intermediates. It will be implemented as a SAM file. TRANS takes a T command from the CGDP and transmits the source relation. REC also sorts it on a join-attribute while receiving. The JOIN takes a J command and sorts the destination relation an a join-attribute. If the source relation is all received, JOIN joins them, stores the result as the destination relation, and sends ACK with the information on the result to the CGDP. Since both relations are sorted already, they can be easily joined by means of merge-join technique[SELIP79]. Thus, we think it is easy to implement the WSM.

5. Concluding Remarks

In this paper, we have presented mainly the distribution problems in the distributed database systems (DDBSs). The distribution problems consist of three subproblems: 1) integration, 2) query decomposition, and 3) distribution information (DI) of the NDD. The integration is a process which derives the GCS relations from the LCS relations. It is also similar to the view definition [STONM76] in the relational model. The main difference between them is that only join operations are used as multi-relation operations in the view definition, but the union operations are required in addition to joins in the integration. We have proposed a GCS definition language (GSDL) which is an extension of a relational calculus language QUEL so as to take the union of relations.

The set of GSDL statements is called a distribution description (DD) for the GCS. The correspondence between the GCS and the LCSs is expressed in a relational calculus form. It is also stored in the NDD as the distribution information (DI) in a relational form. It is desirable for the directory information to exist at the same site as the process which requires it. Hence, the process for the query decomposition (QD) has to have the DI at the same site. This means that the DI is stored fully redundantly at each site. In order that the DI is fully redundantly stored and the overhead for controlling the consistency and concurrency of redundant copies is reduced, the DI have to be as small and static as possible.

Our query decomposition algorithm which is called a TSA aims at minimizing the DI and keeping it static. The QD algorithms developed so far have been based on the estimation of the sizes

of the intermediate results. The more information on statistics of relations we have, the more complete decision of the strategies we can do. It requires the information of a large size which are also dynamic. That is to say, there exists a trade-off between complete decision of the strategies and the management of required information. From such observations that trying to decide the complete strategy in an off-line manner implies the large amount of information required and dynamic, we think that it is better to decide the strategy operationally.

Every LDP has to have the WS manager (WSM) which is also a relational DBS. One of its important task is to join two relations. It is very simple and easy to be implemented, because the source relation is sorted by the REC and the destination one is also sorted before joining.

We have implemented already the QN, QM, ILQP in the QD. We are now trying to implement the TS using our in-house computer network JIPNET.

Acknowledgement

We would like to thank Mr. E. Hamanaka, JIPDEC, for his useful discussions, and Mr. M. Suzuki, Systech Corp., who directs the implementation of our QD for his helpful discussions. We would also like to thank Ms. K. Yamamoto and Mr. S. Koseki, JIPDEC, for their helpful consultations.

References

[ANSIX75]

"Interim Report of the Study Group on Data Management,"

ANSI/X3/SPARC DBMS Study Group Report 75-02-08, Feb. 1975.

[BACHC78]

Bachman, C.W., "Provisional Model of Open System Architecture,"

Proc. of the 3rd Berkeley Workshop on Distributed Data Management

and Computer Networks, San Francisco, Aug. 1978, pp. 1-18.

[CHU W79]

Chu, W.W. and Hurley, P., "A Model for Optimal Query Processing for Distributed Data Bases," Proc. of the IEEE Compcon 79 Spring, Feb. 1979, pp.116 - 122.

[CHENP76]

Chen, P.P., "Entity-Relationship Model - Toward a Unified View of Data," ACM TODS, Vol. 1, No.1, Mar. 1976, pp. 9 - 36.

[CODDE70]

Codd, E.F., "A Relational Model of Data for Large Shared Data Bank," CACM, Vol. 13, No.6, June 1970, pp. 337 - 387.

[EPSTR78]

Epstein, R., Stonebraker, M., and Wong, E., "Distributed Query Processing in a Relational Data Base System," UCB/ERL M79/18, Electronics Research Lab., UC. Berkeley, April 1978.

[HELDG75]

Held, G.D., Stonebraker, M., and Wong, E., "INGRES - A Relational Data Base System," AFIPS Conf. Proc., May 1976, pp. 409 - 416.

[HEVNA78]

Hevner, A.R. and Yao, S.B., "Query Processing on a Distributed Database," Proc. of the 3rd Berkeley Workshop on Distributed Data Management and Computer Networks, San Francisco, CA., Aug. 1978, pp. 91 - 107.

[SELIP79]

Selinger, P.G., et al., "Access Path Selection in a Relational Database Management System," Proc. of the ACM SIGMOD, Boston, MA., May 1979, pp. 23 - 24.

[STONM76]
Stonebraker, M., Wong, E., Kreps, P., and Held, G., "The Design and Implementation of INGRES," ACM TODS, Vol. 1, No. 3, Sept. 1976, pp. 189 - 222.

[TAKIM78]
Takizawa, M., Hamanaka, E., and Ito, T., "Resource Integration and Data Sharing on Heterogeneous Resource Sharing System," Proc. of the ICCC'78, Kyoto, Japan, Sept. 1978, pp. 253 - 258.

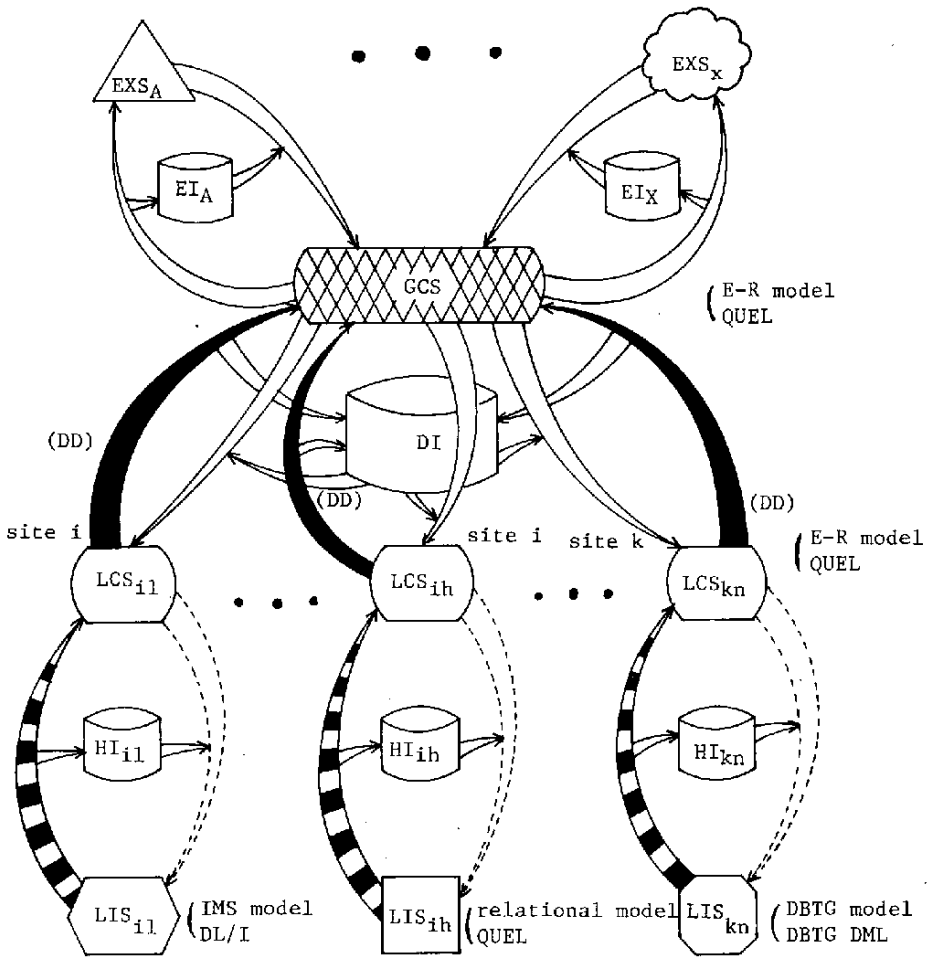
[TAKIM79a]
Takizawa, M. and Hamanaka, E., "The Four-Schema Structure Concept as the Gross Architecture of Distributed Databases and Heterogeneity Problems," Journal of Information Processing (JIP), Information Processing Society of Japan (IPSJ), Vol. 2, No. 3, Nov. 1979, pp. 134 - 142.

[TAKIM79b]
Takizawa, M. and Hamanaka, E., "Query Translation in Distributed Databases," JIPDEC TR79/04, Oct. 1979 (to appear in Proc. of the IFIP Congress, Tokyo, Oct. 1980).

[TAKIM80]
Takizawa, M., "Operational Query Decomposition Algorithm," JIPDEC TR80/02, March 1980.

[TSICD78]
Tsichritzis, D. and Klug, A.(eds.), "The ANSI/X3/SPARC DBMS Framework," Information System, Vol. 3, No. 3, pp. 173 - 191.

[WONGE77]
Wong, E., "Retrieving Dispersed Data from SDD-1: A System for Distributed Databases," Proc. of the 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, Berkeley, May 1977, pp. 217 - 235.



: homogenization

: integration

: query translation

: query decomposition

HI : heterogeneity information

DD : distribution description

DI : distribution information

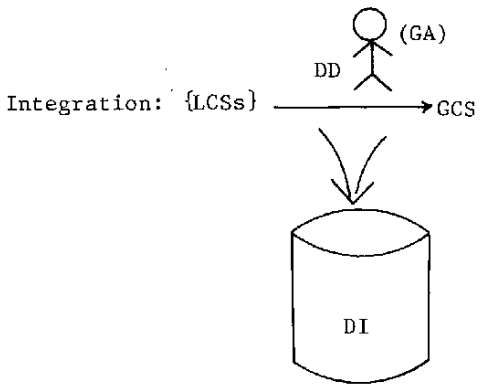
LIS : local internal schema

GCS : global conceptual schema

LCS : local conceptual schema

EXS : external schema

Fig. 2.1 Four-Schema Structure (FSS)



DD: distribution description
DI: distribution information
GA: global administrator
GCS: global conceptual schema
LCS: local conceptual schema

Fig. 3.1 Overview of Integration

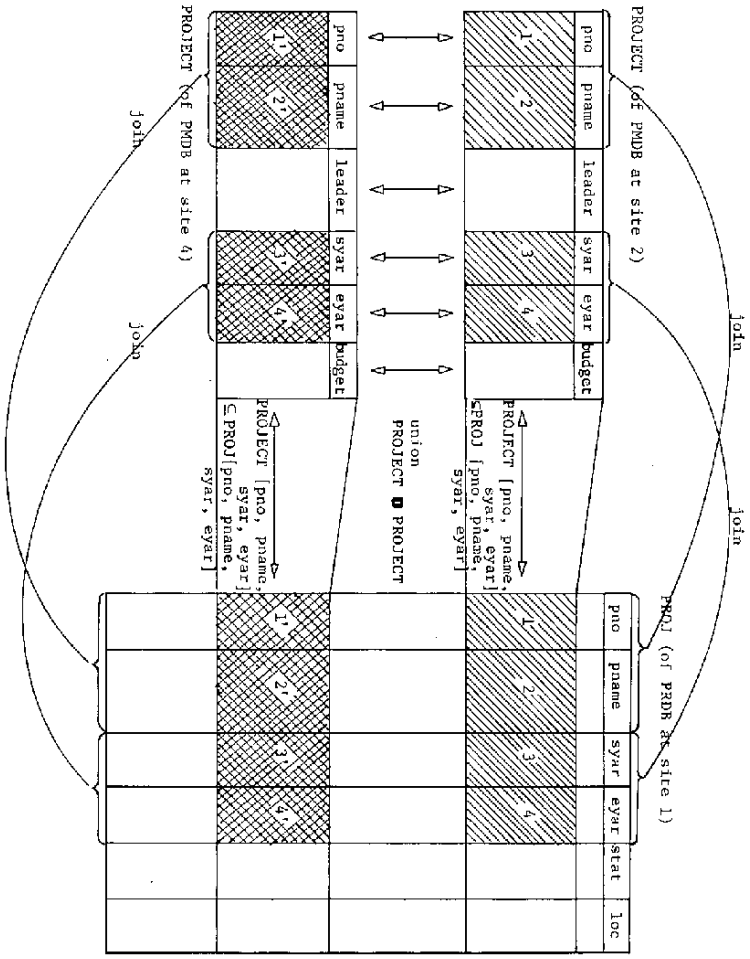


Fig. 3.2 Relationships Between Three Relations

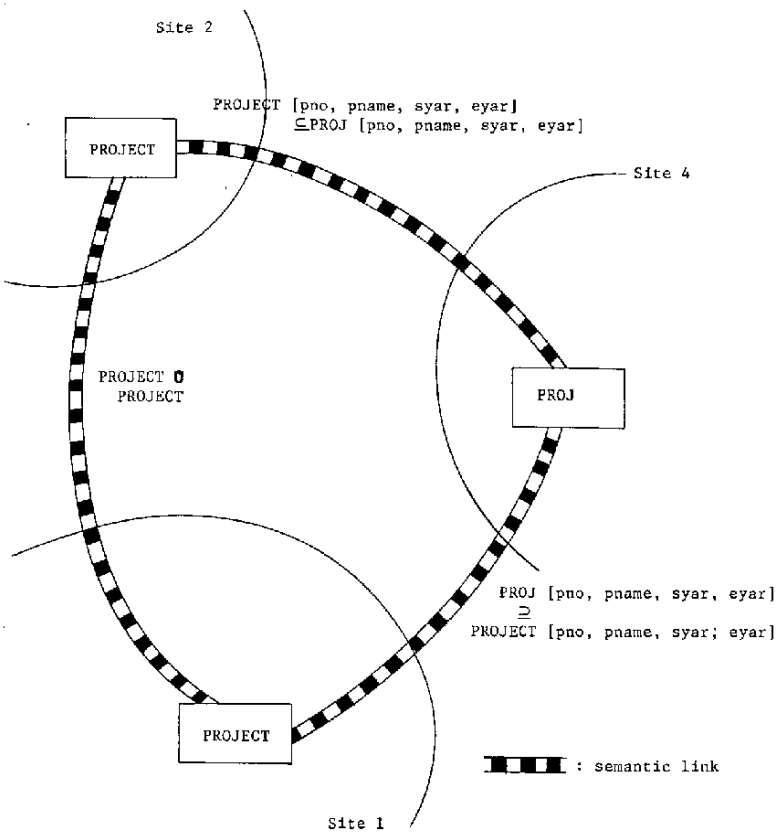


Fig. 3.3 Semantic Links Between Three Databases

```
drange ( p1, p2, p )  ( PROJECT:1, PROJECT:2, PROJECT:4 );
define  ESR PROJECT ( pno, pname, manager, budget, loc          )
(1) { ( p.pno, p.pname, manager=p1.leader, budget=p1.budget, p.loc )
      where  p.pno = p1.pno and p.pname = p1.pname and p.syar = p1.syar and
             p.eyar= p1.eyar ;
(2) { ( p.pno, p.pname, manager=p2.leader, p2.budget, p.loc          )
      where  p.pno = p2.pno and p.pname = p2.pname and p.syar = p2.syar and
             p.eyar= p2.eyar ;
```

Fig. 3.4 The DD of the GCS relation PROJECT

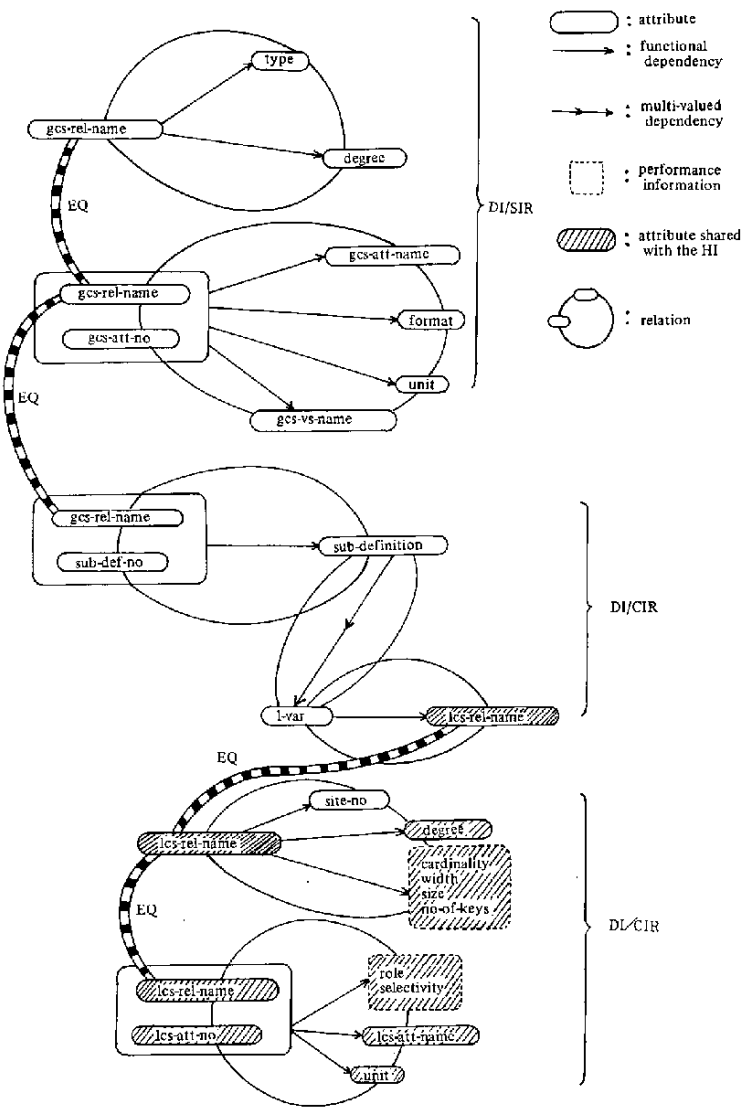
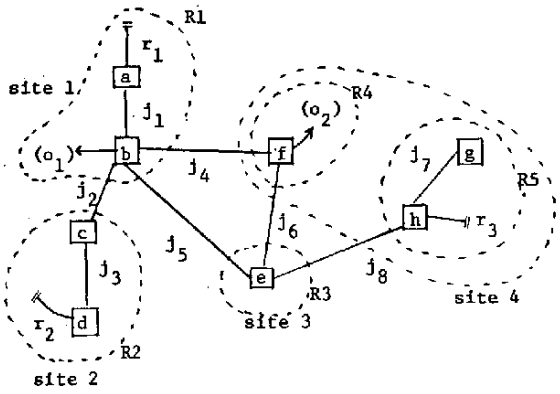


Fig. 3.5 Distribution Information (DI)



□ : node for a tuple variable

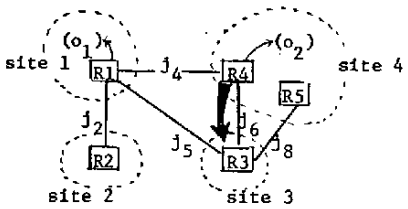
Ri: result relation generated by the ILQP.

□—j—□ : join link

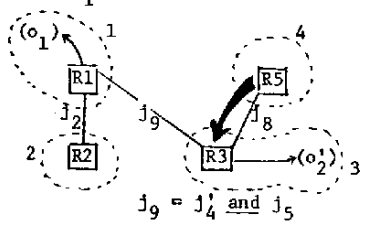
—r—□ : restriction link

□—→(o) : result link

Fig. 4.1 An Example of the GLQG

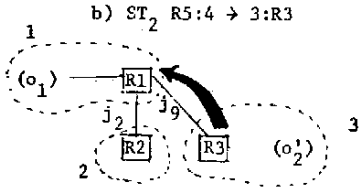


a) ST_1 R4:4 → 3:R3

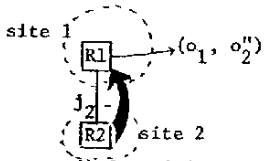


j_9 = j_4' and j_5

b) ST_2 R5:4 → 3:R3

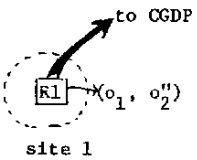


c) ST_3 R3:3 → 1:R1

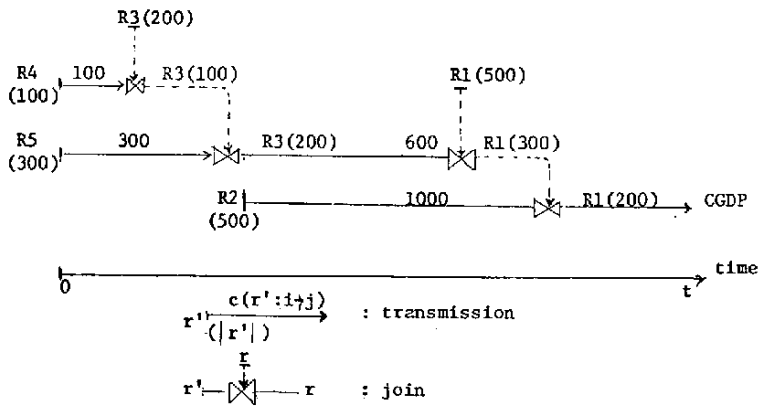


d) ST_4 R2:2 → 1:R1

: transmission



e) final result



f) a summary

Fig. 4.2 An Example of the TS

CGDP algorithm

0) [assumptions]

- let r':i → j:r be a stage where r' and r are a source relation at site i and a destinationa relation at site j.

1) [initial local query processing]

- an ILQP is considered as a stage :→ j:r where r is a result relation of it.
- form the LQs each of which corresponds to each subgroup which consists of the nodes not only at the same site but also connected by join links.
- send the LQs to the corresponding sites.
- create the JQG from the GLQG, whose nodes are the results of the ILQP and links are the inter-site joins.
- mark all the nodes in the JQG "FREE".
- initiate the QPI which contains all the relation schemes corresponding to the nodes in the JQG.
- initiate the THV (threshold value) using the DI.

2) [wait for ACKs]

- if the ACK from r is received,
  then if r' corresponding to r is NIL, i.e. ACK for the ILQP,
  then go to 4).

3) [reduction of the JQG]

- delete r' from the JQG.
- delete the join-links incident on r' from the JQG.
- delete tuples concerning r' from the QPI.

4) [update of the QPI]

- if the ACK is not for the most recent stage to r, then go to 2).
- update the information of r in the QPI using the information carried by the ACK.
- mark r "FREE".

Fig. 4.3 TS Algorithm for the CGDP (1)

5) [final result]

- if the reduced JQG contains only one node,
  then send it to the CGDP, i.e. the node is a final result.
  terminate.

6) [decide a next stage]

- select the nodes r' as a source node and r as a destination
  note such that they satisfy the following conditions:
  i) r' is marked "FREE",
  ii) r is adjacent to r' in the JQG,
  iii) r is marked either "FREE" or "DEST", and
  iv) c(r':i → j) is the minimum and less than the THV value.
  Here, $c(r':i \rightarrow j) = |r'| * LC_{ij}$.

7) [form a stage and send it to the destination site]

- if such r and r' are found,
  then
  - form a stage, r':i → j:r.
  - send a transmission command (T) to site i and
    a join commands (J) to site j.
  - mark r' "SOURCE" and r "DEST".
  - set up the join-links between r and the nodes adjacent to r'
    except r, each of which corresponds to an link between r'
    and each node adjacent to r'.
  - if r' has the result-link, move it to r.
  - update the QPI/ATT so as to meet a new scheme of r.

8) [satisfiable r and r' are not found]

- if all the nodes are marked "FREE",
  then reset the THV using the QPI. go to 6)
  else go to 2).

Fig. 4.3 TS Algorithm for the CGDP (2)

LDP algorithm

TRANS

1) if the transmission command (T) is received from the CGDP,
   wait for the WSA (WS allocated) from the destination site.

2) if the WSA is received,
   transmit the source relation along with its scheme to the
   destination site.

3) if the ACK for the transmission is received, then release the
   source relation, r'.

JOIN

1) if the join command (J) is received from the CGDP,
   then if the WS is available, then send WSA to the source site
      sort r' on the join attribute, wait for transmission.

2) if all the source relation is received,
   send ACK to the source site.
   join the source relation to the destination relation with
   respect to the target-list and qualification in the join
   command using a merge-join (SELIP79].
   form the information on the statistics of the result relation
   of this join.
   send ACK along with this information to the CGDP.

QT

1) if the LQ is received from the CGDP,
   translate the LQ into a DML program by the QT.
   execute the DML program.
   store the result to the WS as a relation.
   send ACK to the CGDP.

REC

1) sort r' on the join attribute while receiving it.
2) if r' is received, send ACK to the source site.

Fig. 4.4 TS Algorithm for the LDP

i	j	LC_{ij}
1	2	2
1	3	2
1	4	5
2	3	2
2	4	2
3	4	1

i,j = site numbers

Fig. 4.5 Logical Transmission Cost Table (LCT)

LCS relations	sizes
R1	500
R2	500
R3	200
R4	100
R5	300

(in byte)

Fig. 4.6 The Sizes of Relations

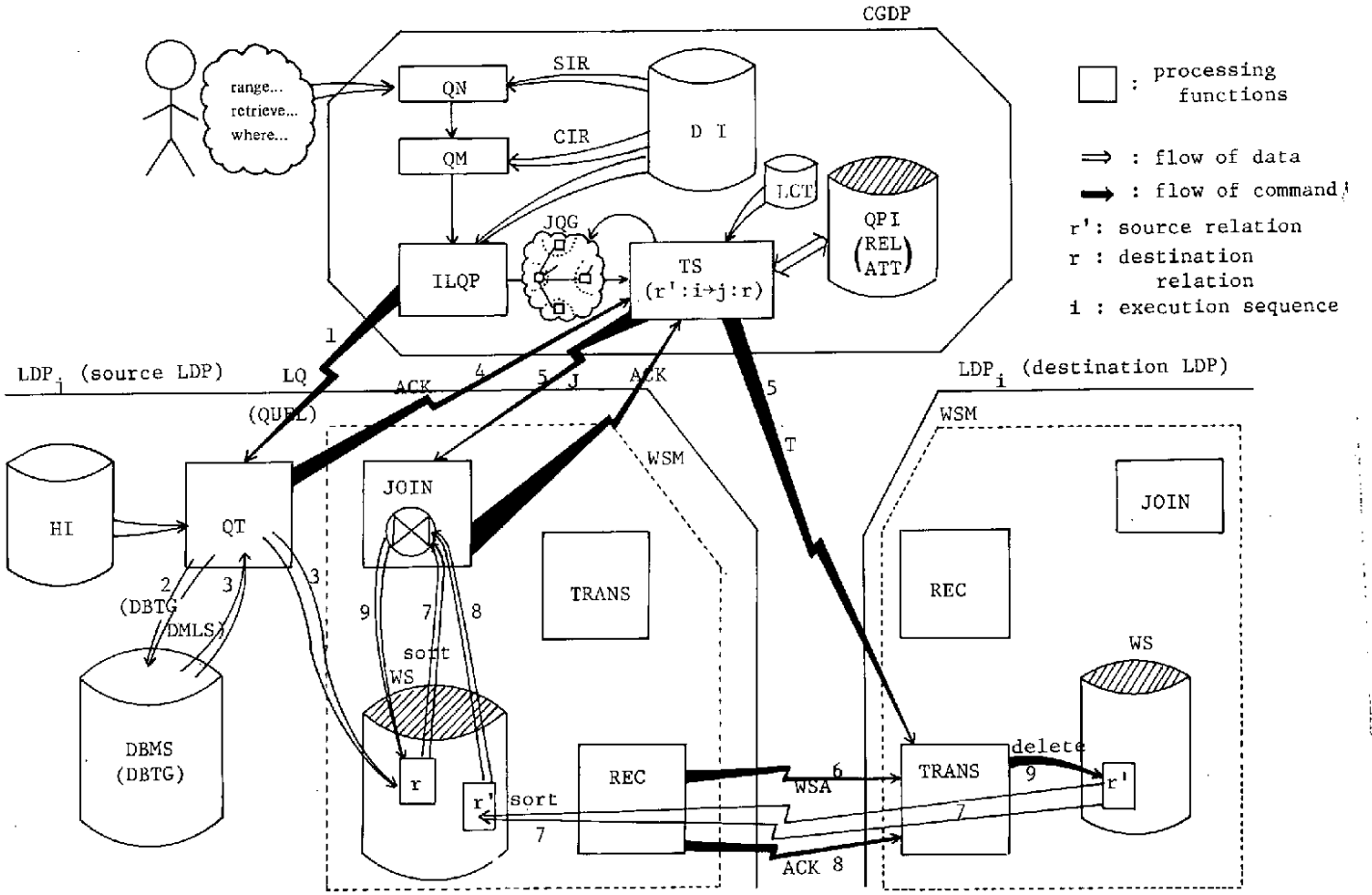


Fig. 4.7 The Architectures of the CGDP and LDPs

昭和５５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　昌　文　社
東京都港区芝５－２６－３０
ＴＥＬ　(452)4931

54－S001