データベースの意味論的側面に関する研究会報告書

Proceedings of the JIPDEC Information Systems Seminar on Semantic Aspects of Databases,

JIPDEC, Tokyo, Japan,

Feb. 21～22, 1980

昭和55年2月

財団法人 日本情報処理開発協会

Japan Information Processing Development Center
Kikai Shinko Kaikan,
5-8, Shibakoen 3 Chome, Minato-ku, Tokyo, 105, Japan
TEL: Tokyo (03) 434-8211
Cable Address: INFOPRODEC TOKYO

PROCEEDINGS OF

JIPDEC Information Systems Seminar on Semantic Aspects of Databases

Sponsored by

Development Department
Japan Information Processing Development Center(JIPDEC)

Feburary 1980

General Chairman: Kinko Yamamoto, JIPDEC

Program Cochairman: Yoshifumi Masunaga,
University of Tohoku
Makoto Takizawa, JIPDEC

JIPDEC Information Systems Seminar on Semantic Aspects of Databases,
JIPDEC, Tokyo, Feb. 21-22 1980

Program

I. General Topics Feb.21 13:10-14:10
Chairman Osuga,S., Univ.of Tokyo
1. "データベースの最近の話題,"
Kambayashi, Y., Kyoto Univ.
2. "マギル大学に滞在して,"
Kambayashi, Y., Kyoto Univ.

II. Data Models-I 14:10-15:00
Chairman Ohsuga,S., Univ. of Tokyo
3. "Infomation Space Model of a Database,"
Tanaka,Y., Hokkaido Univ.

III. Knowlege Bases and View 15:20-17:50
Chairman Kambayashi,Y., Kyoto Univ.
4. "Uses of Knowlege Bases in Data Base Management Systems"
Ohsuga,S., Univ. of Tokyo
5. "An Application of an External Virtual Relation and a Kowledge Filter to Deductive Question-Answering,"
Kunifuji,S., Wakagi,T., Fujitsu
6. "LUP:A Vehicle of Impelementing a View Support Subsystem of a Relational DBMS,"
Masunaga,Y.,. Tohoku Univ.

IV. Data Models-II Feb.22 9:00-10:40
Chairman Kunifuji,S., Fujitsu
7. "Logical Design of a 4NF D-tree Schema for a Relational Database,"
Tanaka,Y., Hokkaido Univ.
8. "Preserveness of Data Dependencies for Relational Operations,"
Tanaka,K., Kobe Univ.

V. Infomation Systems-I 11:00-12:40
Chairman Tanaka,Y., Hokkaido Univ.
9. "A Database System for Processing of Skull Line Drawing in Orthodonitics,"
Kanamori,Y., Tohoku Univ.
10. "Implementation and Evaluation of Relational Inverted Structure(RIS)Files,"
Le Viet Chung, Kambayashi,Y., Tanaka,K., Yajima,S., Kyoto Univ.

VI. Information Systems-II Feb.22 13:40-15:20

Chairman Masunaga,Y., Tohoku Univ.

11."A Requirements Engineering Approach for Database Design,"
Agusa,K., Tabata,K., Ohnishi,A., Ohno,Y., Kyoto Univ.

12."Distribution Problems in Distributed Databases:Integration and Query Decomposion,"
Takizawa,M., JIPDEC

VII. Free Discussions 15:20-16:00

Chairman Ohsuga,S., Univ. of Tokyo

"Relatinal Model and Dependency Theories"

データベースの最近の話題

上林弥彦
（京都大学工学部）

1．まえがき

データベースシステムは，多数の利用者がデータを共有し利用する目的で構成される。計算機科学におけるデータベースは，データそのものよりも，このデータベースシステムにかかわる諸問題を扱うものである。

データベースは比較的歴史のある計算機応用の分野ではあるが，最近まで計算機科学の中心分野としては認められていなかった。しかし，1975年には，2つの代表的なデータベースに関する国際会議（ACM SIGMOD International Conference on Management of Data，International Conference on Very Large Data Bases）が始まり，1976年には主としてデータベースを扱う学術誌（ACM Transaction on Database Systems と Information Systems）も発刊され，さらにデータベースを主要なテーマとして扱う会議（IEEE International Computer Software and Applications Conference，Berkeley Workshop on Distributed Data Base and Computer Networks）も増えつつあり，計算機科学の一つの中心分野に育ちつつある。また，計算機基礎論，人工知能，計算機アーキテクチュア，CAD等の分野の雑誌や国際会議にもデータベースに関係した論文が掲載されるようになり，その他の各種応用分野も含めて，その重要性が認識され始めている。北米の大学では，この分野の求人も多く，大学院学生に人気のある分野の一つとなってきており，教育面でも急速に充実しつつある。

このようなデータベースの人気は，計算機システムの最近の進歩によって高機能のデータベースが経済的に実現できるようになり，従来潜在的であったデータベースの需要を顕在化させたことが大きな原因となっているが，データベースの学問的な意味での最近の発展も重要な原因である。オペレーティングシステムや言語プロセッサ等計算機ソフトウェアの分野では，ある程度実際面の経験が蓄積して理論化の可能性や理論的手法によって解決すべき問題があきらかになってから理論化が行なわれ，それがさらにその分野の高度化に役立ったという歴史があり，現在データベースはこのような理論化の波にあらわれているといえる。データベースよりも歴史が古く，性格的にもよく似た情報検索の分野も，体系化へと進むきざしを見せており，これらデータ科学の理論化とその発展は80年代の計算機科学の一つの目となるであろう。

人間と計算機との対応で言えば，計算機科学の基本をなすべきものは，

(1) 本体の構成に対応する，論理回路，ハードウェア，ソフトウェア，システム
(2) 人間の脳の働らき（計算，推論，記憶）および感覚等（目，口，耳）の入出力機能に対応して

計算	アルゴリズムの設計と解析，数値解析
推論	人工知能
記憶	データ科学（データベース，データ構造，情報検索）
感覚等	パターン処理（画像，音声等），計算機通信

および，それらの (3) 基礎理論 と (4) 応用 であると考えられる。この意味から，記憶に対応するデータ科学は，計算や推論に対応する分野と同じ位の重要度

を計算機科学の中で占めて当然であると考えられ，さらに計算機応用の分野ではこの比重が大きいので，重要性が増してゆくのは当然の方向であろう。

2. データベースシステムに関する話題

データベースシステムは単なるファイルシステムと異なるいくつかの特色を持っている。主要な特色として次のようなものがある。

(1) データの共有：データを共有することにより冗長度を減らし，コストを低減させると共に，データ全体を一律に管理することができる。

(2) ビューの実現：上記のデータの共有による利点を生かしながら，各利用者は個別ファイルと同じような自由度でシステムを使えるようにするため，各利用者の扱うデータの範囲を規定する概念である。

(3) 各種独立性の達成：システムのハードウェア構成やファイル構成，システム内部でのデータのモデル，データベースの利用形態等を分けて定義することができ，それぞれの部分の変更はデータベースの対応する部分にしか影響せず，全面的なデータベースの作り直しは必要でないこと。

(4) データベースのモデルの利用：上記の各定義のために利用できるモデルが必要である。実体とそれらの間の関係等を中心としたモデルを用いて，各定義の形式化やそれらの間のインタフェースの定義が可能となっている。

しかしながら，これらはデータベースシステムの目標とするところとなっていて現状では完全には達成されていない部分が多い。そのことがまたデータベース研究のための動機ともなっている。たとえば，データの共有は，個々の利用者のデータの操作に限界を設ける必要が生じるし，個々のデータに対する利用権の認証さらにはデータの安全性の問題等も生じる。また，何人もの利用者が同時にデータベースを利用するときに効率良く正しく処理するための並行処理や，巨大さに伴う欠点（O(n)以上の次数のアルゴリズムは，nが大きいと直接適用が困難となる）等に対する考慮も必要となる。ビューの自由度も共有という目的によって制約を受けることが多い。各種独立性の達成のためには，関係モデルが主要な3つのモデル（階層，ネットワーク，関係）のうちで最も適しており，質問の自由度も高いが効率は悪いのが問題である。関係モデルに基づくデータベースの諸問題の定式化は他のモデルに基づくものより単純で，そこでの解法を他のモデルの場合にも適用することができるため，データベースの問題を解く上でも関係モデルは重要である。実際上は，階層モデルやネットワークモデルに基づくシステムの方が効率が高いため，効率と機能・自由度とのかね合いでこれらの3種のシステムはともに残ってゆくと思われる。

階層モデルおよびネットワークモデルに基づくデータベースシステムは商用のものが多く知られているので，話題の中心は関係モデルであろう。代表的なシステムにおける，主要な機能の実現は次の表に示されている（O印は実現を示す）。

システム名	開発場所	ビュー	利用権の制御	保全性	並行処理
PRTV	英国IBM	O	—	—	—
MAGNUM	Tymshare社	—	—	—	—
INGRES	カリフォルニア大学バークレー分校	O	—	O	O
SYSTEM R(SEQUEL)	IBMサンホセ研究所	O	O	O	O
QBE	IBMワトソン研究所	O	—	O	—

このように，関係データベースシステムは，可能な機能を十分に実現しているとはいえない段階であるが，すでに商用のものとして，カリフォルニアのTymshare社のMAGNUM（PDP10 のもとで働らく），Honeywell のMULTICSのもとで働らくシステム，IBMのQBE（Query By Example）等が知られている。日電のINQは簡易関係データベースとして成功しているものである。IBMのSYSTEM Rの効率はかなり高いものだと言われており，商用版の開発を行っているのではないかとの噂もあり，これが市販されれば最初の本格的な関係データベースシステムになると考えられている。

データベースシステムの管理に必要な情報を記憶するシステムを，データ・ディクショナリィ・システム DDS と呼ぶ。これは本来データベース管理者に対して，データベースの設計・開発・実行・管理のための種々の情報を提供するものであったが，その機能を拡張して他の目的にも利用しようという試みがある。たとえば，

- データの使われ方を知ることにより，種々のプロセスにおけるデータのライフサイクル等を知り，企業のデータ資源の制御に役立てる
- データベースシステムの変更時にDDSを用いて効率良く処理する
- 種々の変更の影響の分析に用いることができる
- 応用分野の分析に用いる
- 報告書作成に用いる

ソフトウェアの持つ運命として，標準化もデータベースでは大きな問題である。標準化は，成果の相互利用等多くの利点もある反面，自由な発展を阻止するという欠点もあり，その時期の設定は非常に重要であろう。

マイクロコンピュータや分散処理も，データベースと同様に現在非常に興味を集めている分野である。このため，マイクロコンピュータによるデータベースの実現や，実用的な分散データベースの開発も重要となっている。応用面として，特に興味を集めているのは，オフィスの自動化と医療情報処理であると思われる。

3．データベース理論の話題

すでに述べたように，データベース本来の目的を達成するために解決すべき問題はかなりあり，関係モデルによって扱われることが多い。ここでは，理論的に扱われる問題のうち特に重要と考えられるものに絞って紹介する。

データベース設計論：Coddは関数従属性を用いたデータベースの正規化という概念を定義し，更新操作に適した冗長度の少ないデータベースの設計法を示した。最近では，従属性も，多値従属性，階層従属性，潜在多値従属性，相互従属性，一般化相互従属性（Mendelzon, Maier, Oct. 1979），結合従属性等の概念に拡張され，データベース設計の基準も，より高度の正規形（Fagin, May 1979）や独立成分という概念に一般化された。これらの概念を用いた設計法や，各種従属性集合から導かれる従属性集合を求めるための計算法（Parker, Delobel, Oct. 1979 や Maier, Mendelzon, Sagiv, Dec. 1979），従属性導出のための公理系，与えられた従属性が導びかれるかどうかを判定するための計算量の評価等の研究がある。関数従属性の最小カバーに対して多項式時間のアルゴリズムが得られた（Maier, April 1979）が，ほとんどのキーに関する問題は非常に難しい（NP完全）であることが知られている。

意味論：関係モデルの意味の扱いの機能を拡張しようという試みは種々知られ

ている。Codd はＡＣＭＴＯＤＳの７９年12月号の論文でこの観点からの関係モデルの拡張について論じている。空値の扱いについてのみでも，７９年にVassilion（SIGMOD），Lipski（ACMTODS），Lien（VLDB）等の報告がある。従属性と意味との対応については我々の報告（COMPSAC）がある。

データベース言語：データベース言語の完全性は，関係論理に基づく言語と同じ表現力を持つことと定義されており，この定義に疑問を投げかけたのは，Bancilhon（Sept. 1978）であった。完全性の新しい定義は，Aho, Ullman（Jan. 1979）や Chandra，Harel（April, 1979）によって扱われている。７９年のＳＩＧＭＯＤでは，Backus の Functional Style をデータベース言語に取り入れたFQL（Functional Query Language，Buneman，Frankel）やDAPLEL（Shipman）が発表された。

質問処理：Conjective query といわれる形の質問に対しては必ず同形を除いて唯一の最簡形が存在することがChandra，Merlin らによって示されており，Aho らはその結果に関数従属性や多値従属性を加えて考察し，Tableau という概念を導入した。Tableau は関係代数表現の等価性判定に有効であるため，これを用いた論文が Aho，Ullman らのグループから数多く発表されている（SIAM J. Computing May, 79, VLDB Sept. 78, ACMTODS Dec. 79 等）。質問処理の効率向上のためには，等価性から導びかれる最簡表現が重要である。INGRES で採用されている質問分解法は異った立場からの最適化手法であるといえる。もう一つの質問処理の問題は，より一般的なビュー操作で，NCC，COMPSAC 等に論文が発表されている。

並行処理：多数の利用者の質問を同時に処理する場合，デッドロックを避けるだけではなく，結果として誤ったデータ操作にならないように注意しなければならない。このようなことの起らない十分条件として直列可能性という条件が知られており，それを実現する単純な方法として二相ロックという方法が知られている。最近の研究としては，その条件の一般化やロック管理の計算量，さらに分散データベースへの拡張等がある。

安全性：計算機システムの安全性の確保は，計算機犯罪の防止やプライバシーの保護といった面で，単にデータベースに限らず重要な問題である。利用者に対して許す質問の制約が不適当であると，いくつかの質問を合成することにより許されていないデータの内容を知ることのできる場合がある。これは統計データベースの問題として知られていて，１９７９年には ACM TODS，JACM，Information Processing Letters 等に論文が発表されている。暗号を用いるのも有効な方法で，１９７９年に米国ＮＢＳの決めた The Data Encryption Standard の採用が実際的であるが，１９７６年に Diffie と Hellman の提案した公開キー暗号方式に関する研究も多い。多数決的方法による安全性の確保を行なうための多項式の内挿を使う方法も提案されている（Shamir，CACM，Nov. 1979）。

利用権の記証：データベースの利用権については，データベース管理者が全権を持つ３レベルシステムやデータベースの作成者が利用者の決定や所有権の移転できる２レベルシステム，さらに一般化した多レベルシステムが知られている。System R で採用した方式は中央の管理がなく，利用権や利用権を与えることのできる権利の移転のできる一般的な方式である。

保全性：保全性はデータの上に課せられた制約で，静的な制約のほか，値の変

更時に変更前と後の値の間の関係を規定する動的な制約もある。QBEでは，これらの制約の表現が容易である。INGRESでは質問の変更という形で保全性の実現を行っているが，データの変更の途中での検査は行わないようにしなければならない。保全性の一般化とその実現の手段としてはEswarmの提唱したtrigerがある。

ファイル構成：1979年のSIGMODでは，B-treeに関する論文が2件発表された。また，部分マッチングや各種操作に適したファイル構成の研究も重要であろう。ファイル構成の形式的表現も，概念レベルとのインタフェースの問題，最適化の問題等を扱うため重要な研究テーマである。

4．データベースと計算機科学の他の分野

データベースと計算機科学の他の分野は，次のような面により関係している。

(1) 他の分野ですでに開発された手法をデータベース側が利用する（アルゴリズム解析，暗号等）。

(2) データベースが他の分野と結びついて新しい研究テーマを生み出している（データベースマシン，知識ベース等）。

(3) データベースを利用する研究分野も多い（図形処理，CAD等）。

データベースの問題は，計算に関係した問題より単純であるため，後者の分野で解けなかった問題がデータベースの分野で解けるようになる可能性もある。その結果はまた後者の分野の発展に役立つと考えられる。データベースと関係の深い分野について，その双方に関係した問題を中心にそれらの関係を以下にまとめてみる。

ソフトウェア：オペレーティングシステムとデータベースの間のインタフェースの問題．データベース言語の研究（最近非手続き的プログラミング言語の研究がさかんであるがデータベース言語は一般に非手続き的である．Functional Style のものも提案されている）。ソフトウェア工学の手法の利用（要求定義，設計ツール）。発展形データベース（利用形態の変化に非常に柔軟であるもの）。

情報検索：データベースと情報検索はともにデータを蓄積して検索するためのものであるが，2つの分野で異なる制約のもとで研究が進められてきた。主な違いは次の通りである。データベースでは，組識化されたデータを対象に，検索したいデータ集合を正確に表現した質問を用いることができ，データの更新は実時間で行なえ，ビューの実現や独立性の達成を目標にしている．情報検索では，不完全なデータを対象に，自由度の少ない質問用言語を用いて，自分でもはっきりしない検索対象に対する質問を構成するもので，データ量が非常に多いことがあるのでクラスター化等が行なわれるが，検索が主体で，実時間の更新，ビューの実現，独立性の達成等は考えられていない。情報検索は歴史が古いので自然言語によるもの等種々の試みがすでにあるが，データベースでは同じ問題を扱っていてもその間の進歩による手法の違いがあるようである．汎用データベースシステムによる情報検索システムの実現等相互にも強く関係しており，両者の利点を生かした一般的なシステムの研究も望まれている。

人工知能：Codd の提案した言語は才一階述語論理に対応しており，その上に自然言語によるデータベースシステムを作ろうという試みがある。Codd の RENDEZVOUS は Chang の DEDUCE に変換する形で実現されている。

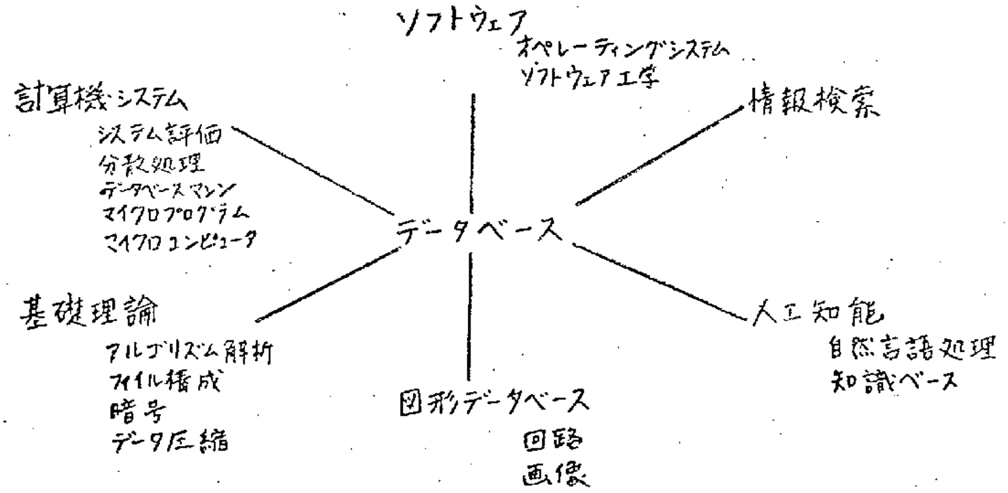
TORUS (Toronto Understanding System) はトロント大学で開発されたものである．イリノイ大で開発したPLANESは自然言語を関係論理へと変換するものである．人工知能の分野で重要な質問応答システムも強力なデータベースのささえを必要とする．

計算機の基礎理論：アルゴリズム解析，ソーティング，ファイル構成，暗号，データ圧縮等の分野がデータベースと直接関係している。

計算機システム：システム評価，分散処理，データベースマシンが関係の非常に深い分野となっている．分散データベースに関しては，並行処理，質問処理，利用権の認証，保全性の確保等に対する分散処理の方法が主要な研究テーマとなっている．分散データベースシステムの開発は，INGRES と Computer Corporation of America の SDD-1 (System for Distributed Databases)が有名である。データベースマシンとしては，RAP(Relational Associative Processor，トロント大学)，CASSM (Context Addressed Segment Sequential Memory，フロリダ大学)，RARES (Rotating Associative Relational Store，ユタ大学)等が良く知られており，日本では，電総研，北大，横浜国大で開発中である．これらのシステムでは，回転性の2次記憶の各部分に対してプロセッサが付けられているが，普通の計算機を後置データベース処理専用機としたものもデータベースマシンと呼ばれている。磁気バブルやCCDの低価格化に伴って前者のタイプが有望になるであろうと考えられている。

その他：図形を対象とするデータベースでは，データの特色の記述や近さの記述が重要である。パーデュウ大学のFuのところでは，Query-by-Pictorial-Exampleの開発を行っている。１９８０年　８月にはWorkshop on Picture Data Description and Management (第２回)があり，２月にはWorkshop in Design Automation Databasesがある等データベース応用方面でのWorkshopも増えつつある。

下図はこれらの分野の関係をまとめたものである．



5．アメリカおよびカナダのいくつかの大学および研究所の現状

アメリカ，カナダの大学や研究所をいくつか訪問したのでそれらの現状を報告する。

<u>ハーバード大学</u>：トロント大学出身のBernsteinが中心となって，関係データベースの研究をすすめている。現在の興味は，データベースの意味論，並行処理および分散データベースで，分散データベースについてはComputer Corp. of America（CCA）と協同研究である。分散データベースでは，並行処理，質問処理，信頼性についての研究を行っている。CCAには，Rothnie，Shipman，Smith夫妻のほか，サバティカル中のオハイオ州立大のD.K. Hisiao等が属しており，データベース研究の中心となっている。

<u>プリンストン大学</u>：8月までここにいたUllmanとベル研究所のAhoを中心に，Beeri（イスラエル），Sagiv（イリノイ大），Mendelzonらにより，データベース理論に関してかなりの数の論文が発表されている。Ullmanは最近プリントトン大学での講義に基づいてPrinciples of Database Systemsという本を著している。なお9月よりUllmanはスタンフォード大学の教授となっている。

<u>イリノイ大学</u>：上記のSagivが助教授として，データベース基礎理論の研究も行っている。Waltzの作成したPLANESはaugumented transition networkを利用した自然語による関係データベースシステムである。

<u>カリフォルニア大学バークレー分校</u>：Stonebrakerらの開発したINGRESは，SYSTEM Rと並んで最も機能の高い関係データベースシステムで，世界各地に移植されている。現在の中心テーマはネットワーク化したINGRESである。実際には，PDP 11/70で3ヶ所（Berbeley，Paris，Tahich）に分散したシステムをシミュレートする形式で実現されている。PDP 11/70には，500キロバイトの主記憶と300メガバイトのディスク，1メガバイトのディスクが付けられている。ミニコンによるものであるため，質問処理は5段階（利用者プログラム，パーザー，1変数質問処理機構，質問分解機構，データベースユーティリティ）よりなっており，質問変更による保全性やビューの実現，質問分解による最適化等非常に特色のあるアルゴリズムを採用している。主記憶250キロバイトで100メガのディスクが2台つながっているVAX 11も入っておりやがてそれに置き換えられる予定である。Z 8000でUNIXが使えるようにして，INGRESをマイクロコンピュータで実現することも考慮中とのことであった。

<u>ベル研究所</u>：Murray HillとHolmdelにデータベース研究者がいる。Murray Hillでは，Ahoがデータベース基礎理論のほかシステムの開発も行なっている。システムの言語はQと呼ばれUNIX上で実現されている。ホテルレストランデータベース等のデモンストレーションができる。

<u>IBMサンホセ研究所</u>：SySTEM Rの開発グループは昨年分散処理とメインテナンスのグループに分解し，R. Taylorはワトソン研究所に移った。Faginは計算機基礎理論のグループのマネージャとなっていたが，Schkolonickに変ったそうである。組織がえでCoddも動いたと言う話である。なおCoddはデータベースの本を執筆中である。

<u>トロント大学</u>：かつてはデータベース研究の中心的な存在であったが，現在ではTsichritzisとLochovskyの二人が専業でいるだけである。理論のほかに，フロッピーを4台付けたLSI-11でデータベースシステムとオフィスオートメーション

用のシステムを開発していた。フロッピーの一台にUNIXを入れ他の1台にデータも入れており，処理速度は遅いが経済的で実用的なシステムといえる。残念ながら，かつて開発されたZETA（その上でTORUSが動く）やRAPはなくなっていた。RAP開発者のS. Shusterは，トロント大学からまずIntelに移ったが，IntelはMRIの方式を採用することになりRAPの開発をしないことを決めたため，さらにTandem Computer社に移っている。彼の予定では，1980年中に，1セル当りの容量を1メガバイトとし，セル数を10～100個として，ディスクに比べた速度は10～1000倍になるものを開発する予定である。文字は1～256の間の任意の長さ，数字は16，32，64ビットのものが扱えるようになり，各種データモデルのサポートについても考察するとのことであった。

・カナダの他の大学：モントリオール大学では関数従属性の公理系を考え出したArmstrongがおり，最近ではフランスのDelobelと協同で成果をあげている。マギル大学のMerrettは，データベース言語やファイル関係の論文をすでに発表しており，教育用のシステムも開発している。1980年の巨大データベース国際会議は，Armstrongを議長として，モントリオールで開かれる。Waterloo大学では，電気工学と計算機科学の両方からサポートされた通信関係のグループの他に，Kamedaが分散処理を，Tompaがデータベース設計論の研究を行っている。University of West Ontarioでは，Osbornがデータベース理論の研究を行っている。

謝辞　本報告は，著者の79年5月～6月のアメリカ・カナダ出張および9～12月のマギル大学滞在中に得られた資料等に基づくもので，これらの渡航に際しお世話になった京都大学工学部矢島脩三教授ならびにマギル大学計算機科学科長Newborn教授に深謝いたします。

マギル大学に滞在して

上林弥彦

1979年の秋学期（9～12月）に，カナダのケベック州モントリオール市のマギル大学計算機科学科にVisting Professorとして滞在し，大学院の講義を担当しましたので，その間に感じたことを報告します。

マギルは，1829年に設立された医学部より始まった歴史のある大学で，百年以上もたった美しいヨーロッパ風の建築がまだ多く残っています。モントリオールの名のおこりであるMt. Royalのふもとにあり，大学自体は静かな環境ですが，2ブロック先の地下鉄のマギル駅の近くは繁華街となっていて，百貨店が4つもあるモントリオールの中心地です。大学の現在の規模は，先生の数が3,000人位で，学生数は16,000人位といったところです。

計算機学科は工学部に属しており，大学院学生は工学部ですが，学部学生は理学部です。より具体的には，工学部は5つのdepartment（化学，土木，電気，機会，冶金）と2つのschool（建築計算機）から成り立っていて，2つのschoolは他のdepartmentとかなり性格が異っています。たとえば，あとで述べるようにカナダの大学は3年制ですが，建築だけは4年制となっています。どこの大学でも，計算機科学をどこに位置付けるかはかなりの問題で，電気工学と一体のもの，工学部のもの，理学部のもの，人文系のもの等があり，電気工学と一体でないものに関しては，それらの間の競合の問題があるようです。マギルの場合は，計算機科学専攻と数学・計算機科学専攻のコースの他に，他学部の学生にも開放された副専攻のコースがあります。電気工学との協力関係についてはあまりうまくいっているとは言えません。

計算機科学科の講師以上のスタッフは横の表に示すように18人で，前年度と総数は同じですが，2人の入れ替りがあります。学科長のM. Newbornは，もともとオートマトン理論をやっていた人で，現在でも組合せ論に興味を持っています。現在の興味の中心は，人工知能とその応用としての計算機チェスで，計算機チェスに関する著書が2冊あります。学会関係では，ACMの北米の北東地区代表やACM計算機チェス委員長等として活動しています。計算機チェスは，人工知能の個別的応用としては非常に成功した例で，現在最も強いプログラムに勝てる人は世界中で100人余りになっただろうと言われています。ACMの年次大会では，1970年以来ずっと計算機チェスの試合を続けており，次回のIFIPでも前回同様計算機チェスをやる予定でしたが，日本側が全く興味を示さなかったので，東京ではやらないことになり，非常に残念がっていました。学科長の好みもあるのか，組合せ論，オートマトン理論に近い人が多いのがこの学科の特色です。表でも判るように，ほとんどの先生が30代で，カナダ人の他，イギリス人，アメリカ人，ベルギー人，チェコスロバキア人，オーストラリア人，インド人等国籍もさまざまです。ほとんどの人が，卒業大学と博士号を取った大学が違うというのも日本と異なる点です。

K. ANANTHA, Ph.D., Madras, 1965; Operating systems, modelling and simulation, minicomputer software.

D. AVIS, Ph.D., Stanford, 1977; Computational complexity, analysis of algorithms.

D.J. BURRAGE, M.Sc., Brunel; 1969; Numerical analysis, database applications.

V. CHVATAL, Ph.D., Wat., 1970; operations research, combinatorics, graph theory.

L. DEVROYE, Ph.D., Texas, 1977; Statistical estimation theory, pattern recognition.

N. FRIEDMAN, Ph.D., Toronto, 1977; Automata theory, computational complexity.

A. GREENBERG, M.Sc., McGill, 1973; Computer architecture, systems analysis.

R.N. HORSPOOL, Ph.D., Toronto, 1976; Operating systems, programming languages, compilers.

Y. KAMBAYASHI, Ph.D., Kyoto, 1970; database theory, automata theory.

G. KLINCSEK, Ph.D., McGill; 1973; operations research, theory of computation.

T.H. MERRETT, D.Phil., Oxford, 1968; Data bases, information retrieval, modelling and simulation.

M.M. NEWBORN, Ph.D., Ohio State, 1967 (Director); Artificial intelligence, data structures, automata theory.

C.C. PAIGE, Ph.D., London, 1971; Numerical analysis, optimization methods, computational linear algebra.

G.F.G. RATZER, M.Sc., McGill, 1966; Minicomputers, air-traffic control systems, medical information systems.

D. THERIEN, Ph.D., Wat., 1979; automata theory, theory of computation.

W.D. THORPE, M.A., Cambridge, 1955; Director of Computing Centre; Time-sharing systems, allocation algorithms, operating systems.

G.T. TOUSSAINT, Ph.D., U.B.C., 1972; Statistical decision theory, pattern recognition, information theory.

P. WYNN, Ph.D., London, 1960; Numerical analysis, applied mathematics.

（学科長も許しませんし，学生からも文句が出ます。フランス系の大学では休講をすると学生が喜ぶそうですが）。カナダの場合，9～12月と1～4月は，先生は講義に追われることになりますが，5～8月に自由で，どこで何をしても良いということになっていて，これが研究の時期です。学生はこの間に学資かせぎをしたりするようです。昨年の5～8月は，たとえば Chvatal は，日本とフランスですごし，Devroye はスタンフォード大に無給で1ヵ月居たあとテキサス大ですごしていました。

大学院入学は書類選考のため，学生の質のバラツキは日本より大きいようで，途中であきらめざるを得ない学生もかなりいるようです。講義を始める前の週は，講義内容や，自分の過去に取った科目や過去の仕事を説明して，この講義が理解できるかどうかを聞きにくる学生が何人かいました。そこで，第1日目は，講義のスケジュール，宿題の程度と頻度，試験の予定と宿題の配点等についても説明しました。修士レベルの入門コースでは，宿題は毎週，試験は2回としたのですが，宿題や試験を返すたびに，自分の意図はこうだったとか，自分は外国人なので英語の長文は苦手であるからとか，ここは点の引きすぎではないかとか，さらには，自分は能力があるが試験に遅れてきたのでこの結果になったとか，この日は熱があった，妻が病気で等々個人的事情まで言う人がいました。この時気付いたのですが，アジア人の学生は，下手な英語をしゃべるのに答案の英語は立派で逆に，南米の学生は，英語の発音は良いのに書けないという傾向があります。1回の講義中に，4～5回は質問がある点も，教室内ではおとなしくなる日本の学生と違う点です。入門コースの方は30人位の学生が居て22人が単位を取り，advanced の方は12人（うち2人は先生）が聞いていましたが単位を取ったのは3人でした。先生は計算機学科の Burrage とマネジメント学部の Howson（大学のソフトウェア受け入れの責任者）がいましたが，数学の話になってきた頃から Howson が脱落しました。成績は，受講者名の計算機出力に成績を書き込んで，私の室のドアに張って発表しました。

計算機センターの計算機は，最近，アムダール社の470－V7になりました。日本との違いは終夜運転をしていること，計算機をよく使う先生の室や自宅に端末のあること位です。一科目につき，学生1人あたり200ドル分の計算機利用権をセンターから割当ててもらうようになっており使いきった学生には追加して最大600ドルまで使えるようになっています。学部学生のプログラム入門コースは，数百人の学生がいるので，先生1人と数人のTA（Teaching Assistant，大学院生）でやっていますが大変なようです。計算機以外を専攻とする学生にはFORTRANが重要なので，構造的FORTRANを教えていますが，計算機専攻の学生だけはPASCALに変えることが教室会議で議論されていました。

学部学生の成績は，A（100－80％），B（79－65％），C（64－55％），D（54－45％），E（44－0％）で表わされ，それぞれを，4，3，2，1，0に換算して，単位数で重みづけて平均したものをGPA（a grade point average）と呼びます。大学院ではAとB以外は単位になりません。ふつうの学生は平均2.0以上なら卒業できますが，優秀な学生のために，Honours Program が設けられており，そこではそれらの学生専用のレベルの高いコースも含めて平均3.2以上（それ以下になるとふつうの学生になる）が要求されています。計算機学科では，この9月より始めたため，今年は2人しかいませんが，良い学生を集めるためだと学科長は言っていました。Honours Program で卒業したというのは非常に名誉なことだそうです。なお，今年の広告には，京大の顔写真の輪郭化の出力が使われていました。学部学生でも認められれば大学院のコースを取ることができる点も異っています。

修士号を取るには6科目と論文という方法と，9科目と3つのプロジェクトという2つの方法があります。1つのプロジェクトの仕事量は1科目とほぼ同じ位です。本年から夏学期もプロジェクトができるようになったので，1年で修士を取ることも可能になりました。

博士号を取るには，3年（修士号を持っていれば2年）は必要で，博士の予備試験（comprehensive examination）に通ってから博士論文を作成することになります。予備試験は3月と10月の年2回

ありますが，合格率はかなり低いようです。範囲は次のとおりです。

Ⅰ群	ファイル構成とデータベース
Ⅱ群	情報構造
Ⅲ群	プログラミング言語とコンパイラ
Ⅳ群	形式言語
Ⅴ群	数値解析
Ⅵ群	パターン認識
Ⅶ群	(1)組合せアルゴリズム　(2)OR　(3)確率・統計における計算法
Ⅷ群	計算機システムと構成

学生はこのうちから5つの群を選び，Ⅶ群を選んだ場合はそのうちの1つを選び解答することになっています。

データベースは，今人気のあるトピックで，来年予備試験を受ける学生（今のところ5人）のうち4人がMerrettの所で研究することを希望していました。講義は，2つ併行に行なわれることが多いので，マイクロコンピュータやデータベースに併行して同じ時間に行なわれる他の講義には学生がさっぱり来ないそうです。

カナダの科学研究費は，アメリカのように少数の人に多額の研究費を割当てるのではなく，ほとんどの人に平等にあたるというふうになっています。割当ての委員も現職のその分野の教授が交替であたり，政府は全く関与していません。アメリカでは，専任の委員が配分を決めています。9月の初めに，計算機関係の委員であるE.W. Elcock と W.M. Gentleman が来て，科研費をもらっている先生に20分ぐらいづつ個別に話をし，大学を見てゆきました。その時のスケジュールは，まず，工学部長と学科長が会い，次に研究費をもらっている先生に個別に会い，それらの先生と昼食を共にし，午後は残った先生と会い，学科の見学，最後にまた工学部長と学科長に会い5時に帰るというものでした。食事を一諸にしたのですが，計算機科学の最近のトピックや，アメリカに対抗するためにはカナダはどうすべきか等ということが主な話題でした。アメリカは，スタンフォードのAIのように拠点を作るが，カナダは分散しているのでそのことの利点欠点等，日本にもあてはまりそうな話もありました。研究費の支出には昨年までは旅費は10%までという制限があったのですが，今年はそれもなくなり用途は完全に自由です。主な出費は，研究費の他，人に講演を依頼した場合の旅費・謝礼金やレストランの支払い，海外出張費等です。

マギル滞在中に，工学部長主催の行事に3度出席しました。9月7日に，工学部長主催の「ビールとピザ」のパーティーがあるというので，これで昼食代が助かったなどと言いながら学科の先生と総勢5名で行ったところが新入生歓迎パーティーでした。昔は，おごそかな入学式があったそうですが，今は学生と先生が一諸にビールを飲んでピザを食べながら話をしようという形式になっていました。室に入ると，超大型のステレオに専門家らしいのが1人付いていて，ロック音楽をガンガン演奏していて，まるで学生用のバーにでも行った感じで話も満足にできません。学生の中にはいろいろと工夫してムチャクチャな格好をしてきているのがいて，ヘルメットの上にサクション（トイレがつまった時に使うもの）を付けたものを着用している人までいました。やがて，工学部長が入ってきて，皆に握手してまわっていました。私は，この時学科長にはじめて工学部長を紹介してもらいました。とにかく，うるさいのでピザを一片食べただけで全員引き上げました。9月11日には，工学部教授会で新人の紹介があるというので出かけてゆきました。はじめから，30%位しか出席しないことを予想した数のイスしかありませんでしたが，イスがすこし余る程度の人数しか出席していませんでした。入口でケーキとコーヒーを取って自分の席にすわって，工学部長の挨拶を聞きながら食べていると，今年は5人新人が居ると言って紹介してくれました。最後は，12月21日の工学部長主催のクリスマスパーティーで，室を暗くしてローソクをともし，パンチとつまみが出ていました。

学科でも新入生歓迎パーティーとクリスマスパーティーをやりました。クリスマスパーティーに

は，ユダヤ人の先生や学生の中にはでてこない人もいて，宗教問題のきびしさを感じました。

カナダの大学は3年で卒業できるのが普通ですが，ケベック州では，日本の高校3年と大学1年にあたる2年間はCEGEPという教養課程に相当するところへ行きますので，日本と卒業は同じ位になります。他の州では一年早く卒業できるところもあるようです。

修士コースの約半数，博士コースの全員が外国人となっていました。ケベック州の資金で，半数以上は外国人である先生が，半数位は外国人の学生を教えるということになっています。優秀な外国人はカナダに残って教育の成果を還元する可能性があるからできることで，日本のように外国人に閉ざされた社会では，外国人学生をこのように増やすことは不可能だと思われます。このように外国人や社会人（夜間もある。計算機学科でも検討中）に開かれた大学であるということは羨ましく思います。中近東やアフリカの大学からここへ2年教えにくる人はいませんかという手紙がくるのも英語を使う強みでしょう。しかし，逆に考えると，この方式でアメリカやカナダは他国から人材を吸いあげてきており，資源が人材にかわった新しい形の植民地主義ではないかという気もして，留学生を必ず国へ帰す日本のやり方の方が正しいのかもしれないと思ったりもしています。

外国人が多いので，奨学金の面でもカナダ人優遇がみられ，カナダ人だと6～8000ドル位の奨学金がもらえますが，外国人は年800ドルからということで、外国人は生活に追われながらがんばっているか，そうでなければ，どこかの企業から派遣されている人や自国の奨学金をもらっている人がふつうのようです。また，外国人といっても英国の共栄圏に属す香港やインドの人が圧倒的です。モントリオールの街は，100年くらいたった建物が，アメリカのようにスラム化しないで残っているので，そういう所に下宿すれば月100ドル位で済むそうで，食物もスーパーの値段は日本の半額以下なので，日本の下宿生活と比べて生活費は変らないか安い位につくようです。

カナダは，イギリスの影響でアメリカと異っている点がいくつかあります。たとえば，学生の先生に対する態度が丁寧であること，単位をとらないのに講義だけを聞くsittingの学生がかなりいること，成績の良い学生のためにHonours Programが用意されている点などです。イギリスではHonourにもランクがいくつかついているものもあるそうですが，フランス系の大学では採用していません。

図書室は，ふつう9時頃まで開いているのが便利な点です。貸出し期間が過ぎると，それに応じた罰金をとられるというのも厳しいところです。雑誌は，ゼロックスをとる時間しか貸してくれませんので、学生のために5セントで1枚とれるゼロックスの機械が置かれています。本を借りる手続きの時に本を机の下に置くので何をしているのかと思っていたのですが，一度係りの人がそれを忘れて、私がその本を持って外へ出ようとすると，出口のブザーが鳴りました。磁気的な装置で貸出し手続きの終った本しか持ち出せないようにしているようです。

最後に，モントリオールの街について簡単に書いておきます。ケベック州にあるため，アメリカイギリス，フランスの3つの国の影響があるという面白い街です。建物やレストラン等はヨーロッパ的で，特にフランス人はアメリカ人より生活を楽しむことを重要だと考えているようです。非常に住み易いところで，北米のパリとかカナダのニューヨークとか言われる文化都市です。人種差別もなく，カンボジアの孤児達が，一般家庭に養子として多数迎えられていました。70年代の初めから始まった女性解放等の弱者の権利闘争は、ここではフランス系カナダ人による支配という形で現われていましたが，79年の″反動の時代″に対応して「法令101（フランス語優先法）」の憲法違反判決，ケベック党の支持率低下等，ケベック州の独立は不可能となってきたようです。ストライキもイギリスに次いで多く，滞在中毎日何かのストがあった位です。モントリオール大学等のフランス語系の大学では教授のストがあり，教育も熱心でないので，勉強したい人はマギルに来るためマギルのフランス系カナダ人が20%を越えたようです。アメリカの大都会と違う点は，治安が良いという点で，真夜中に街を歩いても危険なことはありません（但し男性）。とはいってもモントリオールで1979年の1月～9月の内，殺人55件，泥棒25,148件，ホールドアップ（主に銀行強盗）4,839件，暴行116件，届出のあったrape204件というのは日本に比べ

ると大分多いようで，そのうち検挙率は1/7という悪さです。冬はマイナス40°Cにも達する寒さで，そのため大学でも建物と建物を結ぶ地下道がついています。その他の点では，アメリカよりのんびりしている点が目につきます。しかし、いろいろな人種のいる大都会で欠点がこのくらいというのは，いい方だと言えるでしょう。京大の学生なら，マギルの大学院でやってゆけると思いますので，英語の勉強もしながら，あわよくば1年で修士を取ってやろうという人があれば，事情を聞きに来て下さい。

INFORMATION SPACE MODEL

Yuzuru Tanaka

Graduate School of Information Engineering
Hokkaido University
Sapporo, 060 Japan

The importance of the semantic theories of databases has come to be recognized in various problems of databases. This paper proposes an infosemantic framework based on the relational model that has been an infological framework of database theories. In a relational database, semantic relationships are partly embedded in attribute names and partly embodied by intrarelational semantics. However, most of them are hidden in interrelational relationships that are not explicitly specified by a relational schema. There are two kinds of interrelational structures, analytic ones and synthetic ones. Among them, the synthetic structures play especially important roles in semantic problems. However, there are no proper theoretical basis to deal with these structures. The information space model (R, M) gives an infosemantic framework of this problem, where synthetic structures are formalized as morphisms between relations. The paper gives detail formalization of this model and examples of its applications.

1. INTRODUCTION

Recent studies on databases indicates the necessity of a new theory about formal semantics of databases. Although the lack of formal semantics of databases has come to be noticed through the studies on relational databases, it is not only a problem of this special model but also a more general problem involving all kinds of data models. This problem unfortunately attracted very little attention before because of its difficulties.

Various approaches are possible to cope with this problem, however, we will choose the relational model as the basis of our approach to a new semantic model since the relational model has contributed a lot for these ten years to the development of database theories and we should not neglect this fact.

In a relational database, semantics of information is partly embedded in the names of attributes and partly in the intrarelational relatioships. However, most part of the semantics is hidden in the interrelational relationships that are not explicitly defined as a part of schema description. It may sound reasonable that someone says the interrelational semantics is described by relation names. However, this is the most common misunderstanding of relational semantics. While we can infer the semantic relationship between two relations from their names, it is absolutely impossible for the computer system that manages these relations to do the same inference.

To cope with the interrelational semantics of relational databases, we propose the information space model (R, M_0), where R is a set of first normal form relations of an object database, and M_0 is a set of elementary morphisms among elements in R. An interrelational relationship is described by a morphism between relations that is either elementary or derived from elementary ones by composition. Since the number of elementary morphisms is proved to be always finite, there always exists a finite description (R, M_0) for any database.

The elementariness of a morphism is clearly defined. For each morphism σ in M_0, we define a label $l(\sigma)$. A set $L(M_0)$ denotes $\{l(\sigma) | \sigma \in M_0\}$. A semantic attribute is a concatenation of an attribute and a list of labels, i.e., $Al(\sigma_1)l(\sigma_2) \ldots l(\sigma_n)$. Let $\omega(\sigma)$ denote the semantics of σ that may be informally interpreted as an English noun representing the relationship denoted by σ. Then the semantic attribute $Al(\sigma_1)l(\sigma_2) \ldots l(\sigma_n)$ may be informally interpreted as an English noun phrase A $\text{prep}_1\ \omega(\sigma_1)\ \text{prep}_2\ \omega(\sigma_2) \ldots \text{prep}_n\ \omega(\sigma_n)$, where prep_i is one of the following prepositions; 'of', 'in', 'at', 'on', and 'by' etc.

For an information space schema (R, M_0), a set M_0 generates a set of semantic attributes $\{A\rho | \rho$ is a finite sequence of elements in $L(M_0)$, i.e., $\rho \in L^*(M_0)\}$. This set is denoted by Ω^*, where Ω denotes a set of all the attributes appearing in some relation in R. For each σ in M_0, we can define a morphism $\hat{}\sigma$ between semantic attributes such that

$$\hat{}\sigma : A\rho \longmapsto Al(\sigma)\rho \quad \text{for } A \in \Omega \text{ and } \rho \in L^*(M_0).$$

A set of morphisms $\{\hat{}\sigma | \sigma \in M_0\}$ is denoted by $\hat{}M_0$. While (R, M_0) is a finite category, the category $(R^*, \hat{}M_0)$ is infinite. It should be noticed here that an infinitely large space $(R^*, \hat{}M_0)$ can be defined by a finite description (R, M_0).

In the following sections, after informal introduction of information space model, its formal semantics is formalized. Recursive morphisms and their relationships to schema design are detailed. And finally, denotational semantics of query language vocabulary is explained.

2. INTERRELATIONAL SEMANTIC STRUCTURE

2.1. Analytic Structures and Synthetic Structures

Interrelational semantic structures of a relational database are classified into two categories, i.e., analytic structures and synthetic structures.

Fig.1 (a) shows an example relation in the first normal form. This can be decomposed into two relations shown in (b) because of the existence of a functional dependency /department/→/floor/. This decomposition process defines an interrelational semantic structure between R1 and R2 that reflects

R	employee	department	floor
	J. Smith	A	2
	K. Jones	A	2
	F. Brown	B	3
	:	:	:

(a) an original relation

R1	employee	department
	J. Smith	A
	K. Jones	A
	F. Brown	B
	:	:

R2	department	floor
	A	2
	B	3
	:	:

(b) two relations obtained by the decomposition of (a).

Fig. 1. An example of an analytic interrelational relationship.

the dependency structure they had in (a) before the decomposition. This kind of interrelational relationships is determined by the analysis of the intra-relational dependency structures of the original relation, and hence it is called an analytic structure. The original relation is a so called universal relation of R1 and R2.

However, we can not always assume the existence of a universal relation. Fig.2 (a) shows the instances of two relations for which there exists no universal relation. They are projections of a relation with a lot of null values (Fig.2 (b)). Fig.3 (a) shows an instance of a relation for which we can define a relation with infinitely many attributes (Fig.3 (b)). In these two examples, interrelational relationships are defined by something other than analytic dependency structures. Since their semantics is defined by the way of synthesizing an integrated view of information from original relations, we call such a structure a synthetic structure. Especially, the relationship in Fig.3 (a) is called a recursive synthetic structure. Recursive synthetic structures form a very interesting and important class of synthetic structures.

While there may exist more than one synthetic relationships between two relations, the analytic relationship between them is always unique if any. While analytic structures concern the decomposition of a first normal form universal relation, synthetic structures concern the overall semantic structures of a set of constituent first normal form relations. This paper deals with the synthetic structures. Our approach to analytic structures is detailed in [TANA77] and [TANA79].

2.2. Necessity of Denotational Interrelational Semantics

We show examples of three kinds of problems concerning the necessity of denotational description of interrelational semantics.

The first problem concerns the isomorphic relationship between a query language and a natural language. In Fig.4 (a), we show four example queries to a database in Fig.3 (a) written in both English and a SEQUEL like language [CHAM76]. While the representations of these queries in English are isomorphic, their representations in a SEQUEL like language have different forms. If we view this database as an infinite relation in Fig.3 (b) with an extended set of attributes then the representations of these queries in this query language become isomorphic as in Fig.4 (b). In these example queries, there appears two extended attributes, i.e., /name of the parent/ and /birth date of the parent/. They have the phrase "of the parent" in common. In English, these two appearances of "of the parent" have the same meaning. Obviously, the phrase "of the parent" is a kind of synthetic structures in this database. What is the formal semantics of "of the parent" in this database?

The second problem concerns the formal description of a subspace that is semantically meaningful in a real world of information. In the database in

novel	author
The adventure of Tom Sawyer	Mark Twain
Crime and Punishment	Feodor Dostoyevsky
For Whom the Bell Tolls	Ernest Hemingway
Gone with the Wind	Margaret Mitchell

character	novel
Porfiry	Crime and Punishment
Ishmael	Moby-Dick
Robert Jordan	For Whom the Bell Tolls
Philip Carey	Of Human Bondage
Pilar	For Whom the Bell Tolls

(a) two relations with a synthetic interrelational relationship between them

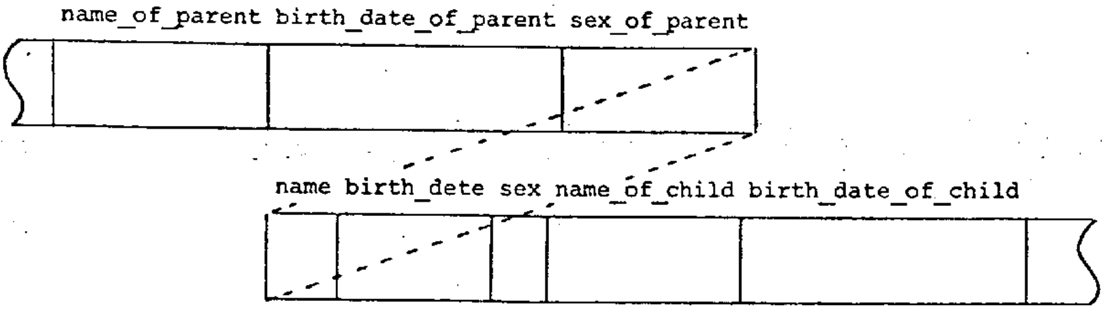
character	novel	author
	The adventure of Tom Sawyer	Mark Twain
Porfiry	Crime and Punishment	Feodor Dostoyevsky
Robert Jordan	For Whom the Bell Tolls	Ernest Hemingway
Pilar	For Whom the Bell Tolls	Ernest Hemingway
	Gone with the Wind	Magaret Mitchell
Ishmael	Moby-Dick	
Philip Carey	Of Human Bondage	

(b) an integrated view of two relations in (a)

Fig. 2. A synthetic interrelational relationship and an integrated view.

name	parent	birth date	sex
J. Smith	A. Smith	Dec. 11 1940	male
R. King	S. Brown	Mar. 20 1920	male
J. Smith	B. Wilson	Dec. 11 1940	male
P. Scott	L. Scott	May 9 1970	female
Y. Tanaka	K. Tanaka	Feb. 17 1950	male
A. Smith	T. Smith	Nov. 15 1915	male
B. Wilson	K. Wilson	Jun. 8 1918	female
H. King	R. King	Jul. 1 1950	male
⋮	⋮	⋮	⋮

(a) a relation with a recursive relationship.



(b) a view of (a) with infinitely many attributes.

Fig. 3. A recursive relationship and a view with infinitely many attributes.

(1) Find the name and the sex of a person whose birth date is Feb. 17 1950.

```
select   name, sex
where    birth date = 'Feb. 17 1950'.
```

(2) Find the name of the parent and the sex of a parson whose birth date is Feb. 17 1950.

```
select   parent, sex
where    birth date = 'Feb. 17 1950'.
```

(3) Find the name and sex of a person whose parent's birth date is Feb. 17 1950.

```
select   name, sex
where    parent  in
         select   name
         where    birth date = 'Feb. 17 1950'.
or
select   e1.name, e1.sex
where    e1.parent = e2.name
      and e2.birth date = 'Feb. 17 1950'.
```

(4) Find the name of the parent and the sex of a parson whose parent's birth date is Feb. 17 1950.

```
select   e2.name, e1.sex
where    e1.parent = e2. name
      and e2.birth date = 'Feb. 17 1950'.
```

(a) four queries written in English and a SEQUEL like language.

```
(1)   select   name, sex
      where    birth date = 'Feb. 17 1950'.

(2)   select   name of parent, sex
      where    birth date = 'Feb. 17 1950'.

(3)   select   name, sex
      where    birth date of parent = 'Feb. 17 1950'.

(4)   select   name of parent, sex
      where    birth date of parent = 'Feb. 17 1950'.
```

(b) queries based on the view in Fig.3 (b).

Fig. 4. Various queries of a database in Fig.3 (a) and those based on the view of this database shown in Fig.3 (b).

Fig.3 (a), the information about the antecedents of J. Smith forms a semantically meaningful subspace. It is very reasonable in some possible applications to restrict the access right of each user within information about his own antecedents. How can we formally specify this kind of subspaces? Relational model can not answer this question since every subspace describable by this model is a subpart of some single relation or a union of such subparts (Fig.5).

The third problem concerns the formal semantics of natural language vocabulary. If we can formally define the semantics of the phrase "of the parent" in Fig.3 (a), then we can also define the semantics of "of the father", "of the brother", "of the sister", etc. However, no single phrase of the latters defines the former. In this database, "of the parent" is an elementary synthetic relationship, while the others are derivable from this. It may be expected that we will be able to define formal semantics of various vocabularies from the semantics of elementary synthetic interrelational relationships.

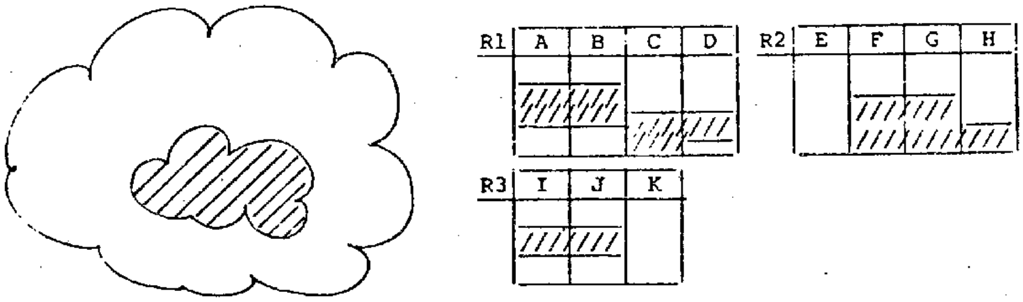
All these problems above concern synthetic structures among relations rather than analytic structures. They prove the importance of the formalization of synthetic interrelational relationships.

3.MORPHISM BETWEEN RELATIONS

3.1. Formal Interpretation of a Synthetic Interrelational Relationship.

Fig.6 shows an example of synthetic interrelational relationships. Suppose that R1 is a relation about the managemental information of an institute and R2 is a relation about bibliographic information for reference use in this institute. The relation R2 includes not only papers written by staffs in this institute but also those by authors outside of this institute. These two relations are related synthetically but not analytically. Integration of these two relations enables us to search papers written by a project in a focus. These are papers written by such authors who are staffs of this project. Such papers are "papers of the project". This adjective phrase "of the project" can be considered as a morphism, i.e., a relational morphism, that relates two relations R1 and R2, i.e., σ:R2 → R1. This morphism induces a mapping that maps information about documents to information about documents of the project. The latter might be considered as a part of the information of the project.

As is shown in Fig.7, a morphism σ:R2 → R1 relates two tuples p in R1 and d in R2 in such a way that /staff/-value of p is equal to /author/-value of d. It conceptually extend a tuple p to p' that is a concatenation of p and the image of d mapped by σ. An extended tuple p' represents information of the project in a focus. For an attribute A in R1, A-component of p' represents information about A of the project, while, for an attribute B in R2, B-component of p' represents information about B of a document of the project. Since obviously p' represents information of the project, we omit the last phrase "of the project" from the names of each component of p'.



(a) a subspace of an information space.

(b) subparts of relations.

Fig. 5. Difference between a subspace of an information space and a union of subrelations.

R1(/project/, /budget/, /staff/)

R2(/author/, /title/, /journal/)

Fig. 6. An example database with a synthetic interrelational relationship.

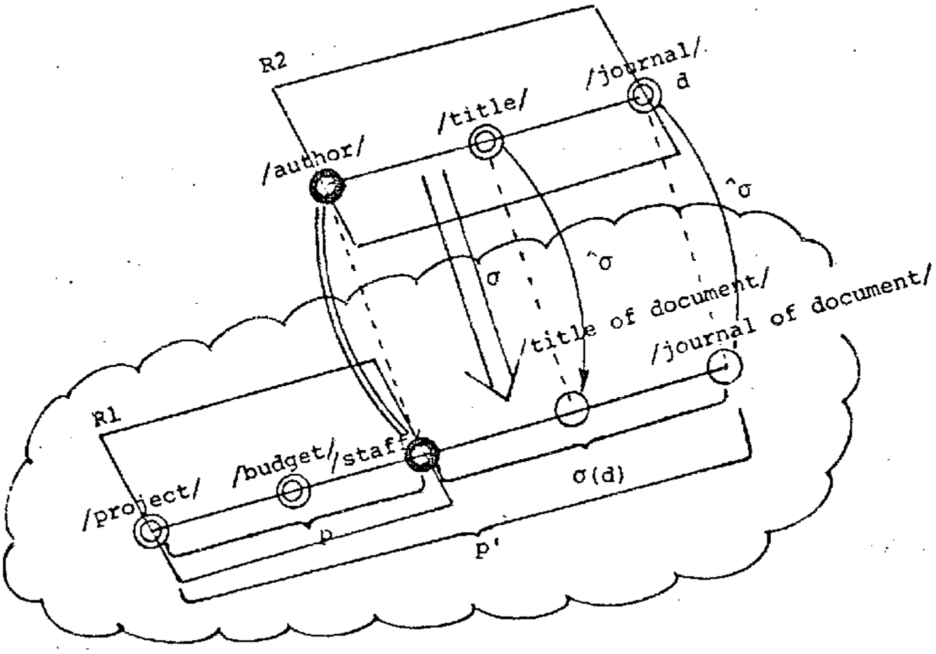


Fig. 7. Pictorial interpretation of an interrelational morphism.

Therefore, A and B-components of p' above can be denoted by 'A' and 'B of document' respectively. A set of such extended tuples as p' forms an information space represented by a cloud in Fig.7. Between each attribute B of R2 and its counterpart 'B of document' in this cloud, we can define a morphism $\hat{\sigma}$ such that $\hat{\sigma}$: B $\longmapsto$ B of document. This morphism is called a labeling morphism since it labels B with "of document". Labeling morphisms should be one-to-one. We can define the morphism σ in the above example in such a way as

<u>morphism</u> σ: R2 → R1
<u>where</u> $\hat{\sigma}$(/author/)=/staff/.

For each morphism σ, we can define its reverse σ^{-}. For σ above, σ^{-} is equal to the following definition;

<u>morphism</u> σ^{-}: R1 → R2
<u>where</u> $\hat{\sigma}^{-}$ (/staff/)=/author/.

3.2. Formal Theory of Morphisms and Semantic Attributes

In the sequel, we use the following notations;

(1) Pred(X), Pred_i(X) : a predicate about attributes in an attribute set X,
(2) [X]R : projection of a relation R to an attribute set X,
(3) [Pred(X)]R : restriction of a relation R with respect to the condition Pred(X)=true,
(4) RS : Cartesian product of two relations,
(5) $^{\cap}$RS, $^{\cup}$RS : intersection and union of two relations R and S with a same attribute set,
(6) ⊳⊲RS : natural join of R and S with respect to the common attributes,
(7) $\langle X\rangle_0$: a relation over an attribute set X,
(8) Ω(R) : the set of all the attributes of a relation R,
(9) |S| : the number of elements in a set S.

Let R denote all the first normal form relations of a database and M a set of morphisms in R. It is assumed that, for each morphism σ in M, a special morphism σ^{-} called a reverse of σ is also included in M, where $(\sigma^{-})^{-}$=σ. We assume that each relation in R has different attributes disjoint from those of the other relations in R. This condition is always satisfied after proper renaming of attributes. The world W is a Cartesian product of all the relations in R, i.e.,

$$W = \Pi_{R \in \mathcal{R}} R. \tag{3.1}$$

We denote this by $\langle\Omega\rangle_0$, where

$$\Omega = \cup_{R \in \mathcal{R}} \Omega(R). \tag{3.2}$$

Let $l(\sigma)$ denote a label corresponding to a morphism σ such that the labeling morphism $\hat{}\sigma$ mapps any attribute A in Ω to $A\ell(\sigma)$. We define semantic attributes as follows;

(1) attributes in Ω are semantic attributes,
(2) if A is a semantic attribute and σ is in M, then $A\ell(\sigma)$ is a semantic attribute,
(3) only those obtained by finite applications of the above two rules are semantic attributes.

We denote the set of finite sequences of labels by $L^*(M)$ and the set of semantic attributes by Ω^*, i.e.,

$$\Omega^* = \{A\rho \mid A \in \Omega \text{ and } \rho \in L^*(M)\}. \tag{3.3}$$

By $X\rho$, we denote a set $\{A\rho \mid A \in X\}$ for any $X \in \Omega^*$ and $\rho \in L^*(M)$.

A morphism σ between relation R and S is defined by a statement:

$$\begin{array}{ll} \underline{\text{morphism}} & \sigma: R \rightarrow S \\ \underline{\text{where}} & \text{Pred}(\hat{}\sigma(X),\ Y), \end{array} \tag{3.4}$$

where $X \subset \Omega(R)$, $Y \subset \Omega(S)$, and $\hat{}\sigma(X) = X\ell(\sigma)$. It is denoted by σRS that R and S are related by a morphism σ: $R \rightarrow S$. Relations R and S are called a domain relation and a codomain relation of σ.

The formal semantics of σ is defined by its natural extension $\hat{\partial}$: $2^{\Omega^*} \rightarrow 2^{\Omega^*}$ as

$$\sigma RS \equiv \hat{\partial}\Omega(R)\Omega(S). \tag{3.5}$$

Let $\alpha(\rho)$ denote a unary operator that renames every semantic attribute A in the immediately following term to $A\rho$. The natural extension $\hat{\partial}$ of σ is defined by a λ-expression:

$$\hat{\partial} = \lambda xy.\ \langle (x\ell(\sigma)) \cup y\rangle. \tag{3.6}$$

The relation $\langle x\rangle$ over an arbitrary subset x of Ω^* is recursively defined as follows;

$$\begin{array}{ll} (1) & \forall x \subset \Omega,\quad \langle x\rangle = \langle x\rangle_{\Omega}\ (=[x]\langle\Omega\rangle_0), \\ (2) & \forall x,y \subset \Omega^* \text{ s.t. } y \cap (\Omega^*\ell(\sigma)) = \phi, \\ & \qquad \langle (x\ell(\sigma)) \cup y\rangle \\ & \qquad = [(x\ell(\sigma)) \cup y][\text{Pred}(X\ell(\sigma),\ Y)](\alpha(\ell(\sigma))\langle x \cup X\rangle\langle y \cup Y\rangle), \\ (3) & \forall x \subset \Omega^*,\ \langle x\ell(\sigma)\rangle = \alpha(\ell(\sigma))\langle x\rangle. \end{array} \tag{3.7}$$

Let σ and τ be two morphisms in M defined as

$$\begin{array}{ll} \underline{\text{morphism}} & \sigma : R1 \rightarrow S1 \\ \underline{\text{where}} & \text{Pred}_1(\hat{}\sigma(X_1),\ Y_1) \\ \multicolumn{2}{l}{(\ X_1 \subset \Omega(R1),\ Y_1 \subset \Omega(S1)\ )} \end{array}$$

and

morphism $\tau : R2 \rightarrow S2$
where $\mathrm{Pred}_2(\hat{}\sigma(X_2), Y_2)$

$$(X_2 \subset \Omega(R2), Y_2 \subset \Omega(S2)).$$

The composite morphism $\tau\sigma$ is defined as

$$(\tau\sigma)(R1)(S2) \equiv (\widehat{\tau\sigma})\Omega(R1)\Omega(S2),$$

where $\ell(\tau\sigma)$ is defined to be $\ell(\tau)\ell(\sigma)$. Independently from the above definition, we define the composition of the natural extensions $\hat{\sigma}$ and $\hat{\tau}$ as

$$\hat{\tau}\cdot\hat{\sigma} = \lambda xy.\ [(x\ell(\tau)\ell(\sigma))^{\cup} y]\bowtie\alpha(\ell(\sigma))(\hat{\tau}x\Omega)(\hat{\sigma}\Omega y). \tag{3.8}$$

Then the following theorem holds.

Theorem 3.1
For any σ, τ in M, it holds that

$$\widehat{\tau\sigma} = \hat{\tau}\cdot\hat{\sigma}. \tag{3.9}$$

proof
Since it holds that

$$\hat{\tau}x\Omega = <(x\ell(\tau))^{\cup}\Omega>$$

and

$$\hat{\sigma}\Omega y = <(\Omega\ell(\sigma))^{\cup} y>,$$

the following equalities hold;

$$\bowtie\alpha(\ell(\sigma))(\hat{\tau}x\Omega)(\hat{\sigma}\Omega y)$$
$$=\bowtie<(x\ell(\tau)\ell(\sigma))^{\cup}(\Omega\ell(\sigma))><(\Omega\ell(\sigma))^{\cup} y>$$
$$=<(x\ell(\tau)\ell(\sigma))^{\cup}(\Omega\ell(\sigma))^{\cup} y>.$$

Hence, the theorem is proved as follows;

$$(\hat{\tau}\cdot\hat{\sigma})xy$$
$$=[(x\ell(\tau)\ell(\sigma))^{\cup} y]\bowtie\alpha(\ell(\sigma))(\hat{\tau}x\Omega)(\hat{\sigma}\Omega y)$$
$$=[(x\ell(\tau)\ell(\sigma))^{\cup} y]<(x\ell(\tau)\ell(\sigma))^{\cup}(\Omega\ell(\sigma))^{\cup} y>$$
$$=<(x\ell(\tau)\ell(\sigma))^{\cup} y>$$
$$=<(x\ell(\tau\sigma))^{\cup} y>$$
$$=\widehat{\tau\sigma}xy.$$

Now we extend the definition of a labeling morphism $\hat{}\sigma$ as $\hat{}\sigma : 2^{\Omega^*} \rightarrow 2^{\Omega^*}$.
From the fact that $\ell(\tau\sigma)=\ell(\tau)\ell(\sigma)$, it should be defined as

$$\hat{}\sigma(x\rho) = x\ell(\sigma)\rho \quad \text{for any subset } x \text{ of } \Omega^* \text{ and any } \rho \text{ in } L^*(M). \tag{3.10}$$

The identity morphism $\hat{\mathrm{I}}$ in 2^{Ω^*} is defined as

$$\hat{\mathrm{I}} = \lambda xy.\ <x^{\cup}y>. \tag{3.11}$$

For a morphism σ defined by (3.4), we define its reverse σ^{-} as follows;

$$\begin{array}{ll} \underline{\text{morphism}} & \sigma^{-} : S \rightarrow R_{-} \\ \underline{\text{where}} & \text{Pred}(X, \hat{}\sigma^{-}(Y)). \end{array} \qquad (3.12)$$

It should be noticed that $\hat{\sigma}^{-} \cdot \hat{\sigma}=1$ does not always hold. In fact, it holds if and only if

(1) $\forall x \varepsilon <X>, \exists y \varepsilon <Y> \quad \text{Pred}(x, y)=\text{true}$

and

(2) $\forall x, x' \varepsilon <X>, \forall y \varepsilon <Y>$
$\quad (\text{Pred}(x, y) \wedge \text{Pred}(x', y)) \supset (x=x').$

Let $(\ell_1/\ell_2, \ell_3/\ell_4, \ldots \ell_{2n-1}/\ell_{2n})$ be a unary operator that renames every semantic attribute $A\rho\ell_{2i}$ in the immediately following term to $A\rho\ell_{2i-1}$. Let σ and τ be same as before. The conjunction $\Lambda\sigma\tau$ of σ and τ is defined as

$$\begin{aligned} \widehat{\Lambda\sigma\tau} &= \Lambda\hat{\sigma}\hat{\tau} \\ &= \lambda xy.\ [(x\ell(\Lambda\sigma\tau)) \cup y] \\ &\quad \cap (\ell(\Lambda\sigma\tau)/\ell(\sigma))(\hat{\sigma}xy)(\ell(\Lambda\sigma\tau)/\ell(\tau))(\hat{\tau}xy), \end{aligned} \qquad (3.13)$$

while the disjunction $V\sigma\tau$ is defined as

$$\begin{aligned} \widehat{V\sigma\tau} &= V\hat{\sigma}\hat{\tau} \\ &= \lambda xy.\ [(x\ell(V\sigma\tau)) \cup y] \\ &\quad \cup (\ell(V\sigma\tau)/\ell(\sigma))(\hat{\sigma}xy)(\ell(V\sigma\tau)/\ell(\tau))(\hat{\tau}xy). \end{aligned} \qquad (3.14)$$

Here we also extend the definition of a restriction operator [Pred(X)] as

$$[\text{Pred}(X)] = \lambda x.\ [x][\text{Pred}(X)]<x \cup X>. \qquad (3.15)$$

Example 3.1

For a database with a single relation

R(/name/, /parent/, /birth date/, /sex/),

we can define the following morphisms, where $\omega(\sigma)$ denotes the English word such that "of $\omega(\sigma)$" corresponds to $\ell(\sigma)$.

(1) $\omega(\sigma 1)$ = 'parent', $\qquad$ $\omega(\sigma 1^{-})$ = 'child'

$$\begin{array}{ll} \underline{\text{morphism}} & \sigma 1 : R \rightarrow R \\ \underline{\text{where}} & \hat{}\sigma 1(\text{/name/})=\text{/parent/}. \end{array}$$

(2) $\omega(\sigma 2)$ = 'father'

$$\begin{array}{ll} \underline{\text{morphism}} & \sigma 2 : R \rightarrow R \\ \underline{\text{where}} & (\hat{}\sigma 2(\text{/name/})=\text{/parent/}) \wedge (\hat{}\sigma 2(\text{/sex/})=\text{'male'}). \end{array}$$

(3) $\omega(\sigma 3)$ = 'mother'

<u>morphism</u> $\sigma 3 : R \rightarrow R$
<u>where</u> $(\hat{}\sigma 3(\text{/name/}) = \text{/parent/}) \wedge (\hat{}\sigma 3(\text{/sex/}) = \text{'female'})$.

The morphisms $\sigma 2$ and $\sigma 3$ can be defined by $\sigma 1$, i.e.,

$$\hat{\partial}2 = [\text{/sex/}l(\sigma 1) = \text{'male'}]\hat{\partial}1,$$

$$\hat{\partial}3 = [\text{/sex/}l(\sigma 1) = \text{'female'}]\hat{\partial}1.$$

On the other hand, it holds that

$$\hat{\partial}1 = \vee\hat{\partial}2\hat{\partial}3.$$

With $\sigma 1$ and $\sigma 1^{-}$, we can define various English words as composition of these morphisms.

(4) $\omega(\sigma 4) = \text{'son'}$

$$\hat{\partial}4 = [\text{/sex/}l(\sigma 1^{-}) = \text{'male'}]\hat{\partial}1^{-}.$$

(5) $\omega(\sigma 5) = \text{'daughter'}$

$$\hat{\partial}5 = [\text{/sex/}l(\sigma 1^{-}) = \text{'female'}]\hat{\partial}1^{-}.$$

(6) $\omega(\sigma 6) = \text{'brother'}$

$$\hat{\partial}6 = [\text{/sex/}l(\sigma 1^{-})l(\sigma 1) = \text{'male'}](\hat{\partial}1^{-}\hat{\partial}1).$$

If a boy is not considered as a brother of himself, then $\sigma 6$ is expressed as follows, where diff(R)(S) denotes set difference of two relations with a same attribute set.

$$\hat{\partial}6 = \lambda xy.\text{diff } [\text{/sex/}l(\sigma 1^{-})l(\sigma 1) = \text{'male'}](\hat{\partial}1^{-}\hat{\partial}1)xy\ \hat{1}xy$$

(7) $\omega(\sigma 7) = \text{'grandfather'}$

$$\hat{\partial}7 = [\text{/sex/}l(\sigma 1)l(\sigma 1) = \text{'male'}]\hat{\partial}1\hat{\partial}1.$$

These examples give answers to the third questions in section 2.2.

<u>Example 3.2</u>

Suppose we have the following two relations:

R1(/project/, /staff/, /budget/)

R2(/title/, /author/, /journal/)

We can define the following morphism:

$\omega(\sigma) = \text{'document'}$, $\omega(\sigma^{-}) = \text{'project'}$

<u>morphism</u> $\sigma : R2 \rightarrow R1$
<u>where</u> $\hat{}\sigma(\text{/author/}) = \text{/staff/}$.

In this database, the composite morphism $\sigma\sigma$ or $\sigma^{-}\sigma^{-}$ is nonsense. However, we do not prohibit the use of these composite morphisms since it is not harmful to formally define these. For example, $\hat{\partial}\hat{\partial}\{\text{/title/}\}\{\text{/project/}\}$ is defined as follows from the definition of morphisms and their interpretation.

∂∂{/title/}{/project/}

=</title/ℓ(σ)ℓ(σ), /project/>

=[/staff/1=/author/1]</staff/1, $\underline{\text{/project}}$/></author/1, $\underline{\text{/title}}$/ℓ(σ)>

=[/staff/1=/author/1]([/staff/1, $\underline{\text{/project}}$/]R1)

([/staff/2=/author/2]</author/1, /staff/2></author/2, $\underline{\text{/title}}$/>)

=[/staff/1=/author/1]([/staff/1, $\underline{\text{/project}}$/]R1)

([/staff/2=/author/2]([/author/1, /staff/2]R1R2)

([/author/2, $\underline{\text{/title}}$/]R2)).

Since a cartesian product R1R2 appears at the end of the second line of the last transformation and no interrelational restriction is specified between R1 and R2, the /title/ and the /project/ in ∂∂{/title/}{/project/} do not have any significant relationship between themselves.

The composition τσ is meaningless iff the codomain relation of σ is equal to the domain relation of τ.

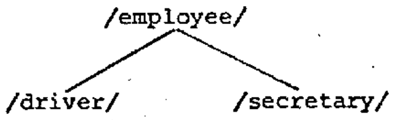
Example 3.3

In the following database, there are subordinate relationships between attributes, i.e., drivers and secretaries are also employees.

R1(/employee/, /salary/, /address/)

R2(/driver/, /license no./)

R3(/secretary/, /typing speed/)



These subordinate relationships are also represented by morphisms below.

σ1 morphism σ1 : R2 → R1
where $\hat{\sigma}$1(/driver/)=/employee/.

σ2 morphism σ2 : R3 → R1
where $\hat{\sigma}$2(/secretary/)=/employee/.

We will return to these subordinate relationships afterwards in section 7.

4. INFORMATION SPACE MODEL

4.1. Elementary Morphism

We say a morphism σ is elementary if it is defined in the following form;

morphism σ : R → S
where $\hat{\sigma}$(A)=B, (4.1)

where A and B are elements of $\Omega(R)$ and $\Omega(S)$ respectively. In most of the applications, the most general form of morphism definitions may be as follows;

$$\begin{array}{l} \underline{\text{morphism}} \quad \sigma : R \rightarrow S \\ \underline{\text{where}} \quad (\hat{}\sigma(X_1)=Y_1) \wedge \text{Pred}_1(\hat{}\sigma(X_2)) \wedge \text{Pred}_2(Y_2), \\ (\ X_i \subset \Omega(R),\ Y_i \subset \Omega(S),\ |X_1|=|Y_1|\ ). \end{array} \tag{4.2}$$

For such morphisms, the following theorem holds.

Theorem 4.1

Any morphism with the form

$$\begin{array}{l} \underline{\text{morphism}} \quad \sigma : R \rightarrow S \\ \underline{\text{where}} \quad \bigwedge_{i=1}^{k} (\hat{}\sigma(A_i)=B_i) \wedge \text{Pred}_1(\hat{}\sigma(X)) \wedge \text{Pred}_2(Y) \\ (\ A_i \in \Omega(R),\ \ B_i \in \Omega(S),\ \ X \subset \Omega(R),\ \ Y \subset \Omega(S)\ ) \end{array} \tag{4.3}$$

can be defined using elementary morphisms.

proof

Let σ_i denote an elementary morphism defined as

$$\begin{array}{l} \underline{\text{morphism}} \quad \sigma_i : R \rightarrow S \\ \underline{\text{where}} \quad \hat{}\sigma_i(A_i)=B_i. \end{array} \tag{4.4}$$

Then it holds that

$$\hat{\sigma} = [\text{Pred}_1(\hat{}\tau(X))][\text{Pred}_2(Y)]\hat{\tau}, \tag{4.5}$$

where

$$\hat{\tau} = \bigwedge_{i=1}^{k} \hat{\sigma}_i. \tag{4.6}$$

This theorem indicates that only a set of elementary morphisms is sufficient to describe synthetic interrelational relationships in an object database. A set of elementary morphisms from which any morphism in M can be derived is denoted by M_0.

For a given set R of relations, a set M of morphisms is said to be sufficient if any synthetic interrelational relationships in R can be represented by elements of this set. A pair (R, M) is called an information space schema if M is sufficient with respect to R. In most of the applications, a schema (R, M) has an equivalent schema (R, M_0), where M_0 is a set of elementary morphisms. The schema (R, M_0) is called a normal form schema of (R, M). It should be noticed that the number of elementary morphisms is always finite. Therefore, we can always define information space schema with finite description.

4.2. World and View Point

For each ρ in $L^*(M_0)$, the set $\Omega\rho$ forms a world of information labeled with ρ.

We denote this world by $W\rho$. Let f_σ be defined as

$$f_\sigma = \lambda x.\ \hat{\sigma}x. \tag{4.7}$$

The morphism f_σ is interpreted as a view point shifter that moves the view point from the world $W\,l(\sigma)\rho$ to $W\rho$. The world W is especially called a base world.

Example 4.1

$I = (R,\ M_0)$

$R = \{\mathrm{R1},\ \mathrm{R2}\}$

R1(/project/,/budget/,/manager/,/employee/,/salary/,/department/,
/location/,/subproject/)

R2(/report no./,/title/,/author/,/journal/,/key word/)

$M_0 = \{\sigma 1,\ \sigma 1^-,\ \sigma 2,\ \sigma 2^-,\ \sigma 3,\ \sigma 3^-\}$

$\sigma 1$ $\omega(\sigma 1)$ = 'project', $\omega(\sigma 1^-)$ = 'document'

morphism $\sigma 1 : \mathrm{R1} \rightarrow \mathrm{R2}$
where $\hat{\ }\sigma 1$(/employee/)=/author/

$\sigma 2$ $\omega(\sigma 2)$ = 'manager', $\omega(\sigma 2^-)$ = 'subordinate'

morphism $\sigma 2 : \mathrm{R1} \rightarrow \mathrm{R1}$
where $\hat{\ }\sigma 2$(/employee/)=/manager/

$\sigma 3$ $\omega(\sigma 3)$ = 'subproject', $\omega(\sigma 3^-)$ = 'superproject'

morphism $\sigma 3 : \mathrm{R1} \rightarrow \mathrm{R1}$
where $\hat{\ }\sigma 3$(/project/)=/subproject/

The diagramatic representation of this schema is shown in Fig.8. In Fig.9, we show the pictorial representation of the relationships among worlds. An eye in Fig.9 indicates the view point.

4.3. Formal Description of an Information Subspace

By semantic subspace, we mean a relation over a subset X of semantic attributes that satisfies the condition Pred(Y), where Y is also a subset of semantic attributes. Let this subspace be named W. Then W is defined as

$$W = [X][\mathrm{Pred}(Y)]\langle X \cup Y\rangle. \tag{4.8}$$

We formally describe W as

S-subspace W
over X
where Pred(Y). (4.9)

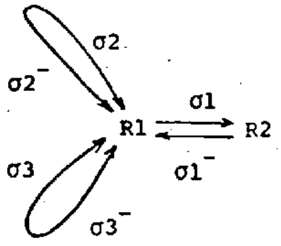
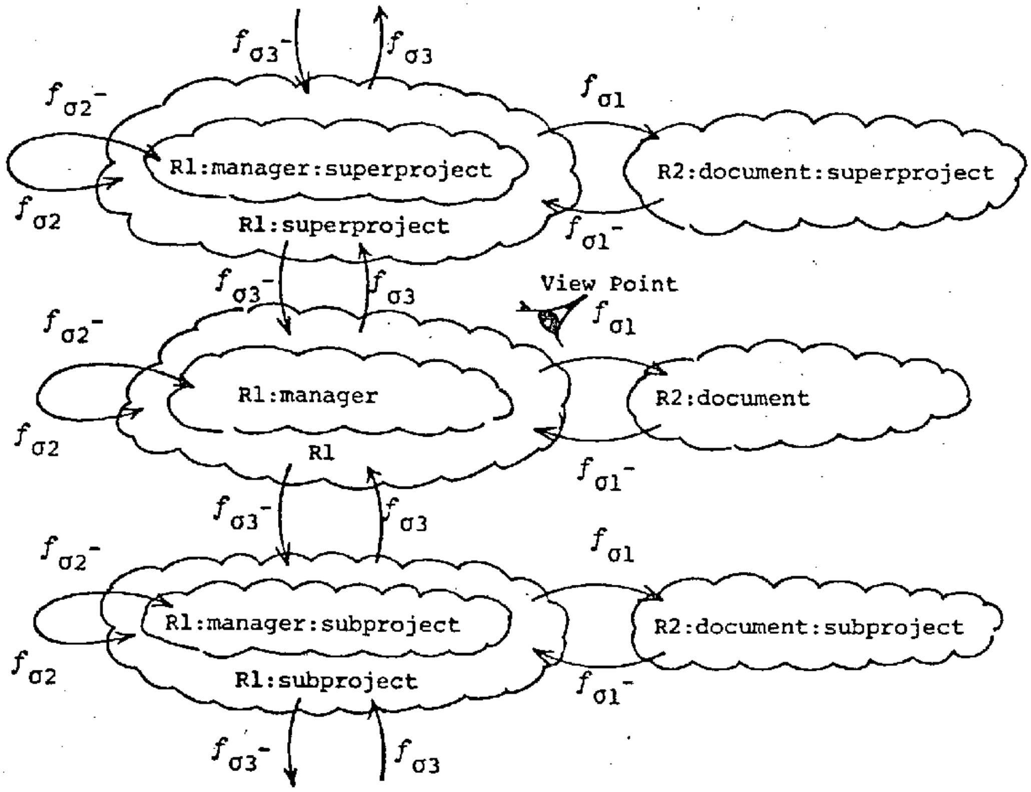


Fig. 8. Diagramatic representation of the schema in example 4.1.



It is assumed that R1 is in a focus.
Meanigless composition of morphisms is neglected.

Fig. 9. Pictorial representation of the relatioships among worlds.

Let W_1 and W_2 be a semantic subspace. Then the intersection of these relations is equal to a relation W defined as

$$\begin{aligned} W &= W_1 \cap W_2 \\ &= \cap([X_1][\mathrm{Pred}_1(Y_1)]<X_1 \cup Y_1>)([X_2][\mathrm{Pred}_2(Y_2)]<X_2 \cup Y_2>) \\ &= [X_1 \cap X_2][\mathrm{Pred}_1(Y_1) \wedge \mathrm{Pred}_2(Y_2)]<X_1 \cup Y_1 \cup X_2 \cup Y_2>. \end{aligned}$$

Therefore W is also a semantic subspace described as

S-space W
over $X_1 \cap X_2$
where $\mathrm{Pred}_1(Y_1) \wedge \mathrm{Pred}_2(Y_2)$. (4.10)

However, the union of two semantic subspaces can not be described as (4.9). Hence, $W_1 \cup W_2$ is not a semantic subspace unless one of the followings holds;

(1) $W_1 \supset W_2$,

(2) $W_2 \supset W_1$,

(3) $X_1 = X_2$,

(4) $\mathrm{Pred}_1(Y_1) = \mathrm{Pred}_2(Y_2)$.

We define an information subspace as follows;

(1) a semantic subspace is an information subspace,

(2) if W_1 and W_2 are information subspaces then $W_1 \cup W_2$ is an information subspace.

(3) if W_1 and W_2 are information subspaces then $W_1 \cap W_2$ is an information subspace.

(4) only those obtained by finite applications of the above rules are information subspaces.

This definition of an information subspace is a very natural formal definition of a meaningful subspace of information mentioned in section 2.2. The above observation indicates that any information subspace can be defined as a union of finite number of semantic subspaces in such a way as follows;

I-subspace W
where $W = \cup_{i \in I} W_i$, (4.11)

for every $i \in I$,

S-subspace W_i
over X_i
where $\mathrm{Pred}_i(Y_i)$. (4.12)

For example, in a database in example 3.1, the information about the antecedents of J. Smith and that about the descendants of R. King forms the information subspace W described below;

I-subspace W
where $W = W_1 \cup W_2$,
  S-subspace W_1
  over /name/$(l(\sigma_1))^*$, /birth date/$(l(\sigma_1))^*$,
  /sex/$(l(\sigma_1))^*$
  where /name/ = 'J. Smith',
  S-subspace W_2
  over /name/$(l(\sigma_1^{-}))^*$, /birth date/$(l(\sigma_1^{-}))^*$,
  /sex/$(l(\sigma_1^{-}))^*$
  where /name/ = 'R. King',

where Al^* denotes a set of all the semantic attributes $A\rho$ such that ρ is a list of finitely iterated l.

This facility to define information subspaces enhances the capability of the database management system in access control by query modifications proposed in [ASTR76][CHAM76]. This problem as well as other applications of this facility will be reported elsewhere.

5. RECURSIVE MORPHISM AND DIRECT SUM DECOMPOSITION

In this section, we further investigate semantic structures induced by recursive morphisms. We restrict our discussion to such a case with only one first normal form relation. Since this section deals only with the decomposition of a relation with a recursive morphism and a recursive morphism is defined within a single relation, the following result is also applicable to the cases with more than one relations.

Let R be an object first normal form relation. A recursive attribute is defined if there exists two attributes A and B in the attribute set $\Omega(R)$, such that

(1) the domains of these two attributes Dom(A) and Dom(B) intersect with each other,
(2) we can assume for any $x \in \mathrm{Dom}(A)$ x is also a member of Dom(B) without any contradiction and vice versa,
(3) either of the following two MVDs holds;

$$A \twoheadrightarrow \Omega_1 \mid \Omega_2 B,$$

$$B \twoheadrightarrow \Omega_1 \mid \Omega_2 A,$$

where $\{A\}$, $\{B\}$, Ω_1, Ω_2 are partition of $\Omega(R)$.

A pair of these attributes A and B is called a recursive attribute pair. Suppose that A and B is a recursive attribute pair satisfying $A \twoheadrightarrow \Omega_1 \mid \Omega_2$ B, where {A}, {B}, Ω_1, Ω_2 form a partition of $\Omega(R)$. Then B is called a recursive attribute, and A a superordinate attribute. We introduce a new attribute B^a corresponding to the antonym of B. The superordinate attribute of B is denoted by B^s.

For example, /name/ and /parent/ form a recursive attribute pair in example 3.1. Since it holds that

$$/\text{name}/ \twoheadrightarrow /\text{birth date}//\text{sex}/ \mid /\text{parent}/$$

but that

$$/\text{parent}/ \twoheadrightarrow /\text{name}//\text{birth date}//\text{sex}/,$$

/parent/ is a recursive attribute. We can introduce /child/ as an antonym of /parent/.

Suppose that there exists no such subset Ω' of Ω that satisfies an MVD

$$\phi \twoheadrightarrow \Omega' \quad \text{in } \Omega,$$

where ϕ denotes an empty set. If there exists one, then we can apply the following result to Ω' and $\Omega(R)-\Omega'$ independently because R is a Cartesian product of these in such a case.

Suppose that there exists h recursive attributes B_i ($1 \le i \le h$). Let Ω° denote

$$\Omega^\circ = \Omega \cup (\cup_{1 \le i \le h} \{B_i, B_i^a, B_i^s\}).$$

We call the following condition an S-condition;

$$\{B_i^s \mid 1 \le i \le h\} \twoheadrightarrow \{B_i\} \quad \text{in} \quad \{B_i^s, B_i \mid 1 \le i \le h\}.$$

Suppose that a set of h recursive attributes $\{B_i \mid 1 \le i \le h\}$ in Ω satisfies S-condition. Let Ω^+ denote $\Omega^\circ-(\cup_{1 \le i \le h} \{B_i^a\})$, and Ω^+_i be a minimal subset Ω' including B_i such that

$$\{B_i^s\} \twoheadrightarrow \Omega' \quad \text{in } \Omega^+.$$

Let Ω_i be defined as

$$\Omega_i = \Omega^+_i \cup \{B_i^a\} \quad (1 \le i \le h),$$
$$\Omega_0 = \Omega^\circ-(\cup_{1 \le i \le h} \Omega_i).$$

Then Ω° is represented as a direct sum of Ω_i ($0 \le i \le h$).

Theorem 5.1

$$\Omega^\circ = \Omega_0 \oplus \Omega_1 \oplus \dots \oplus \Omega_h. \tag{5.1}$$

This means that $\{\Omega_i \mid 0 \le i \le h\}$ is a partition of Ω°. Fig.10 shows an example relation with 2 recursive attributes and its direct sum decomposition.

We can define a recursive morphism σ_i for each recursive attribute pair (A_i, B_i)

attribute set :

Ω = {/person/,/project/,/section/,/department/,/company/,/subsidiary/,
/location/,/subproject/,/subproject-name/}

We assume that a project is called by different names under different superprojects.

recursive attributes :

A_1 = /subsidiary/

A_2 = /subproject/

h = 2

antonym :

A_1^a = /parent company/

A_2^a = /superproject/

superordinate attributes :

A_1^s = /company/

A_2^s = /project/

$\Omega^\circ = \Omega \cup$ {/parent company/, /superproject/}

$\Omega^+ = \Omega$

S-condition :

{/company/,/project/} $\twoheadrightarrow$ {/subsidiary/} | {/subproject/}

in {/company/,/project/,/subsidiary/,/subproject/}

direct sum decomposition :

Ω_1 : {/company/} $\twoheadrightarrow$ {/subsidiary/} in Ω^+

Ω_1 = {/subsidiary/, /parent company/}

Ω_2 : {/project/} $\twoheadrightarrow${/subproject/,/subproject-name/} in Ω^+

Ω_2 = {/subproject/,/superproject/,/subproject-name/}

Ω_0 : $\Omega_0 = \Omega^\circ - \Omega_1 - \Omega_2$

= {/person/,/project/,/section/,/department/,/company/,/location/}

Fig. 10. Direct sum decomposition of a relation with two recursive attributes.

as follows, where B_i is assumed to be a recursive attribute.

$$\underline{\text{morphism}} \quad \sigma_i : R \rightarrow R$$
$$\underline{\text{where}} \quad A_i l(\sigma_i) = B_i. \qquad (5.2)$$

It is recommended by various reports that the information about a recursive pair (A_i, B_i) should be separated from the rest of R. This is done by decomposing $\langle\Omega^{\circ}\rangle$ into $\{\langle\Omega_i\rangle \mid 0 \leq i \leq h\}$. If some Ω_i has a set of recursive attribute pairs satisfying S-condition then Ω_i is further decomposed by the direct sum decomposition method mentioned above. The original relation $\langle\Omega\rangle$ is related to $\{\langle\Omega_i\rangle\}$ by the following relation;

$$\langle\Omega\rangle = [\Omega][\Lambda_{0\leq i\leq h} (B_i{}^s = B_i{}^a)]\Pi\langle\Omega_i\rangle. \qquad (5.3)$$

6. DESIGN OF AN INFORMATION SPACE MODEL

Suppose that set $\mathcal{R}$ of first normal form relations are given, and that, for any two different relations R and S in $\mathcal{R}$, the attribute sets of these are mutually disjoint. This condition is always satisfiable by proper renaming of attributes.

The procedure for the design of an information space schema for $\mathcal{R}$ is summarized below.

(1) $W = \Pi_{R\in\mathcal{R}} R$ (the base world).

(2) For each R in $\mathcal{R}$, find out a set of recursive attributes $\{B_i \mid 1\leq i\leq h\}$ satisfying S-condition and decompose R by the direct sum decomposition method. For each component of the decomposition of R, apply this step recursively until all components can not be further decomposed.
Define a set P of recursive morphisms each of which corresponds to some recursive attribute found by this step.

(3) Find out other elementary morphisms in $\mathcal{R}$. Let E denote a set of them. Let M_0 be the union of P and E. A pair $(\mathcal{R}, M_0)$ is the designed information space schema.

After the third step, we can apply our 4NF D-tree schema theory [TANA79] to decompose each first normal form relation in $\mathcal{R}$ into the fourth normal form relations. D-tree schema theory gives us the clear description about analytic interrelational relationships among the fourth normal form relations obtained by the decomposition.

7. QUERY LANGUAGE AND VOCABULARIES

Queries using semantic attributes can be described in the following form;

select X
where Pred(Y),

where X and Y denotes subsets of Ω^*. The execution of this query corresponds to the evaluation of

$[X][\mathrm{Pred}(Y)]\langle X \vee Y\rangle$.

The relation $\langle X \vee Y\rangle$ can be evaluated following the definition in section 3.2.

However, queries using semantic attributes are not sufficient to make them easy to understand.

Consider the database in Fig.2 (a). This has two relations below;

R1(/novel 1/, /author/)

R2(/character/, /novel 2/).

In this database, there are two morphisms σ and σ^- defined as

morphism $\sigma : R1 \to R2$
where $\hat{}\sigma$(/novel 1/) = /novel 2/.

In this case, it is very difficult to find out a proper adjective phrase for $\zeta(\sigma)$. To solve this problem, we define vocabularies used in queries of this database with attribute names and morphisms. This is done as follows;

author ::= /author/$\zeta(\sigma)$,
novel ::= /novel 1/$\zeta(\sigma)$,
character ::= /character/.

Queries are written with these vocabularies. They are translated into the right hand sides of definitions by a query translator.

However, the definition of a word 'novel' as aboye may lead to wrong evaluation. Consider a query:

select novel, character.

This is evaluated as

<novel, character>
=</novel 1/$\zeta(\sigma)$, /character/>
=[/novel 1/$\zeta(\sigma)$, /character/][/novel 1/$\zeta(\sigma)$=/novel 2/]
$\alpha(\zeta(\sigma))$</novel 1/></novel 2/, /character/>
=[/novel 1/$\zeta(\sigma)$, /character/][/novel 1/$\zeta(\sigma)$=/novel 2/]
$\alpha(\zeta(\sigma))$[/novel 1/]R1 R2.

This is not equal to the desired result R2. Same is true with respect to the definition:

novel ::= /novel 2/.

This problem occurs if two attributes A in R and B in S are related and either of <A> or <B> is a subset of the other. This is solved by considering a new relation R0 that is a unary relation ∪<A><B>. Then the new information space schema of this database becomes as follows;

$\mathcal{R}$ = {R0, R1, R2}

R0(/novel/) = ∪[/novel 1/]R1 [/novel 2/]R2
R1(/novel 1/, /author/)
R2(/character/, /novel 2/),

M_0 = {σ1, σ2}

<u>morphism</u> σ1 : R1 ⟶ R0
<u>where</u> ^σ1(/novel 1/) = /novel/

<u>morphism</u> σ2 : R2 ⟶ R0
<u>where</u> ^σ2(/novel 2/) = /novel/.

The vocabularies are defined as

author ::= /author/ℓ(σ1),

novel ::= /novel/,

character ::= /character/ℓ(σ2).

The query is evaluated as

<novel, character>
=</novel/, /character/ℓ(σ2)
=[/novel/, /character/ℓ(σ2)][/novel 2/ℓ(σ2)=/novel/]:
α(ℓ(σ2))</novel 2/, /character/>R0
=α(ℓ(σ2))</novel 2/, /character/>
=α(ℓ(σ2))R2.

<u>Example 7.1</u>

Consider the database with subordinate relationships between attributes shown in example 3.3. Since it holds that

</employee/> ⊃ </driver/>,

and

</employee/> ⊃ </secretary/>,

we can define the vocabularies of this database as follows;

employee ::= /employee/,

salary ::= /salary/,

address ::= /address/,

driver ::= /driver/ℓ(σ1),

license no. ::= /licence no./ℓ(σ1),

secretary ::= /secretary/$l(\sigma 2)$,

typing speed ::= /typing speed/$l(\sigma 2)$.

Example 7.2

The final example is a case with recursive morphisms shown in example 3.1. The vocabularies of this database are as follows;

name ::= /name/

parent ::= /parent/

birth date ::= /birth date/

sex ::= /sex/

father ::= /name/$l(\sigma 2)$

mother ::= /name/$l(\sigma 3)$

child ::= /name/$l(\sigma 1^-)$

son ::= /name/$l(\sigma 4)$

daughter ::= /name/$l(\sigma 5)$

grandparent ::= /parent/$l(\sigma 1)$

grandfather ::= /name/$l(\sigma 2)l(\sigma 1)$

grandmother ::= /name/$l(\sigma 3)l(\sigma 1)$

⋮

descendant ::= /name/$(l(\sigma 1^-))^*$

antecedent ::= /name/$(l(\sigma 1))^*$

of parent ::= $l(\sigma 1)$

of father ::= $l(\sigma 2)$

of mother ::= $l(\sigma 3)$

of child ::= $l(\sigma 1^-)$

⋮

As shown above, the vocabularies of a database consist of the noun definition and adjective definition. The detail formalization of vocabularies is reported elsewhere.

8. CONCLUDING REMARKS

While the relational model has been an infological framework of database theories, the information space model in this paper has been proposed as an infosemantic framework of database theories. Various semantic problems need theoretical basis for semantics, especially interrelational semantics. The idea of this model is very simple, i.e., a pair of R and M_0. The model is sufficient to solve various semantic problems shown in section 2.2.

The information space model should not be confused with the studies of functional programming in data bases [BUNE79][SHIP79]. Their main concern is the query program manipulating information. The information space model concerns the description of infosemantic structures of a schema as well as the improvement of query languages. While our model can cope with query programming problems as shown in section 7, recent studies on functional query language can not cope with the general description of information structures. Especially, they can not describe meaningful subspaces of information.

The use of a dictionary that defines nouns and adjectives from attribute names and morphisms may be a new approach to database semantics. This approach is enabled by the finiteness of the definition of an information space schema. We call this approach a denotational semantic approach to database semantics.

Reference

[ASTR76] M.M. Astrahan et al, "System R: A Relational Approach to Data Base Management," ACM Trans. on Database Systems, Vol.1, No. 2 (June 1976).

[BUNE79] P. Buneman, R.E. Frankel, "FQL--A Functional Query Language," Proc. of ACM-SIGMOD 1979, Boston, May 1979, pp.52-59.

[CHAM76] D.D. Chamberlin et al, "SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control," IBM J. of R & D, Vol. 20, No. 6 (Nov. 1976).

[SHIP79] D. Shipman, "The Functional Data Model and the Data Language DAPLEX," Proc. of ACM-SIGMOD 1979, Boston, May 1979.

[TANA77] Y. Tanaka, T. Tsuda, "Decomposition and Composition of a Relational Database," Proc. of 3rd VLDB Conf., Tokyo, Oct. 1977, pp.454-461.

[TANA79] Y. Tanaka, "Logical Design of a Relational Schema and Integrity of a Data Base," Proc. of IFIP TC-2 Working Conf. on Data Base Architecture, also in *Data Base Architecture* (G. Bracchi, G.M. Nijssen ed.), North-Holland, 1979, pp.297-316.

USES OF KNOWLEDGE BASES IN DATA BASE MANAGEMENT SYSTEMS

Setsuo OHSUGA*

*Institute of Space and Aeronautical Science, University of Tokyo.
4-6-1, Komaba Meguro-ku, Tokyo 153, Japan

This paper describes a knowledge based system named the KAUS(Knowledge Acquisition and Utilization System) designed to aid man in his intellectual works such as scientific research, decision making, engineering design and medical diagnosis. In particular, its subsystem named the SBDS(Structure Based Deduction System) is discussed in detail, which takes charge of handling both the knowledge base and the data bases. In this system, information is processed by being converted among four different forms, i.e., external representation such as natural language and graphic representation, program , data base and the representation of knowledge. Some important mappings between different forms and the deductive inference algorithm are defined.

1. INTRODUCTION

This paper describes a system named the SBDS(Structure Based Deduction System). This is a subsystem of a knowledge system named the KAUS(Knowledge Acquisition and Utilization System) that is being developed as an intelligent interactive system to aid man in his works such as industrial design, decision making, medical diagnosis and so on. The KAUS is expected to accept, understand and reply user's request presented in a natural-(like)-language and/or in a simple graphic form, or in other words, in man's everyday forms of representations. We call representations in these forms the external representations.

The SBDS takes charge of handling the knowledge bases and the data bases in the KAUS. It adopts, as its basic formalism for representation of knowledge, an extended form of the first order predicate logic because it is considered very suited for conversion to/from any of three major forms of information appearing in and arround the computer systems, i.e., (1) external representations, (2) data bases and (3) programs. Information is processed by being converted from one form to another among them. In the KAUS, all conversions are achieved via the representation of knowledge as is shown in Fig. 1.

Note here that there is no exact one-to-one correspondence among expressions in these different forms because of the difference in the expressive power of and redundancy contained in each representation. In general, the expressive power of the external representation is the largest among all. Therefore, it is often necessary to change a representation of information in the same formalism preserving its meaning until, at last, a representation is reached which has a corresponding representation in another form. This is the role of the deductive inference mechanism.

The advantage obtained by using the representation of knowledge is that it can be defined to have aspects very close to every other form so that the conversion to/from the other forms can be cleary defined.

Another advantage is in the fact that it can also be defined so that the inference algorithm can well be defined to it.

The mappings that the system should be provided with are those listed in Table 1.

In this paper, the outline of the SBDS is presented. The data bases viewed through representation of knowledge and the conversion algorithm from the latter to the former is discussed with a simple example.

2. REPRESENTATION OF KNOWLEDGE

From the discussion above and Fig.1, the requirements for the representation of knowledge are as follows:

(1) Conversion to/from external representation are possible and its algorithm can be obtained,
(2) Conversion to program is possible and its algorithm can be obtained,
(3) Conversion to/from data bases are possible and its algorithm can be obtained and
(4) Deductive inference algorithm can be well defined.

The basic configuration of the KAUS is as is shown in Fig.2 in which the first order predicate logic is adopted as the kernel concept of knowledge representation. We say it the kernel because, in the real system, it is extended towards higher order logic and also it is modified and combined with a concept of hierarchically structured universe to get high logical and physical performances. Since, however, our primary objective is to show how information is processed by being converted among different forms, we first discuss the simpler case of the first oder logic.

2.1 Representation of knowledge in the predicate form

We first show an example of representation of knowledge in the KAUS as follows:

$$(\forall x/\mathrm{PHYSOB})(\forall y/\mathrm{REAL})(\forall z/\mathrm{REAL})(\forall u/\mathrm{REAL})[(\mathrm{mass}\ x\ y)\cap(\mathrm{vol}\ x\ z)\cap(\mathrm{div}\ u;\ y\ z)\Rightarrow(\mathrm{spc.grv}\ x\ u)]$$

where each predicate has the following meaning ;
(mass x y): "mass of x is y",
(vol x y): "volume of x is z",
(div u; y z):"y/z=u" and

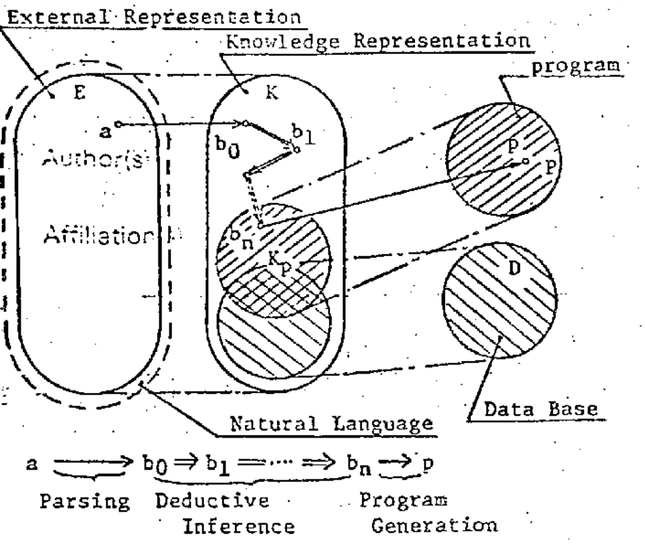


Fig.1 Relations among various representations

1	E→K : Parsing, Pattern Classification etc.
2	K→E : Output Generation
3	K→P : Program Generation
4	(K→D): Data Base Description
5	D→K : Generalization of Representation
6	K→K : Deductive Inference

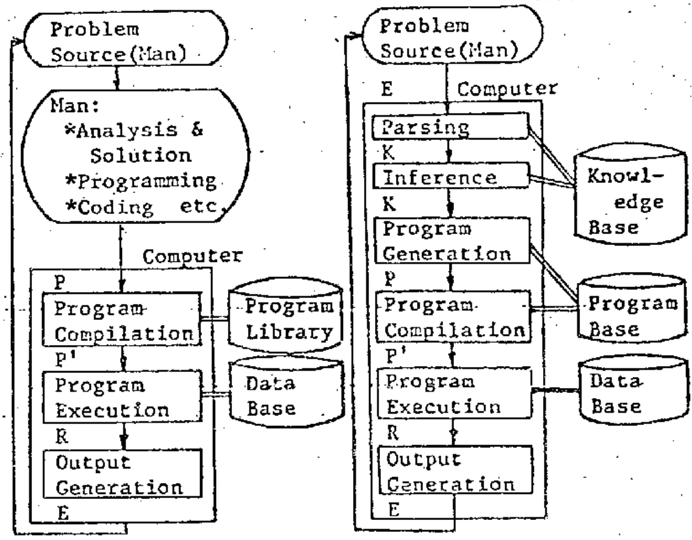
Table 1 Necessary onversions

(spc.grv x u):"specific gravity of x is u".
This formula is read as "for every physical object x and all real values y, z and u, if mass of x is y, volume of x is z and the quotient of y divided by z is u, then specific gravity of x is u." It gives the definition of "specific gravity". In general, definitions of concepts, mathematical theories, physical laws, facts and so on can be represented in the same style. The collection of such the formulas forms a knowledge base, and so, each formula as this example is called the knowledge element.

Note that, in this formula, a domain is explicitly specified to each variable and the formula is defined only over the specified domains of variables. It is very natural way of representation because these domains are brought in by the external words when a knowledge element is given by man in the form of external representation. In the KAUS, the logical structure of the matrix of a formula is representsed by an AND-OR tree as shown in Fib 3(b). Information on the variables are collected in a table called the variable table or V-table in short as is shown in the figure.

2.2 Knowledge structure

A universe is defined as the set of all objects referred in the system. Then a domain of variable in the predicate is one of its subsets. The universe itself forms a hierarchical structure based on the set-theoretical relations between sets in it.
In the KAUS this structure is represented by a graph



(a)Conventional System (b)New System

E; External Representation
K; Knowledge Representation
P; Program
R; Result

Fig.2 Conventional and new computer system

composed of nodes and two types of arcs. Each node represents a defined set and arcs represents two types of the set-theoretical relations between nodes; inclusion and disjointness. The structure thus formed is called the skeleton structure and is illustrated in Fig.3(a). To each node corresponds one or more external word(s). The association is made by a dictionary each entry of which contains a character string and a pointer to the corresponding node.

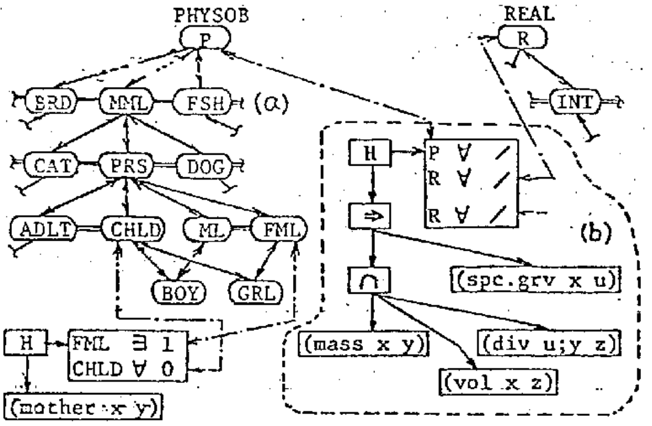
Every knowledge element is connected to one or more nodes in the structure in such a way that if the knowledge element has a domain X, then it is linked to the node X (ref. Fig.3(b)).

2.3 Procedural Type Atom (PTA) and Non-procedural Type Atom

In a practical application, there may be a lot of knowledge elements in the system. Because the system must allow user to define and use his own predicate, the set of predicate symbols is kept open. However, some predicates of which predicate symbols are defined by some external words such as verbs, adjectives, prepositions and some nouns are defined in advance.

All atomic formulas or atoms in short are classified into two classes: procedural type atoms(PTAs) and non-procedural type atoms(NTAs). To each PTA a procedure is defined, which evaluates the logical value of the predicate as well as the value of some variable(s) in it. A few examples are as follows;
(add z; $\underline{x}$ $\underline{y}$):"if x + y = z, then true",
(div z; $\underline{x}$ $\underline{y}$):"if x/y = z, then true",
(g-than $\underline{x}$ $\underline{y}$):"if x y, then true",
(sin y; $\underline{x}$) :"if sin x = y, then true".

The set of these procedures forms the procedure base. Each procedure is evoked when a PTA in a formula becomes ready and is picked out for evaluation, that



(∀y/CHLD)(∃x/FML)(mother x y)
(:"every child has mother")

I-Relation;
MML ⊃ PRS, CAT, DOG, ---
PRS ⊃ ADLT, CHLD, ML, FML, --
CHLD ⊃ BOY, GRL, --
ML ⊃ BOY, --

D-Relation;
PRS ∧ CAT = φ
PRS ∧ DOG = φ
ML ∧ FML = φ

BRD	:BIRD
MML	:MAMMAL
FSH	:FISH
PRS	:PERSON
ADLT	:ADULT
CHLD	:CHILD
ML	:MALE PERSON
FML	:FEMALE PERSON
GRL	:GIRL
INT	:INTEGER

Fig.3 Structure of concepts and knowledge

is, a value has been given to every independent variable (underlined in the above examples) during the deductive operation as will be described in Sec.3. If a PTA includes one (or more) dependent variable(s), the result of the operation of the procedure defined to the PTA is substituted into it (them).

Besides these PTAs defined in the system, user can define his own PTA by writing and registering a procedure for it.

An NTA, on the other hand, is the one the logical value of which can not be obtained by itself. Its truthhood is proved if and only if it is implied by another formula that has already been asserted true.

3. DEDUCTIVE INFERENCE

3.1 Principle of deductive inference

Deductive inference algorithm is defined to the set of knowledge elements. There are two types of knowledge elements; simple assertion type written in the form of $A_1 \cap A_2 \cap --- \cap A_k$ and conditional type written as $B \Rightarrow C$ and read as "if B then C" or, in the logical term, "B implies C" where $B = B_1 \cap B_2 \cap --- \cap B_\ell$.

Suppose a query Q is presented. If it is proved that Q is implied by some A_i, $i\in\{1,2,\dots,k\}$ of the assertion type knowledge element, i.e. $A_i \Rightarrow Q$, then Q is proved true. On the other hand, if Q is implied by the consequense of a conditional type formula, that is, $C \Rightarrow Q$, then the truthhood of Q depends on that of B. Hence Q can be replaced by B. We say that P satisfies the implicative condition to Q if either P;A_1 --- $\cap A_k$ and $A_i \Rightarrow Q$,$i\in\{1,2,--,k,\}$ or P; $B_1 \cap B_2 \cap -- \cap B_\ell \Rightarrow Q$ and $C \Rightarrow Q$.

In many cases, the query Q includes more than one atoms. Then, some atoms, say D_j, is selected out of them, and above test is applied to it. That is, if $Ai \Rightarrow D_j$ holds to the knowledge of the form $A_1 \cap --- \cap A_k$. then D_j is true. In this case, a label T is given to the node D_j in the AND-OR tree. On the other hand, if $C \Rightarrow D_j$ holds to the knowledge of the form $B_1 \cap --- \cap B_\ell \Rightarrow C$, then D_j is replaced by $B_1 \cap --- \cap B_\ell$ and the new AND-OR tree Q' is obtained to which the same procedure as above is repeated.

Thus the deductive procedure is formed of four major routines as shown in Fig.4 in which,

(1) SR(Selection Routine) selects some atom D_j out of atoms in the query,
(2) TIC(Test for Implicative Condition), given a query Q, searches and retrieves a knowledge element P that satisfies the implicative condition to the query,
(3) RR(Replacement Rule), given the query and the knowledge element P retrieved in the TIC above, replaces the old query by a new formula Q' which is obtained by modifying Q depending on P, and
(4) TT(Test for termination) evaluates the logical value of the query Q' above and determines if the process terminates.

3.2 SR

The SR selects an NTA in the query which includes the largest number of universally quantified variables or constants or both among all NTAs not evaluated yet. A variety of strategies can be applied at this stage to select one out of many (abbreviated here).

3.3 TIC

3.3.1 Test for the implicative relation

Suppose a couple of single atom formulas, say, P and Q, is presented. The TIC determines if the implicative relation, $P \Rightarrow Q$, holds between them by resolving the logical implicative relation into a set of set-theoretical relations between domains of each corresponding variables and as well into the condition on the orders of variabled in the prenex of each formula.

Let P;$(Q_{pi_1} x_{i_1}/X_{i_1})\dots(Q_{pi_n} x_{i_n}/X_{i_n})(F\ x_1 \dots x_n)$ and
Q;$(Q_{qj_1} y_{j_1}/Y_{j_1})\dots(Q_{qj_n} y_{j_n}/Y_{j_n})(F\ y_1 \dots y_n)$,

then $P \Rightarrow Q$ holds if

(1) for every pair of variables (x_i y_i), the condition shown in Table 2 holds, and
(2) there does'nt exist any such indices pair (i j) that (a) in P, the universally quantified x_i precedes the existentially quantified x_j, i.e. P;... ..$(\forall x_i/X_i)\dots(\exists x_j/X_j)\dots(F\ x_1 \dots x_n)$, while (b) in Q, the existentially quantified y_j precedes the universally quantified y_i,i.e., Q; ...$(\exists y_j/Y_j)$.. ..$(\forall y_i/Y_i)\dots(F\ y_1 \dots y_n)$.

This process is equivalent to the unification of the resolution principle.

Q_{pi}	$Q_{qi}(\bar{Q}_{qi})$	Conditions on Domains
∀	∀ (∃)	$X_i \supset Y_i$
∀	∃ (∀)	$X_i \wedge Y_i \neq \phi$
∃	∀ (∃)	Non Implicative
∃	∃ (∀)	$X_i \subset Y_i$

Table 2. Implicative rule

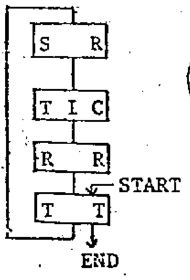


Fig.4 Deductive procedure

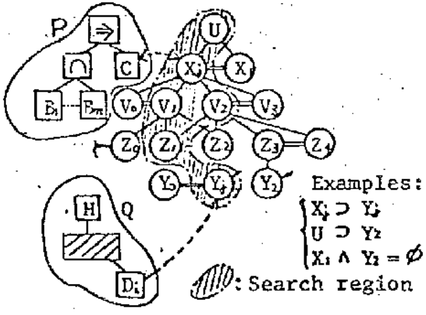


Fig.5 Searching a candidate and its test

3.3.2 Searching the relevant knowledge element

The knowledge element that satisfies the implicative condition to the given query is effectively found in the knowledge base using the conditions in Table 2. Note that the set-theoretical relations in the table are the ones that are used to form the skeleton structure. Suppose some atom Di of the query is selected by SR and it contains a universally quantified variable y_j with the domain Y_j. If there is a knowledge element that satisfies the implicative condition to D_i then its variable x_j at the corresponding position to y_j must also be universally quantified and its domain X_j must include Y_j (Table 2). All formulas satisfying this condition are connected to the node X_j in the skeleton structure, which can be reached by following the inclusion type arcs upwards starting from given Y_j (Fig. 5). Among these formulas, those of which the predicate symbols are the same with that of D_i have possibility to satisfy the implicative condition to D_i. We call such the knowledge elements the candidates to D_i. Since the skeleton structure expands rapidly down-word, the search region is considerably limited.

Next, suppose a candidate is found. Then all the remaining variables besides X_j in this formula must be tested for the implicative condition of Table 2. This time, both the query and the candidate are in hand and therefore, both domains of all corresponding pairs of variables are in hand. Then the set-theoretical relations between domains can be changed to the positional relations between corresponding nodes in the skeleton structure. Its algorithm is so simple that it can be realized in hard wired logic.
The TIC process is equivalent to the unification in the resolution principle.

3.4 RR

Suppose P of the form $B_1 \cap \cdots \cap B_\ell \Rightarrow C$ includes the set of variables, $x = \{x_1 \cdots x_m\}$, of which the domain of each variable x_i is X_i, i=1, 2, ⋯, m. Let the single literal formula composed of the literal C and including only those variables contained in C be denoted $\hat{C}$. Suppose $\hat{C} \Rightarrow \hat{D}_j$ to D_j in Q of the form $D_1 \cap \cdots \cap D_\ell$ and Q includes the set of variables $y = \{y_1 \cdots y_n\}$, $y_j \in Y_j$, j=1, 2, ⋯n. Then a new formula R is generated from P and Q according to the following rules of the RR. First, the matrix of R is obtained by replacing D_j in Q by $B_1 \cap \cdots \cap B_\ell$ of P. R inherits all variables included in P and Q except those which disappear by the replacement operation. Let those variables be divided into two classes: those included in C (and D_j) denoted z_0 and the remainings denoted z_1. Then the quantifier and the domain of each variable in R is obtained as shown in Table 3.

	Q_{pi}	$Q_{qi}(\bar{Q}_{qi})$	$Q_{ri}(\bar{Q}_{ri})$	Domain of z_i
$z_i \in z_0$	∀ ∀ ∃	∀ (∃) ∃ (∀) ∃ (∀)	∀ (∃) ∃ (∀) ∀ (∃)	Y_i $X_i \wedge Y_i$ X_i
$z_i \in z_1$	- -	∀ (∃) ∃ (∀)	∀ (∃) ∃ (∀)	Y_i Y_i
	∃ ∀	- - - -	∀ (∃) ∃ (∀)	X_i X_i

Table 3. Rule for generating new formula

3.5 TT

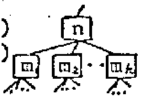
The deductive procedure terminates either when the logical value of the query can be determined or when an AND-OR tree with no NTA is reached. An atom at the leaf of the AND-OR tree is labelled T meaning "TRUE" if it is evaluated as true while it is labelled F meaning "FALSE" if its negation is true. If no knowledge element could be found satisfying the implicative condition to the atom, then this is labelled U meaning "UNKNOWN". Then an intermediate node is evaluated using values of nodes below that by the rule shown in Fig.6. In this figure, the quantifiers at the left side and right side of slant symbol show the cases of the nodes being conjunctive node and disjunctive node respectively. Since the root node representes the query itself, if it is evaluated, then the deductive operation terminates. On the other hand, if there is no NTA remained in the query at all after the finite repetitions of deductive cycle, then the deductive process terminates too. Now, here is a question. Is every deductive process terminates in finite time?

Suppose a sequence of knowledge elements used for a deductive process for a given query is obtained. If, in this sequence, every predicate symbol once appeared in the consequence of a formula never appears in the premise of any of succeeding formulas, then the deductive process terminates in finite time because of the finiteness of the knowledge base. Resultant AND-OR tree with only PTA's corresponds to a program structure and it is not difficult to convert it to the program. To the contrary, if some predicate symbol ever appeared comes again in the succeeding or in the same formula, an endless repetition can happen to be. This is the case of recursive definition of knowledge element. Even in this case, however, if the recursive form is properly defined, the deductive procedure terminates in finite time. As an example, a recursive definition of the factorial function is given as;

(A) (fct 1 0)
(B) (∀x/I)(∀y/I)(∀m/I)[*(eql n 0) *(sub m:n 1)
    *(mult x:y n) (fct y m) ⇒ (fct x n)],

where (fct x n) means "n! ≐ x" and * on the left shoulder of some atoms denotes that these atoms are PTA's. Its deductive process is shown in Fig.7 assuming that 2's factorial is asked by the query. The figure shows how the non-procedural form of the query is converted to the set of PTA's. Thus the deductive procedure acts also as the program generator.

n : node to be evaluated (∪ or ∩)
m_i: node below n (i = 1,2,....,k)



	(n:∩ / n:∪)	(n:∩ /n:∪)
a) if for ∀i	$(m_i=T/m_i=F)$	then (n=T /n=F)
b) if for ∃i,s.t.	$(m_i=F/m_i=T)$	then (n=F /n=T)
c) if for ∀i, for ∃j,s.t.	$(m_i\neq F/m_i\neq T)$ $(m_j=U/m_j=U)$	and then (n=U /n=U)

T: True, F: False, U: Unknown

Fig.6 Evaluation of nodes in AND-OR tree

4. ACCESS TO DATA BASES

4.1 Meaning and description of files

Thus we could realize the path from b_0 to p in Fig.1. Next we will discuss on the relationship between the representation of knowledge and databases. There are a number of different data models proposed within the database technology, each of which has own scheme for representing meaning. Here we will, among them, consider the relational model as an example.

In this model a relation is defined as a set(file) of n-tuple each of which is an instance of objects being in a relation f. Let i-th component in the n-tuple be confined to some set X_i. We say in this case, the i-th attribute of the relation f has the domain X_i. Then an n-tuple is considered as a point, in geometrical sence, in the Cartesian product space $X = X_1 \times X_2 \times$ ---- $\times X_n$.

Suppose the set of all n-tuples in the relation f covers the whole space X. Then the relation f represents the same meaning as the single atom formula,
$(\forall x_1/X_1)\ldots(\forall x_n/X_n)(\ f\ x_1\ldots\ldots x_n)$.
In this case, a relation f in the relational database corresponds to the single atom formula in the logical form.
In general, however, the set is a proper subset of the product space. In this case a special procedural type atom denoted FGET is defined and used to describe the relation. It's format and meaning are,
(FGET $R_i;x_1\ x_2\ \ldots\ x_n$); "a tuple $(x_1\ldots x_n)$ is included in the file R_i (and retrieve it)". By using this,an arbitrary relation R can be defined and written as,
$(\forall x_1/X_1)\ldots(\forall x_n/X_n)[(\text{FGET } R\ ;x_1\ldots x_n) \Rightarrow (f\ x_1\ldots x_n)]$
that is read as "for every tuple $(x_1\ldots x_n)$such that $x_1\in X_1,\ldots,x_n\in X_n$, if it is included in a file R, then it satisfies the relation (f $x_1\ldots x_n$)".

4.2 Basic database operators

A set of primitive operators is defined as is shown in Table 4 for the relational model. Note that the set of primitives are defined differently from that of the conventional DBMSs but the set has the same effect as the latter. For example, the conventional JOIN operation is formed, in this system, as a combination of an AND operation and two MULT operations.

In the ordinary database systems, a user manipulates the databases using the primitive operators. In oder for the knowledge system to be the intelligent interactive front end to the conventional database systems, it must be able to convert user's request represented in the knowledge form to a sequence of defined primitive operators.

```
Q:(∃x/I)(fct x 2)?:"What is 2's factorial?"

[0th] TT: Is there any atom that can be evaluated?
          --- No. Next
[1st] SR: D = Q (Select Q as D)
      TIC: (A) ⇏ D
           RIGHT HAND FORMULA OF (B)
                   (=(∀x/I)(∀n/I)(fct x n)) ⇒ D
           n ← z
      RR: ∼(eql 2 0)∩(sub m; 2 1)∩(mult x; y z)∩
           (fct y m)
       TT: ∼(eql 2 0)Ⓐ T, (sub m; 2 1)Ⓐ T, m ← 1
           then Q → (mult x; y 2)∩(fct y 1)
          (eliminate atoms that have been evaluated)
[2nd] SR: D' = (∃y/I)(fct y 1)
      (Repeat similar procedure to [1st]'s and obtain)
      then Q → (mult x; y 2)∩(mult y; z 1)∩(fct z 0)
[3rd] SR: D" = ( z/I)(fct z 0)
      TIC: (A) ⇒ D", z ← 1, (fct 1 0)Ⓐ T
       TT:  Q → (mult x; y 2)∩(mult y; 1 1)
            END
      where (---)Ⓐ T; (---) is labelled T,
            x ← α ; α is substituted to x,
            Q → (Formula); Q is changed to (Formula)
```

```
(A) (fct 1 0)
(B) (∀x/I)(∀y/I)(∀m/I)(∀n/I)[∼(eql n 0)∩(sub m; n 1)
    ∩(mult x; y n)∩(fct y m) ⇒ (fct x n)
   where (fct x n); "n! = x"
```

Fig.7 Definition of factorial function and its evaluation through deductive operation

4.3 Generation of database access procedures

Suppose the deductive process terminates leaving a formula composed only of PTA's including FGET atom(s). Plural FGETs are then united to a single FCET to which the new file is defined through the set operations to the files in the original FGETs.

Suppose the formula Q is composed of N FGET atoms and represented as
$Q\ ;\ (Q_1x_1/X_1)\ldots(Q_nx_n/X_n)[S\{(\text{FGET } R_j;\ x_{j_1}\ldots x_{jk_j})\}]$,
$(j = 1,2,\ldots,N)$,
where Q_i denotes either ∀ or ∃ and $S\{Z_j\}$ represents a structure of AND-OR tree of this formula consisting of the literals Z_j, $j = 1,\ldots,N$. Then it is transformed to the access procedure to the databases as follows:
(1) First of all, every FGET in the original formula is normalized. Let the set of all variables of this formula and the set of variables in the j-th FGET be $\mathfrak{x} = \{x_1,\ldots,x_n\}$ and $\mathfrak{x}_j = \{x_{j_1},\ldots,x_{jk_j}\}$ respectively. Then j-th file R_j in the j-th FGET is expanded by being multiplied by the domains of variables in $\mathfrak{x} - \mathfrak{x}_j$. It is denoted as $R_j{*} = R_j \times (\mathfrak{X} - \mathfrak{X}_j)$ where $\mathfrak{X}$ and $\mathfrak{X}_j$ are the set of domains of variables in $\mathfrak{x}$ and $\mathfrak{x}_j$ respectively. Then the original (FGET $R_j;\ x_{j_1},\ldots,x_{jk_j}$) is equivalent to (FGET $R_j{*};\ x_1,\ldots,x_n$) in a logical sence provided the ordering of column in R_j is insignificant.
(2) After all FGETs were normalized, they are united to one FCET. A structure of set operations $\hat{S}\{\ \}$ is derived from the structure $S\{\ \}$of the AND-OR tree by first replacing every conjunctive node by the set theoretical intersection and every disjunctive node by the set theoretical union and then replacing every FGET atom at the terminal node by the expanded file $R_j{*}$, $(j = 1,2,\ldots,N)$. Then a sequence of set operations can be obtained from the structure thus obtained and it will produces a file R* defined in the space $X_1\times\ldots\times X_n$. That is, $R^* = \hat{S}\{R_j{*}\} = \hat{S}\{R_j \times (\mathfrak{X} - \mathfrak{X}_j)\}$.

Then,

$(Q_1x_1/X_1)\ldots(Q_nx_n/X_n)[S\{(\mathrm{FGET}\ R_j{}^*;x_j,\ldots,x_{jk_j})\}]$ is logically equivalent to

$(Q_1x_1/X_1)\ldots(Q_nx_n/X_n)(\mathrm{FGET}\ R^*;\ x_1,\ldots,x_n)$------(*A).

(3) Then this formula is transformed to a sequence of the basic operators lisred in Table 4 using the folliwing equations;

$$(Q_1x_1/X_1)\ldots(Q_{n-1}x_{n-1}/X_{n-1})(\forall x_n/X_n)(\mathrm{FGET}\ R^*;x_1,\ldots,x_n)$$
$$=(Q_1x_1/X_1)\ldots(Q_{n-1}x_{n-1}/X_{n-1})(\mathrm{FGET}\ \mathrm{DIV}(R^*\ X_n);x_1,\ldots,x_{n-1})$$
$$(Q_1x_1/X_1)\quad(Q_{n-1}x_{n-1}/X_{n-1})(\exists x_n/X_n)(\mathrm{FGET}\ R^*;x_1,\ldots,x_n)$$
$$=(Q_1x_1/X_1)\ldots(Q_{n-1}x_{n-1}/X_{n-1})(\mathrm{FGET}\ \mathrm{PRJ}(R^*\ X_n);x_1,\ldots,x_{n-1})$$

These equations are applied to the formula until finaly the form of (FGET $\hat{F}$;ϕ) is reached where $\hat{F}$ is a sequence of basic operators applied to R*. $\hat{F}$ has value either 1 or 0 depending on whether the derived sequence of operations ends successfully, that is, ends without having resulted in the empty set (file) or ends unsuccessfully. Here, if $\hat{F}$=1, then the formula (*A) and, therefore, the original query is proved true, while, if $\hat{F}$=0, then it is untrue.

(4) Though $\hat{F}$ gives a mapping from the knowledge element to a database operation, this is not always an optimal one, but it can include redundant file operations and generate ineffective intermidiate files.

It is, therefore, optimized. The optimizing rules listed in Table 5 are used. These rules exchange the oder of operators in F to minimize the ineffective operations and are equivalent to that of [16].

Example

$$Q=(\forall x/X)(\exists y/Y)(\forall z/Z)[(\mathrm{FGET}\ F_1;\ x\ y)\cap(\mathrm{FGET}\ F_2;\ y\ z)]$$
$$=(\forall x/X)(\exists y/Y)(\forall z/Z)[(\mathrm{FGET}\ F_1\times Z;\ x\ y\ z)\cap(\mathrm{FGET}\ F_2\times X;\ x\ y\ z)]$$
$$=(\forall x/X)(\exists y/Y)(\forall z/Z)[(\mathrm{FGET}\ F_1\times Z\wedge F_2\times X;\ x\ y\ z)]$$
$$=(\mathrm{FGET}\ \mathrm{DIV}(\mathrm{PRJ}(\mathrm{DIV}(\mathrm{AND}(\mathrm{MULT}(F_1\ Z)\mathrm{MULT}(F_2\ X_1))Z)Y)X);\phi)$$
$$=(\mathrm{FGET}\ \mathrm{DIV}(\mathrm{PRJ}(\mathrm{AND}(F_1\ \mathrm{MULT}(X\ \mathrm{DIV}(F_2\ Z)))Y)X);\phi)$$

(5) In real problems, both FGET type atoms and ordinary PTAs may be included in an AND-OR tree. Then ordering of atoms is necessary because a PTA can be evaluated when a value is assigned to every independent variable and, if some inedpendent variable of a PTA is also the dependent variable of another PTA, then the latter PTA must precede to the former.

Note here that to execute a FGET type atom has the same effect as to assign values to variables x_i ,----,x_i included in the FGET. We call such the variables the evaluated variables. A PTA of which every independent variable is the evaluated variable can be evaluated. Inversely, it is possible to select minimum set of FGETs in the AND-OR tree such that the union of the set of all variables included in them and the set of already evaluated variables include the set of all independent variables of some PTA(s). To such the set of FGET atoms, the procedure (1) through (4) above are applied and to the end of the subsequence of file operations thus obtained the PTA(s) is appended. After then the set of the variables in those FGETs and the dependent variables of the PTA(s) are added to the set of evaluated variables. This process is repeated until all atoms are transformed to the sequence of operations.

(6) When the programming stage ends, the system enters the execution stage. Some variable(s) will be used to generate the answer and, therefore, must be saved to the end of the execution of the program. However, both DIV and PRJ erase, by definition, not only all the tuples that do not satisfy the conditions shown in Table 4 but also the key variables (and their values) irrespective of the variables being asked by the query or not, where by the key variable we mean the variables used as the divider in the DIV and the one used for projection in the PRJ. In order to save these variables, another set of oerators denoted as VPIV and VPRJ are provided. These are defined similarly to DIV and PRJ respectively except they do not erase the key variables but give them a special flag.

At the programing stage each variable is checked and determined if its values are to be saved or they can be erased at the execution stage. There are two cases of a variable to be saves. If it is included in the query then it should be saved until the answer is generated and (2) if a variable is included in some PTA as the independent variable but is not directly included in the query, then it is saved until the PTA is fabricated in the generated program.

5. SIMPLE EXAMPLE

The system thus obtained satisfies the basic requirements for it being an intelligent interactive system. To show how the system works, we will present a very simple example.

Suppose a set of knowledge elements(1) through (5) shown in Fig.8 exists in the knowledge base,where (1)and (2)represent facts such as "everything lighter than fluid float on it "and" the specific gravity of an object can be obtained by dividing the weight of the object by its volume" respectively.

The system is assumed to have three different files representing the relations holding between "sample

i) $\mathrm{AND}(F_1,F_2) = \{(x_1^i \ldots x_n^i)/(x_1^i \ldots x_n^i)\in F_1\wedge F_2,\ i = 1,2,\ldots\}$

ii) $\mathrm{OR}(F_1,F_2) = \{(x_1^i \ldots x_n^i)/(x_1^i \ldots x_n^i)\in F_1\vee F_2,\ i = 1,2,\ldots\}$

iii) $\mathrm{MULT}(F_1,\ Y) = \{(x_1^i \ldots x_n^i, y^j)/(x_1^i \ldots x_n^i)\in F_1, y^j\in Y,\ i = 1,2,\ldots,\ j = 1,2,\ldots\}$

iv) $\mathrm{DIV}(F_1,x_k) = \{(x_1^i \ldots x_{k-1}^i, x_{k+1}^i \ldots x_n^i)/(x_1^i \ldots x_n^i) \times X_k\in F_1,\ i = 1,2,\ldots\}$

$\mathrm{DIV}(\hat{X}_k,X_k) = 1$ (if $\hat{X}^k\supset X^k$) or 0 (if $\hat{X}_k\not\supset X_k$)

v) $\mathrm{PRJ}(F_1,X_k) = \{(x_1^i \ldots x_{k-1}^i, x_{k+1}^i \ldots x_n^i)/(x_1^i \ldots x_n^i)\in F_1,\ i = 1,2,\ldots\}$

$\mathrm{PRJ}(\hat{X}_k,X_k) = 1$ (if $\hat{X}_k\wedge X_k\neq\phi$) or 0 (if $\hat{X}_k\wedge X_k=\phi$)

vi) $\mathrm{CLP}(F_1,X_k) = \{(x_1^i \ldots x_k^i \ldots x_n^i)/(x_1^i \ldots x_k^i \ldots x_n^i)\in F_1,\ x_k^i\in X_k,\ i = 1,2,\ldots\}$

vii) $\mathrm{EXPD}(F_1,X\ ) = \{(x_1^i \ldots x_n^i\ x^i)/(x_1^i \ldots x_n^i)\in F_1, x^i\in X,\ i = 1,2,\ldots\}$

where

F_1 ; $\{(x_1^i \ldots x_n^i)/x_r^i\in X_r,\ 1\leq r\leq n,\ i = 1,2,\ldots N_1\}$

F_2 ; $\{(x_1^i \ldots x_n^i)/x_r^i\in X_r,\ 1\leq r\leq n,\ i = 1,2,\ldots,N_2$

Table 4 Primitive database operations

(1) $\mathrm{DIV}(\mathrm{MULT}(X,F),Y) = \begin{cases}\mathrm{MULT}(X,\mathrm{DIV}(F,Y)), & \text{if } X\neq Y\\ F, & \text{if } X = Y\end{cases}$

(2) $\mathrm{DIV}(\mathrm{AND}(F_1,F_2),X) = \mathrm{AND}(\mathrm{DIV}(F_1,X),\mathrm{DIV}(F_2,X))$

(3) $\mathrm{PRJ}(\mathrm{MULT}(X,F),Y) = \begin{cases}\mathrm{MULT}(X,\mathrm{PRJ}(F,Y)), & \text{if } X\neq Y\\ F, & \text{if } X = Y\end{cases}$

(4) $\mathrm{PRJ}(\mathrm{OR}(F_1,F_2),X) = \mathrm{OR}(\mathrm{PRJ}(F_1,X),\mathrm{PRJ}(F_2,X))$

(5) $\mathrm{DIV}(\mathrm{OR}(F_1,\mathrm{MULT}(X,F_2)),X) = \mathrm{OR}(\mathrm{DIV}(F_1,X),F2)$

(6) $\mathrm{PRJ}(\mathrm{AND}(F_1,\mathrm{MULT}(X,F_2)),X) = \mathrm{AND}(\mathrm{PRJ}(F_1,X),F2)$

(7) $\mathrm{AND}(\mathrm{MULT}(X,F_1),\mathrm{MULT}(X,F_2)) = \mathrm{MULT}(X,\mathrm{AND}(F_1,F_2))$

(8) $\mathrm{OR}(\mathrm{MULT}(X,F_1),\mathrm{MULT}(X,F_2)) = \mathrm{MULT}(X,\mathrm{OR}(F_1,F_2))$

Table 5 Optimizing rules

(material) number and its (measured) weight","sample (material) number and its (measured) volume" and sample (liquid) number and its specific gravity" respectively. The formula (3) through (5) are descriptions of these files in the logical form. These are a portion of system's knowledge and user is not required to be aware of them. He can only ask the system like "Which sample material floats on all liquid sample?" (Q in Fig.8). Since there is no such data file (description) that contains data to reply this query directly, the system try to resolve the query into another formulas with equivalent meaning. "float on" is equivalent to a certain relation to hold between the specific gravities of two objects (Q1). The system tries to find answer to this new query but fails again because there is no data file on the specific gravity of the sample materials. Therefore, it resolves the query again and replaces the specific gravity of the sample material by its equivalent definition, i.e., the relation between the weight and the volume of the materials (Q2). This time, both weight and volume of materials are given in the files, and Q3 is obtained. Since Q3 contains no NTA at all, the deductive process terminates and a program including the file manipulations is obtained. This is converted to the database operations.

Fig.9 shows the results of this example obtained by our experimental system. This result is optimal in the sense that no ineffective file operation is achieved.

6 CREDIBILITY OF INFORMATION

In many cases, information in the real world contains ambiguity to some extent and also is not fully credible. In the KAU system, ambiguity is considered as a kind of incredibility. To utilize incredible information, its credibility must be evaluated.

There are two lines of approaches to evaluate the credibility. In the first approach the logical value of information and its credibility is inseparable and are evaluated at the same time. The multi-valued logic, the Fuzzy logic and so on belong to this class of approaches. The second method evaluates the credibility of information independent of logical value. The probability method belongs to this approach. According to the probability theory, a probability is defined to each (logical) event which is obtained as a consequence of logical operation. The KAU system adopts the latter approach because the former approach is anticipated to cause the system unnecessarily complicated. With the probability method, every logical formula is considered as an event and is assigned a real value in $[0,1]$ as a credibility of the formula. The rule of evaluating the credibility decrement caused by the change of the formula at the deductive operation is defined. There are at least four causes of deteriorating the credibility for information. These are;

(1) incredibility of knowledge element itself because of ambiguity in the knowledge source,
(2) ambiguity implied in the conditional type knowledge element, that is, the uncertainty contained in B in $A \Rightarrow B$ even if A is certain,
(3) fuzziness existing in the set theoretical inclusion, $x \in X$, and
(4) error contained in each data in databases such as observation error.

Knowledge:

$$(\forall x/P)(\forall y/F)(\forall z/R)(\forall u/R)[(\text{spc-grv } x\ z) \cap (\text{spc-grv } y\ u) \cap (\text{less-than } z\ u) \Rightarrow (\text{float } x\ y)] \quad (1)$$

$$(\forall x/P)(\forall y/R)(\forall z/R)(\forall u/R)[(\text{mass } x\ y) \cap (\text{volume } x\ z) \cap (\text{div } u;\ y\ z) \Rightarrow (\text{spc-grv } x\ u)] \quad (2)$$

$$(\forall x/P)(\forall y/R)[(\text{FGET } F_m;\ x\ y) \Rightarrow (\text{mass } x\ y)] \quad (3)$$

$$(\forall x/P)(\forall y/R)[(\text{FGET } F_v;\ x\ y) \Rightarrow (\text{volume } x\ y)] \quad (4)$$

$$(\forall x/L)(\forall y/R)[(\text{FGET } F_l;\ x\ y) \Rightarrow (\text{spc-grv } x\ y)] \quad (5)$$

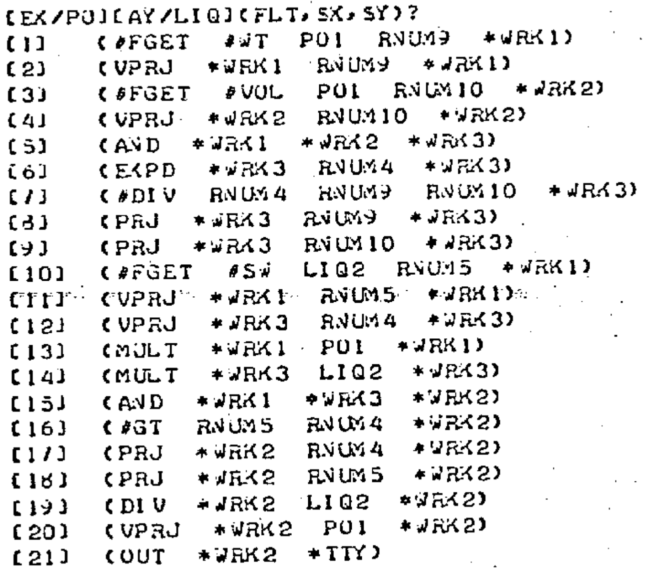
$$Q\text{: } (\exists x/P)(\forall y/L)(\text{float } x\ y)\ ;\ \text{"which material floats on all liquid samples"} \quad (6)$$

$$Q_1\text{: } (\exists x/P)(\forall y/L)(\exists z/R)(\exists u/R)[(\text{spc-grv } x\ z) \cap (\text{spc-grv } y\ u) \cap (\text{less-than } z\ u)] \quad \text{(from (1) and Q)}$$

$$Q_2\text{: } (\exists x/P)(\forall y/L)(\exists z/R)(\exists u/R)(\exists v/R)(\exists w/R)[(\text{mass } x\ z) \cap (\text{volume } x\ u) \cap (\text{div } v;\ z\ u) \cap (\text{spc-grv } y\ w) \cap (\text{less-than } v\ w)] \quad \text{(from (2) and } Q_1)$$

$$Q_3\text{: } (\exists x/P)(\forall y/L)(\exists z/R)(\exists u/R)(\exists v/R)(\exists w/R)[(\text{FGET } F_m;\ x\ z) \cap (\text{FGET } F_v;\ x\ u) \cap (\text{div } v;\ z\ u) \cap (\text{FGET } F_l;\ y\ w) \cap (\text{less-than } v\ w)] \quad \text{(from (3), (4), (5) and } Q_2)$$

Fig.8 An Example of query processing



```
[EX/PO][AY/LIQ](FLT,SX,SY)?
[1]    (#FGET   #WT   PO1   RNUM9   *WRK1)
[2]    (VPRJ   *WRK1   RNUM9   *WRK1)
[3]    (#FGET   #VOL   PO1   RNUM10   *WRK2)
[4]    (VPRJ   *WRK2   RNUM10   *WRK2)
[5]    (AND   *WRK1   *WRK2   *WRK3)
[6]    (EXPD   *WRK3   RNUM4   *WRK3)
[7]    (#DIV   RNUM4   RNUM9   RNUM10   *WRK3)
[8]    (PRJ   *WRK3   RNUM9   *WRK3)
[9]    (PRJ   *WRK3   RNUM10   *WRK3)
[10]   (#FGET   #SW   LIQ2   RNUM5   *WRK1)
[11]   (VPRJ   *WRK1   RNUM5   *WRK1)
[12]   (VPRJ   *WRK3   RNUM4   *WRK3)
[13]   (MULT   *WRK1   PO1   *WRK1)
[14]   (MULT   *WRK3   LIQ2   *WRK3)
[15]   (AND   *WRK1   *WRK3   *WRK2)
[16]   (#GT   RNUM5   RNUM4   *WRK2)
[17]   (PRJ   *WRK2   RNUM4   *WRK2)
[18]   (PRJ   *WRK2   RNUM5   *WRK2)
[19]   (DIV   *WRK2   LIQ2   *WRK2)
[20]   (VPRJ   *WRK2   PO1   *WRK2)
[21]   (OUT   *WRK2   *TTY)
```

Fig.9 Result of example problem

The ambiguity implied in the conditional knowledge of (2) above is defined as the conditional probability and denoted P(B/A). Then, if the credibility of A alone is P(A), then that of B alone, P(B), is obtained by the rule of conditional probability as P(A)·P(B/A). The fuzziness of (3) is just the definition of the membership function in the fuzzy set theory. Though it is not the same as the probability in the strict sense, we regard them the same and apply the probability theory as an approximation.

Then the credibility of the given query is evaluated in parallel with the evaluation of its truth value in such a way that, each time information in the system is used in the deductive process, the credibility of the information is multiplied to update the intermediate value of the credibility of which the initial value was one. When the logical value of the query is determined, the current intermediate value gives its credibility.

7. EXTENSION OF FORMALISM

The first order logic is very suitable for representing knowledge, in partisular, for defining conversions, K-K, K-P and K-D in Table 1, as has been shown up to now. Its expressive power, however, is not good enough to allow rich class of the external representations. So we need to expand it. We first define the minimal set of allowable external expressions denoted here E* as follows;

(1) E* should be large enough so that every expression necessary for representing knowledge for intended application is to be included, and
(2) E* should be such that any user can easily understand without any special training how to express his knowledge or query by using it.

The fact that most knowledge element can be expressed in natural language tells us that system should be able to accept all basic syntactic patterns of language. Since every complex sentence can be analyzed to the logically connected set of basic sentences, the necessary condition of the system for representing knowledge is that the system can accept basic patterns of sentences and phrases.

In case of English, the simple sentences can be classified into about twenty-five classes of different patterns. Only one-third of those patterns can be represented in the framework of the first order predicate.

To expand the framework for representing the rich class of knowledge element, we extend the definition of atoms. So far, an atom is defined as an ordered set of simple variables preceded by the predicate symbol. The first extention is to allow the predicate symbol itself to be a variable. By this extention, a query of which the predicate symbol is the interrogated term can be represented. An example is " In what relation is A to B?" represented in a form (∃x/RELATION)(x A B)?.

The second extension is to allow such an atom of which some positions are occupied not by simple variables but by some formulas. That is, the atom of the form (Variable$_0$, Variable$_1$/(Formula)$_1$,...,Variable$_n$/(Formula)$_n$) is allowed where Variable$_i$/(Formula)$_i$ means that either a simple variable or a formula can occupies i-th position. An example is (∃x/PERSON)(know #Tom ((sick x)∩(family x #Jim)));"Tom knows that someone of Jim's family is sick.".

Accompanying to these extensions in the formalism, the processing algorithm must also be modified. Especially to modify the deductive algorithm is inevitable. This modification in algorithm is not so difficult. But its foundation is on a deep theoretical consideration on the structure of the world treated in the system. Since the topics is too specific for the purpose of the paper, we only mention here that, according to the extention of the formalism, the TIC of the deductive algorithm is mainly affected. Let
P;$(Q_{pi_1} x_{pi_1}/X_{pi_1})\cdots(Q_{pi_n} x_{pi_n}/X_{pi_n})(x_1 x_2 \cdots x_m)$ and
Q:$(Q_{qj_1} y_{qj_1}/Y_{qj_1})\cdots(Q_{qj_n} y_{qj_n}/Y_{qj_n})(y_1 y_2 \cdots y_m)$.
Where each of $x_1,\ldots,x_n$, $y_1,\ldots,y_n$ is either a simple variable or a formula. As a formula includes more than one variables, n is equal to or larger than m. Then the rule (1) in Sec. 3.3.1 for the condition of P⇒Q is modified to;

(1') for every corresponding pair $(x_i\ y_i)$, either the condition of Table 2 is satisfied or the implicative condition $(x_i)\Rightarrow(Y_i)$ holds depending on both x_i and y_i being the simple variables or the inner formulas.

To realize this algorithm, the control structure of the deductive procedure is modified so that the TIC is recursively applied to the nest form of representation.

8. CONCLUSION

The outline of the first version of the KAUS(Knowledge Acquisition and Utilization System) has been presented This system is designed to bridge over the gap between man and computer and to aid man in his intelligent works. In this system, information is processed by being transformed among four different forms; external representation, program, databases and knowledge representation. The mappings between different forms and the deductive procedure to the representation of knowledge have been designed.

An experimental system has been implemented, which realized some important basic procedures as has been described in Sec.2 through Sec.6. The procedure to evaluate the credibility of information and the expansion of the framework are not realized yet but is in planning stage.

REFERENCES

[1] Bobrow, D.C.(chaired) "A Panel on Knowledge Representation "In Proc. 5th IJCAI,1977, pp983-992.

[2] Chang,C.L. $ Lee, R.C.T. "Symbol Logic and Mechancal Theorem Proving" Academic press, 1973.

[3] Codd,E.F. "Recent Investigations in Relational Data Base System", Proc. IFIP Congress, Aug. 1974.

[4] Date C.J. "An Introduction to Database Systems", Addison-Wesley, 1975.

[5] Emden Van "Programming with Resolution Logic", Macgine Intelligence 8, pp266-299.

[6] Feigenbaum E.A. "The Art of Artificial Intelligence, Themes and Case Studies of Knowledge Engineering, 5th IJCAI pp1014-1029.

[7] Green C. "The Use of Theorem Proving Techniques in Question Answering System", Proc. 23-rd Nat. Conf. ACM, 1968.

[8] Huet G.P. "A Mechanization of Type Theory" 3rd IJCAI 1973: pp139-146.

[9] Kellog C.H. & Klahr P. "Adding a Deductive Capability to a Data Management System", Second VLDB Sept. 1976.

[10] Kowalski R.A. "Predicate Logic as Programming Language", Proc. IFIP-74. pp569-574.

[11] Krivine J. "introduction to Axiomatic Set Theory", D.Reidel pub. Co. 1971.

[12] Ohsuga S. "Semantic Information Processing in Man-Machine Systems", Proc. 1977 IEEE Conference on Decision & Control. pp1351-1358, Dec. 1977.

[13] Ohsuga S. "Towards Intelligent Interactive Systems", Proc. the IFIP W.G.5.2 Workshop Ssilac on Methodorogy of Interaction (to apper) North-Holland Pub. Co. 1979.

[14] Robinson J.A. "A Machine Oriented Logic Based on the Resolution Principle", JACM Vol. 12 Jan. 1965.

[15] Shapiro S.C. "Path-Based and Node-Based Inference in Semantic Networks", TINLAP-2. Jan. 1978 pp219-225.

[16] Smith J.M. and Chng P.Y.T. "Optimizing the Performance of a Relational Algebra Databases Interface", CAMA vol.18.No.10, 1975, pp568-579.

於 "JIPDEC Information Systems Seminar on Semantic Aspects of Databases, JIPDEC, Tokyo, Feb. 21-22, 1980."

拡張仮想関係と知識フィルタの演繹的質問応答への応用について

An Application of an Extended Virtual Relation and a Knowledge Filter to Deductive Question-Answering

1980年2月21日(木)

富士通(株)国際情報社会科学研究所
国藤進, 若木利子.

〔目次〕

1. まえがき p.2

2. 問題提起 p.5

2.1 内包と外延 p.5
2.2 問合せの評価 p.6
2.3 システム構成の概略 p.7

3. 従属関係 p.9

3.1 1階多類論理の構文法 p.9
3.2 従属関係の定義 p.10
3.3 条件付従属関係 p.12

4. 拡張仮想関係 p.14

4.1 仮想関係の解釈 p.14
4.2 仮想関係のHorn節による定義と有向連結グラフ p.15
4.3 拡張仮想関係の形式 p.17
4.4 拡張仮想関係と書換え規則 p.18
4.5 長所と応用 p.19

5. 知識フィルタ切断法 p.20

5.1 全称束縛の自由化による知識フィルタ切断法 p.20
5.2 従属関係型知識フィルタ p.21
5.3 視座濾過法 p.22
5.4 従属関係の推論構造 p.23
5.5 基本システム構想 p.24

6. 応用例 p.26

6.1 FD型知識フィルタ p.26
6.2 MVD型知識フィルタ p.27
6.3 帰納的に定義された拡張仮想関係 p.28

7. あとがき p.30

〔謝辞〕 p.31
〔参考文献〕 p.32

1. まえがき

情報システム形成の基本的要請として，北川[1,2]は対象化・作用化・社会化の3つの観点があることを指摘している．この考え方により，人間機械系を情報処理過程[1,2]の中で位置付けると，蓄積された情報を検索・流通・利用するという段階は，情報の作用化過程とみなすことができる．データベース・システムの発展につれて，この作用化過程の活性化の方式の確立が望まれている．そのような方式のひとつとして，Feugenbaum[3]を中心とする知識工学専門家は，データベース（DB: Data Base）の高度な利用を促進するため知識ベース（KB: Knowledge Base）を設置すべきであるという見解に基づく知識工学方法論を展開している．このKBとDBの機能分化というデータベース・システムの設計方針は，大須賀のKAUS[4,5]やReiterのシステム[6]といった推論機能をもつ演繹的質問応答（QA: Question-Answering）システム，あるいはShortliffeのMYCIN[7]やWeiss & KulikowskiのEXPERT[8]といった診断支援機能をもつ医療相談システムの実働化指針に，深い影響を与えつつある．

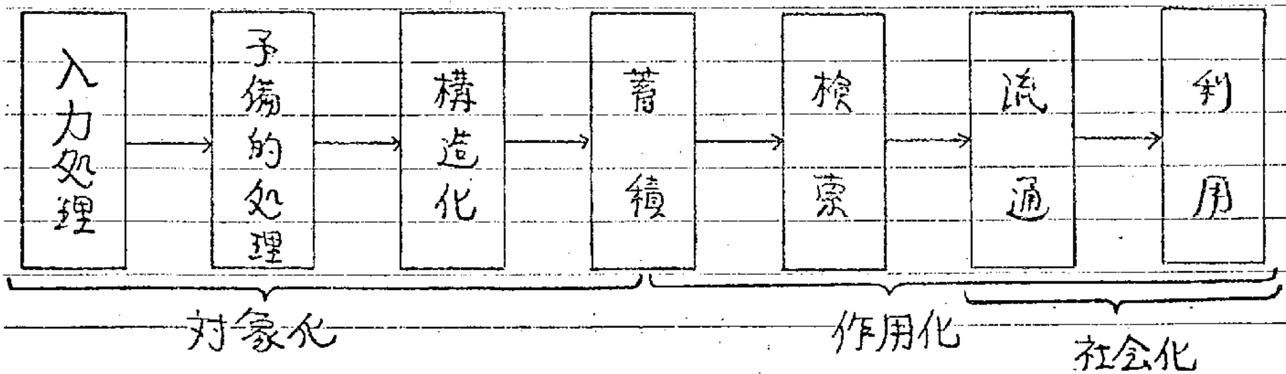


図1.1　北川の情報処理過程論[2]

さてKBとDBの機能分化の必要性は，従来から北川[9]によって，ブレーンウェア(1)とブレーンウェア(2)（BW: Brainware）という概念によって喚起されていた．そこにおいて，BW(1)は“方法資料，概念資料，モデル資料，文法資料”であり，BW(2)は“データベース，データ辞典，研究者用ファイル”

であると例示的に説明されている．その限りでは従ってBW(1)とBW(2)は，それぞれ概ねKBとDBに対応する概念であると理解される．

ところでわれわれは，KBおよびDBに格納されている知識に関して，一階述語論理のみを使用するという制約を設けて考察する．その理由は計算機処理システムの現状にかんがみて，ある知識が充足不能かどうかを定める部分的決定手続きが確立しているからである．同様の制約の下にReiter[6]は，一階述語論理の特殊なクラスである一階多類論理(1MSL：1st-order Many-Sorted Logic)の見地から，IDB(Intensional Data Base)とEDB(Extensional Data Base)の両者からなる演繹的QAシステムの関係データベース・システム設計方法論について考察している．またChang[10]も，関係データベース向きの問合せ言語DEDUCE 2の中で，基底関係(BR：Base Relation)から仮想関係(VR：Virtual Relation)を定義する方法を提案している．ここにIDBとEDB，VRとBRは，それぞれ一階述語論理の命題(wff：well-formed formula)のみを知識とする特殊なKBとDBである．ReiterおよびChangにおいては，KB，DB，および問合せの否定のwff表現が全てHorn節である場合について理論的考察を展開している．

一方上述の研究とは独立に，関係データベース管理システムの設計方法論に関連する多くの研究が成されている．それらの成果を概括し，本来KBに格納すべき知識を体系的に整理し，それらを述語論理での用語法にならって公理(axioms)と呼ぶことにする．このような公理としては，統合性制約や視座定義が知られている．この内前者の一種である従属関係(D：Dependency)については，Codd[11]やFagin[12]やNicolas[13]らにより，運用論的側面である更新不整合排除・冗長性除去・情報無損失分解可能性という三つの観点から，関係データベースの分解・合成への影響がスキーマ設計法として克明に調べられている．また後者については，田中[14]，増永[15]，上林[16]，および筆者[16,17]らにより，意味論的側面である個別End Userの視点の反映とかアクセス・パス決定への利用等についての研究が行われている．

以上の調査をもとに，本小論では演繹的QAシステムに発せられる問合せのある種のものは，統合性制約と視座定義の両者の知識を用いて解答抽出した方が，システム構成上簡潔であり，かつ性能上有利であることに注目する．すなわち，筆者らが文献[16, 17]で述べた考え方を一般化し，統合性制約として既知の各種従属関係を，知識フィルタ（KF: Knowledge Filter）として利用する方法を述べる．

まず与えられた基底関係達の自然結合に対して，ある知識フィルタを仮説として設定する．次にこの知識フィルタにより，それらの自然結合をフィルタリングされた関係の実例(instance)とフィルタリングされなかった関係の実例とに直和分割する．このことは，基底関係達の自然結合を，ある仮説を満足するものの全体と満足しないものの全体とに直和分割し，演繹的QAにおける解答抽出に利用する方法を提案したことに相当する．本小論で展開する具体的手法は，知識フィルタという考え方を，Changの仮想関係[10]を拡張した拡張仮想関係の意味記述に反映させて，それを個別End Userの，前処理[16, 17]でなく実時間処理での照会に利用可能とした点に特徴がある．

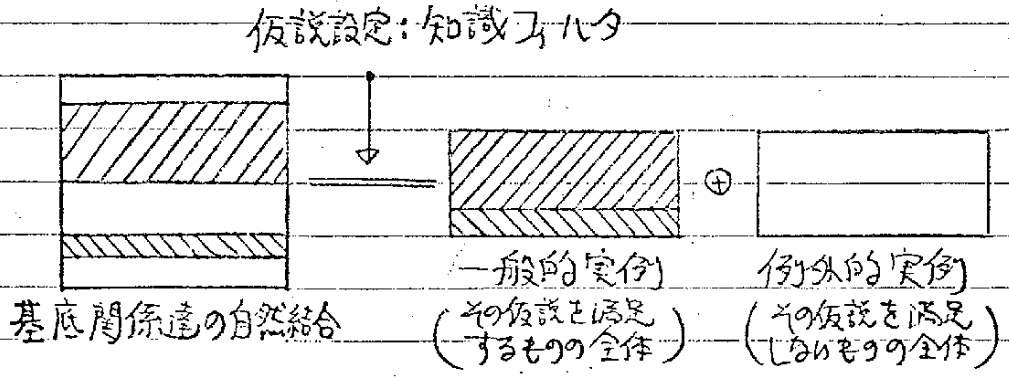

仮説設定：知識フィルタ
⊕
基底関係達の自然結合
一般的実例
（その仮説を満足するものの全体）
例外的実例
（その仮説を満足しないものの全体）

図1.2　知識フィルタによる切断

2. 問題提起

2.1. 内包と外延

DBを意味論的見地でとらえた時，関係の特性は，DB研究者であるNicolas, Minker, Reiter, Changら[18]が指摘するように，内包と外延の2側面で把握される．関係の内包とは，その関係に固有の不変(invariant)な意味的制約であり，関係の外延とは，その関係の実例，即ちタップルの集合である．

一方DBを古典(二値)論理で解釈すると，タップル$\bar{x}$がある関係Rに属することが述語$R(\bar{x})$に真(true)の値Tを割当てることに，タップル$\bar{x}$が関係Rに属さないことが述語$R(\bar{x})$に偽(false)の値Fを割当てることに対応する．この時外延は古典論理でのモデルを規定する．あるいは関係の外延とは，定数記号のみを引数とする正の単位リテラル(positive unit literals)のみからなる節集合のことである．また関係の内包とは，解釈が変化しようとも，そのようなモデル上で，常に真となる公理によって規定される．このような公理は，しばしばthe proper (or non-logical) axioms [19]と呼ばれる．関係の内包と外延をこのように定めた時，関係の更新は，モデルの変化に相当する．Reiterら[6]にならい，関係の外延にあたるタップルの集合をEDB，関係の内包にあたる公理の集合をIDBと呼ぶことにする．

さてわれわれは仮説検定に関する北川の形式化[28]，すなわち一般前提と仮説との区別に従い，しかも述語論理での用語をここでも用いて，IDB内の公理を次の2つに区分する：

(1) 一般前提的公理(general assumptive axioms) —— DBの実例が更新しても常に成立する，そのDBの族の普遍的枠組を表明する公理．(例) 統合性制約．

(2) 仮説的公理(hypothetical axioms) —— そのDB上で検索対象としたいデータの束を仮説としてうちたて概念規定する公理．(例) 仮想関係定義，視座定義．

ここに視座定義とは，個々のEnd Userが独自の視座で行う，そのDB上で検索対象としたりデータの束(bundle)の切出し方を概念規定する公理のことである.

(1)(2)は「内包に追加しうる公理は，一般前提的公理に矛盾しない仮説的公理のみ許容される」というDBの運用論的要請により，概念的に区別される.

2.2. 問合せの評価

関係データベース上の演繹的QAシステムにおける問合せ評価法を，Chang[10]はnon-evaluational approachとevaluational approachに分類している.

non-evaluational approachでは，通常の定理の自動証明ツール(theorem prover)を用い，wffで表現された問合せが，EDBとIDBに属する公理の論理的帰結(logical consequence)であることを示すことにより，問合せを評価する.

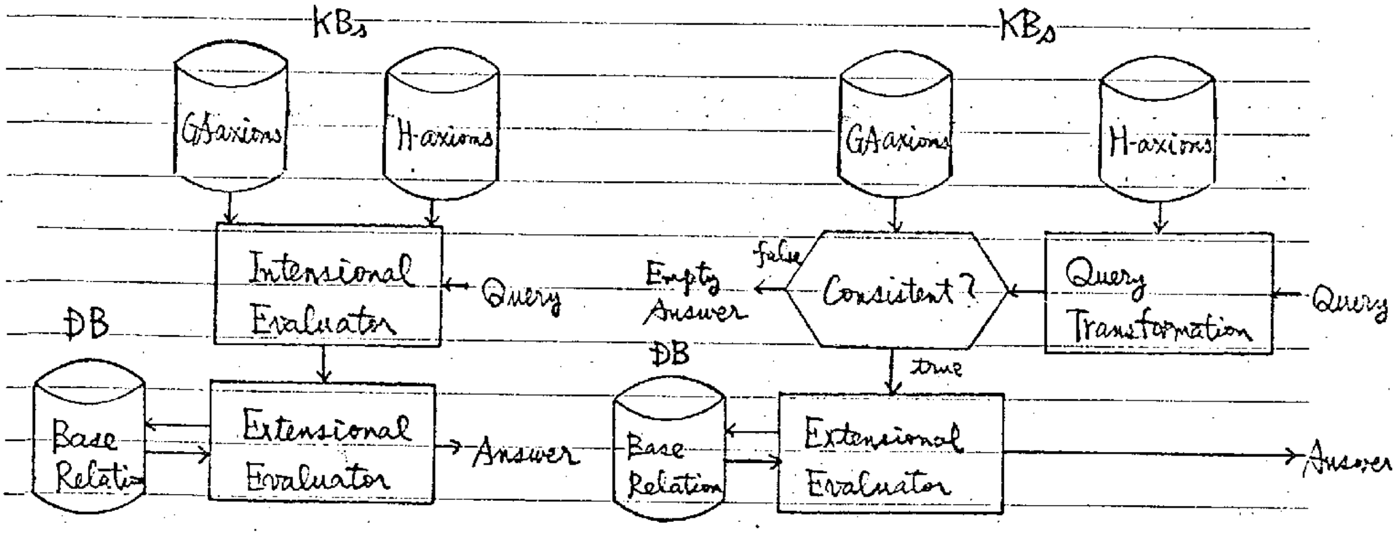
一方evaluational approachはReiterやChangらにより採用されているが，彼らのシステムは，EDBを検索するExtensional EvaluatorとIDBを検索するIntensional Evaluatorに完全に機能分割しているのが特徴である. Intensional Evaluatorの主な機能はtheorem proverであり，Extensional Evaluatorの主な機能は通常のデータベース管理システムである.

このことは巨大データベースにおけるEDBの情報が，IDB内の公理よりはるかに多く，そのようなEDBをtheorem proverで検索し統一(unify)するよりは，データベース管理システムでEDBを検索する方が効率がよいことによる. ただしtheorem proverでEDBを統一していく場合，定数$\vec{c}$および基底関係$\mathcal{R}$に対応する述語定数記号$P_{\mathcal{R}}$から成る素命題(atf: atomic formula) $P_{\mathcal{R}}(\vec{c})$が真か否かを評価するのに対して，データベース管理システムでは関係$\mathcal{R}$にタップル$\vec{c}$が属するか否かを評価する違いがある.

特にChangのシステムでは，その外延は定義されないがIDBの公理により定義される仮想関係を想定している．まずIntensional Evaluatorに相当するメカニズムが，仮想関係を含む問合せを，基底関係のみを含む問合せに変形する．更にこれをSEQUELのような情報検索言語に変換し，System R [20] あるいはRDB/V1 [21] のような関係データベース・システムにより高速な問合せ評価を行い，必要な解答を抽出していく．

2.3. システム構成の概略

本小論でもevaluational approachをとるが，従来のシステムとわれわれが提案するシステムとの相異を，システム構成の概略として述べてみよう．



（本システム）　　（従来のシステム）

図2.1　Query Process

従来のシステム[6,10]では，問合せを仮説的公理を用いて変形し，プランと呼ばれる基底関係のみよりなる問合せを順次生成した後，各プランが一般前提的公理と整合的かどうかを判断し，通常の情報検索言語による基底関係の評価を行っていた．これに対しわれわれが考察するシステムは，問合せが一般前提的公理から推論されない場合のみ，仮説的公理を用いての問合せ変形を行い，プランの順次生成を行い，各プランの一般前提的公理との整合性検査および情報検索言語による基底関係の評価を行う．このようなシステム構成は，一般前提的公理から問合せ（あるいはその否定）が推論される場合，無駄な問合せ変形を防ぐので，問合せの評価過程のコストが大巾に軽減する場合がある．この評価過程の詳細については，5章で説明する．

3. 従属関係

3.1. 1階多類論理の構文法

1階多類論理（1MSL：1st-order Many-Sorted Logic）の通常の表記法[6]に従い，wffの構文法を次のように定める．

まず個体定数を $c_1, c_2, \cdots, c_p$ で，個体変数を $x_1, x_2, \cdots$ で，結合記号を ∧（連言 and），∨（選言 or），∼（否定 not），⊃（含意 implies），および ≡（等価 equivalence）で表わす．
また1MSLの1引数述語定数に関係データベース・モデルの定義域（domain），n引数述語定数（$n \geqq 2$）にn項関係を割当て，前者をτで，後者を $P, Q, \cdots, =$（等号）で表記することにする．定義域は単純タイプ（simple type）あるいは類（sort）と呼ばれることもある．

［定義1］　項（term）
個体定数および個体変数は項である．

［定義2］　定義域句（domain literal）
τが定義域でtが項の場合，$\tau(t)$ および $\sim\tau(t)$ は定義域句である．

［定義3］　句（literal）
Pがn項関係で，$t_1, t_2, \cdots, t_n$ が項の場合，$P(t_1, \cdots, t_n)$ および $\sim P(t_1, \cdots, t_n)$ は句である．

［定義4］　定義域命題（τ-wff：domain well-formed formula）
(i) 定義域句はτ-wffである．
(ii) W_1 と W_2 がτ-wffsならば，$(\sim W_1)$, $(W_1 \wedge W_2)$, $(W_1 \vee W_2)$, $(W_1 \supset W_2)$, および $(W_1 \equiv W_2)$ もτ-wffsである．
(iii) xが個体変数でWがτ-wffならば，$((\forall x)W)$ や $((\exists x)W)$ もτ-wffである．

〔定義5〕　命題（wff : well-formed formula）

(i) 句は wff である．

(ii) W_1 と W_2 が wffs ならば，$(\sim W_1)$，$(W_1 \wedge W_2)$，$(W_1 \vee W_2)$，$(W_1 \supset W_2)$，および $(W_1 \equiv W_2)$ も wffs である．

(iii) x が個体変数，τ が定義域，および W が wff であれば，$((\forall x)(\tau(x) \supset W))$ および $((\exists x)(\tau(x) \wedge W))$ も wffs である．

ここに wff の〔定義5〕(iii) は，しばしば $(\forall x/\tau)W$ あるいは $(x/\tau)W$，および $(\exists x/\tau)W$ あるいは $(Ex/\tau)W$ と略記される．また束縛範囲の自明な括弧対は省略されることが多い．

3.2. 従属関係の定義

属性集合 $\mathcal{X} = \{X_1, \cdots, X_n\}$ 上の n 項関係 $R[\mathcal{X}]$ を考える．互いに素な属性集合 X, Y $(\subset \mathcal{X},\ X \cap Y = \phi)$ に対して，次のような属性集合 Z を与える：

$X = \{X_{i_1}, \cdots, X_{i_p}\}$，$Y = \{X_{j_1}, \cdots, X_{j_q}\}$，$Z = \mathcal{X} - (X \cup Y) = \{X_{k_1}, \cdots, X_{k_r}\}$.
ただし，$p + q + r = n$．

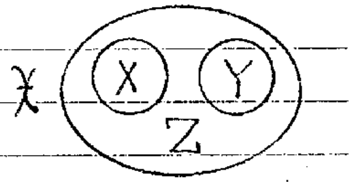


図3.1　互いに素な属性集合

また各属性 $X_i \in \mathcal{X}$ に対して，添字 i で属性の識別を行い，定義域 $\mathcal{D}_i$ を走る変数 x_i $(\in \mathcal{D}_i)$ を対応させる．

この時，タップル変数 $\vec{x} \triangleq (x_1, \cdots, x_n)$ に対して，量限定 $(\forall x_1) \cdots (\forall x_n)$ を $(\forall \vec{x})$，量限定 $(\exists x_1) \cdots (\exists x_n)$ を $(\exists \vec{x})$ で略記することにする

この略記法と前節の構文法を用い，関係 $R[\mathcal{X}]$ に対応する n 項関係 $R(\vec{x})$ において成立する属性 X から属性 Y への従属関係の諸定義を，以下に掲げる．ただし知識フィルタの性質を調べるのに有用な三種の従属関係の定義にとどめる．

それらはR〔$\mathfrak{X}$〕上におけるXからYへの関数従属（FD：Functional Dependency），多値従属（MVD：Multi-Valued Dependency），および相互従属（MD：Mutual Dependency）であり，それぞれ$X \rightarrow Y$，$X \twoheadrightarrow Y$，および$X \leftrightarrow Y$と表記される．下記の諸定義に現われる“．”，“＝”は，それぞれ射影および等号であるが，これらは周知の素命題への分解公理を用い，全て句あるいは定義域句に分解されるものとする．

〔定義6〕　関数従属（FD）

$$X \rightarrow Y \mathrel{\hat{=}} (\vec{t})(\vec{v})\Big[\big\{R(\vec{t}) \wedge R(\vec{v}) \wedge (\vec{t}.X = \vec{v}.X)\big\} \supset (\vec{t}.Y = \vec{v}.Y)\Big] \qquad (1)$$

〔定義7〕　多値従属（MVD）

$$X \twoheadrightarrow Y \mathrel{\hat{=}} (\vec{t})(\vec{v})(\vec{w})\Big[\big\{R(\vec{t}) \wedge R(\vec{v}) \wedge (\vec{t}.X \cup Z = \vec{w}.X \cup Z) \wedge (\vec{v}.X \cup Y = \vec{w}.X \cup Y)\big\} \supset R(\vec{w})\Big] \qquad (2)$$

〔定義8〕　相互従属（MD）

$$X \leftrightarrow Y \mathrel{\hat{=}} (\vec{t})(\vec{u})(\vec{v})(\vec{w})\Big[\big\{R(\vec{t}) \wedge R(\vec{u}) \wedge R(\vec{v}) \wedge (\vec{t}.X \cup Z = \vec{w}.X \cup Z)$$

$$\wedge (\vec{u}.Y \cup Z = \vec{w}.Y \cup Z) \wedge (\vec{v}.X \cup Y = \vec{w}.X \cup Y)\big\} \supset R(\vec{w})\Big] \qquad (3)$$

〔定義6〕～〔定義8〕は，Nicolas〔13〕のような“ただし書き付き代入”を用いておらず，LMSLのwffの正記法となっている．FD，MVD，およびMDの最も簡単な例を，図3.2に示す．

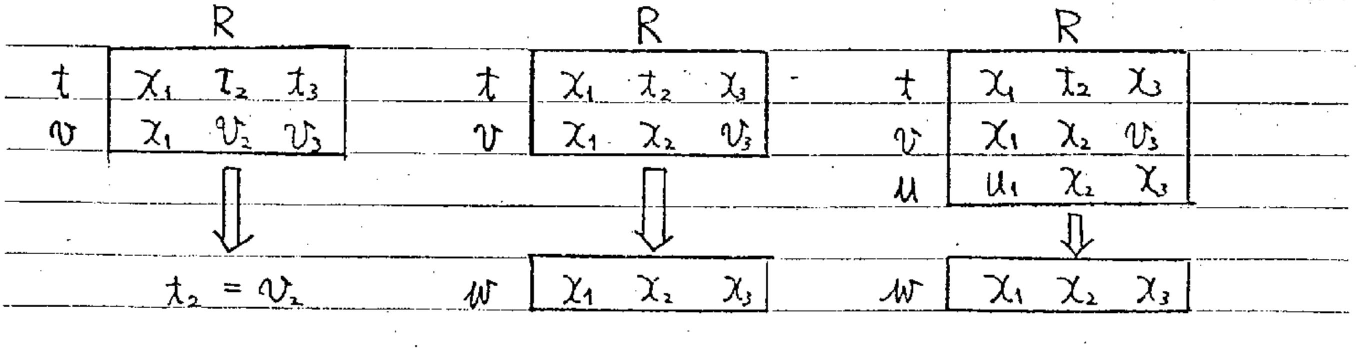
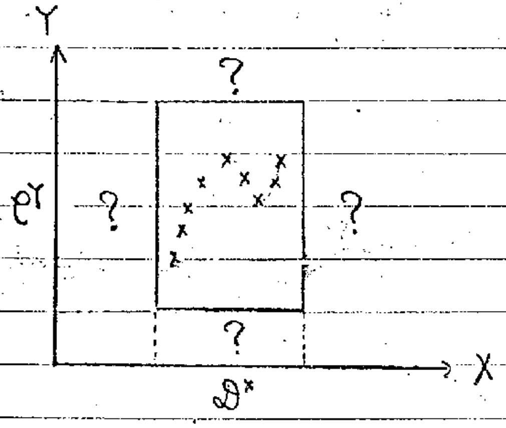


図3.2　FD.，MVD，およびMDの典型的な例

3.3. 条件付従属関係

関係データベース・モデルの従属関係理論研究の過程において，著者の一人[22]は，上林[24]の問題提起した"context"の問題を解決する手段として，あるいは複数の関係データベースを合成する必要性から，従属関係が成立しうる"場(field)"を陽的表示しうる条件付従属関係（CD: Conditional Dependency）という概念を導入した。

前節の如くとられた $X, Y\ (\subseteq \mathcal{X})$ に対して，直積 $\mathcal{D}^X \doteq \mathcal{D}_{X_{i_1}} * \mathcal{D}_{X_{i_2}} * \cdots * \mathcal{D}_{X_{i_p}}$ および $\mathcal{D}^Y \doteq \mathcal{D}_{X_{j_1}} * \mathcal{D}_{X_{j_2}} * \cdots * \mathcal{D}_{X_{j_r}}$ を考える。ここに直積 $\mathscr{D}^X * \mathscr{C}^Y$（ただし $\mathscr{D}^X \subseteq \mathcal{D}^X$, $\mathscr{C}^Y \subseteq \mathcal{D}^Y$）内において，XからYへの従属関係が存在する時，$\mathscr{D}^X$ と $\mathscr{C}^Y$ をそれぞれ始域(domain field)と終域(codomain field)と呼ぶ。あるいは始域 $\mathscr{D}^X$ から終域 $\mathscr{C}^Y$ へのCDが存在するとも呼ぶ。



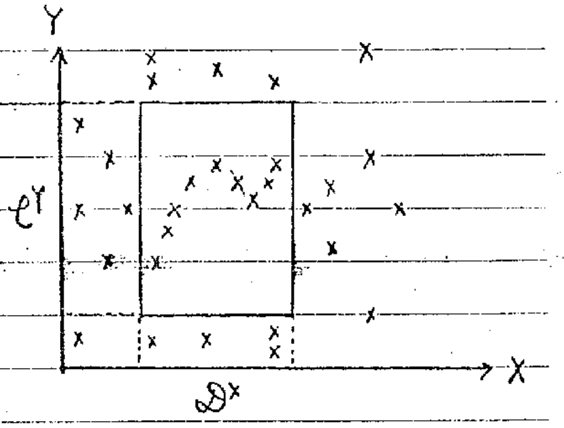


図3.3 条件付関数従属の例　　　図3.4 始終域切断による場の設定

さてCDには，場の無条件設定，始域切断による設定，終域切断による設定，始終域切断による設定という4種類のタイプ[23]が成立した。このうち始終域切断による場の設定というタイプは全て1階多類論理のwffで表記できる。しかしながら他の3タイプは，前々節で与えたwffでは表記できない。その原因は，τ-wffとwffを完全に分離して取扱ったからである

〔例1〕 条件付関数従属（CFD: Conditional Functional Dependency）

・始終域切断により場を設定したCFDの場合，

$$\mathscr{D}^X \to \mathscr{C}^Y \triangleq \forall \vec{x}\, \forall \vec{v} \left[\{ R(\vec{x}) \wedge R(\vec{v}) \wedge \mathscr{D}^X(\vec{x}.X) \wedge \mathscr{D}^X(\vec{v}.X) \wedge \mathscr{C}^Y(\vec{x}.Y) \wedge \mathscr{C}^Y(\vec{v}.Y) \wedge (\vec{x}.X = \vec{v}.X) \} \right.$$
$$\left. \supset (\vec{x}.Y = \vec{v}.Y) \right]$$

$$= (\vec{x}.X / \mathscr{D}^X)(\vec{x}.Y / \mathscr{C}^Y)(\vec{x}.Z)(\vec{v}.X / \mathscr{D}^X)(\vec{v}.Y / \mathscr{C}^Y)(\vec{v}.Z) \left[\{ R(\vec{x}) \wedge R(\vec{v}) \wedge (\vec{x}.X = \vec{v}.X) \} \right.$$
$$\left. \supset (\vec{x}.Y = \vec{v}.Y) \right] . \tag{4}$$

・場を無条件設定したCFDの場合，

$$\mathscr{D}^X \to \mathscr{C}^Y \triangleq \forall \vec{x}\, \forall \vec{v} \left[\{ R(\vec{x}) \wedge R(\vec{v}) \} \supset \{ \mathscr{D}^X(\vec{x}.X) \wedge \mathscr{D}^X(\vec{v}.X) \wedge \mathscr{C}^Y(\vec{x}.Y) \wedge \mathscr{C}^Y(\vec{v}.Y) \right.$$
$$\left. \wedge ((\vec{x}.X = \vec{v}.X) \supset (\vec{x}.Y = \vec{v}.Y)) \} \right] . \tag{5}$$

□

従って始終域切断により場を設定したFD・MVD・MDをそれぞれCFD・CMVD・CMDとすれば，それらは前々節で導入したwffである．ここに式(1)と式(4)の[ ]内が同一形式をとっていることに注意されたい．すなわち1階多類論理のwff表記法は，"ある知識が成立している場"と"知識の内容"とを形式的に分離して取扱うことを可能とするのである．

本節および前節で導入した従属関係の間には，次の推論図式が成立する．

$$\begin{array}{ccccc} \mathrm{FD} & \supset & \mathrm{MVD} & \supset & \mathrm{MD} \\ \cup & & \cup & & \cup \\ \mathrm{CFD} & \supset & \mathrm{CMVD} & \supset & \mathrm{CMD} \end{array}$$

図3.5 従属関係関連図

4. 拡張仮想関係

4.1 仮想関係の解釈

Chang[10]の仮想関係は，Horn節型公理の一個の正の単位句部分として定義されている．述語論理では，そのような公理が真となるあらゆるモデルの下で真となる論理的帰結として，仮想関係を解釈しうる．

［例2］　仮想関係

単純属性 X_1, X_2 に対応する定義域 $\tau_1=\{a,b,c\}$, $\tau_2=\{a\}$ が与えられているとする．$\{X_1, X_2\}$ 上の基底関係 $R[X_1, X_2]$ の外延を $R=\{(a,a),(b,a)\}$ とする．また仮想関係 $P_v[X_1, X_2]$ が，次式(6)のような内包としての論理表現で定義されているとする．

(外延)

τ_1

a
b
c

τ_2

a

R

X_1	X_2
a	a
b	a

(内包)　$(x_1/\tau_1)(x_2/\tau_2)\{R(x_1,x_2)\supset P_v(x_1,x_2)\}$　(6)

すると2.1節の関係データベースの古典論理での解釈法により，$R(x_1,x_2)$ は "$R(a,a)=T$, $R(b,a)=T$, かつ $R(c,a)=F$" と解釈される．そこで式(6)を真とする論理的帰結を求めると，次のようになる．

$\{\sim R(a,a)\vee P_v(a,a)\}\wedge\{\sim R(b;a)\vee P_v(b,a)\}\wedge\{\sim R(c,a)\vee P_a(c,a)\}=T$

$P_v(a,a)\wedge P_v(b,a)\wedge\{T\vee P_v(c,a)\}=T$

$\therefore\ P_v(a,a)=T$ かつ $P_v(b,a)=T$

P_v

X_1	X_2
a	a
b	a

$P_v(c,a)$ については，TともFとも言えない．

□

〔例2〕のごとく解釈すると，証明できないものは仮想関係に含まれず，その値の真偽が分らないはずである．故に$Pv(c,a)$の真偽は分らないはずだが，Changの仮想関係モデルでは，$Pv(c,a) = F$を割当てている．

実はこのことは，Reiterの用いた「真であると証明できないものは偽とみなす」閉世界仮説（CWA: Closed World Assumption）を，明示せずに導入していることに他ならない．仮想関係にCWAを組込めば，生成される仮想関係の直前に現われる"⊃"は，実は"≡"の意味と同値となる．Changは，書換え規則(rewriting rule)の適用による仮想関係消去アルゴリズムにおいて，"⊂"の部分を用いなかったので，明示しなかったと考えられる．

Changの立場を離れて〔例2〕を解釈すると，式(6)の論理的帰結として「真であると証明されるもののみ真である」とみなすこともできる．この立場によれば CWAを採用する必然性はなく，むしろ帰納的定義における最小性の仮定を論理に導入すればよいことになる．

4.2 仮想関係のHorn節による定義と有向連結グラフ

一般にevaluational approachでは仮想関係の外延を，〔例2〕のように論理的帰結として計算しない．2.2節で述べたように IDBの公理は，仮想関係表現での問合せを，基底関係のみ含む問合せに変形する際にのみ用いられ，仮想関係の評価には使用されない．この問合せ変形法として，書換え規則法〔10〕が有向連結グラフを用いて考察されている．

有向連結グラフは，Horn節で書かれた公理と問合せの否定形のそれぞれに節点を対応づけ，節点内外の仮想関係の統一可能な句同士を枝で連結したものである．各枝はラベル付けされており，このラベル情報により，どの句同士が統一置換されるべきかが具体的に分る．枝の向きは，正の句から出て負の句に入ることにより定義される．従って L_i $(i=1,\cdots,l)$ と V を仮想関係の句，B_j $(j=1..m)$ を(等号を含む)基底関係

の句とする時，一般に各節点は，次のように図示される：

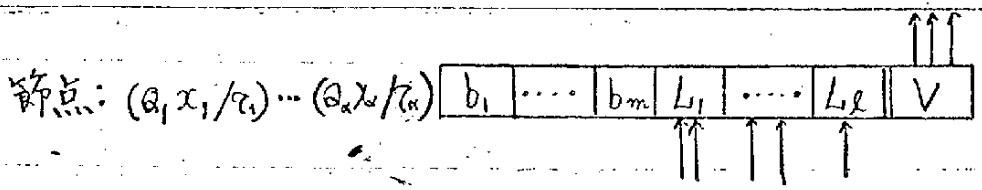


図4.1

公理：$(Q_1x_1/\tau_1)\cdots\cdots(Q_\alpha x_\alpha/\tau_\alpha)\{(b_1\wedge\cdots b_m\wedge L_1\wedge\cdots\wedge L_\ell\supset V\}$ (7)

但し $x_1,\cdots,x_\alpha$ は上記の節に現われる個体変数の全てであり，$Q_1,\cdots,Q_\alpha$ は量記号である∀あるいは∃であり，$\tau_1,\cdots,\tau_\alpha$ は定義域である．特に V に出現する全ての変数 $x_1,\cdots,x_\beta$ $(1\leq\beta\leq\alpha)$ に対する量記号 $Q_1,\cdots,Q_\beta$ は全称記号∀である

図4.1において節点の区切り"|"は選言を，区切り"‖"の左側にはHorn型公理の前提側の句，右側には帰結側の句が入る

〔例3〕 先祖関係

"xの父はyである"という関係 father(x,y) という表が与えられているとする．また "xの先祖がyである"という関係を Ancestor(x,y) とする．この時，"yがxの先祖である"という概念は，"(1) xの父がyならば，yはxの先祖である；(2) xの父がzであり，かつzの先祖がyとなるようなzが存在すれば，yはxの先祖である"という帰納的定義によって規定される．この事を1階多類論理の公式，および有向連結グラフ表現すると，下記のようになる．

$$(x)(y)\,[\,\text{father}(x,y)\supset\text{Ancestor}(x,y)\,],$$
$$(x)(y)(z)\,[\{\text{father}(x,z)\wedge\text{Ancestor}(z,y)\}\supset\text{Ancestor}(x,y)]. \quad (8)$$

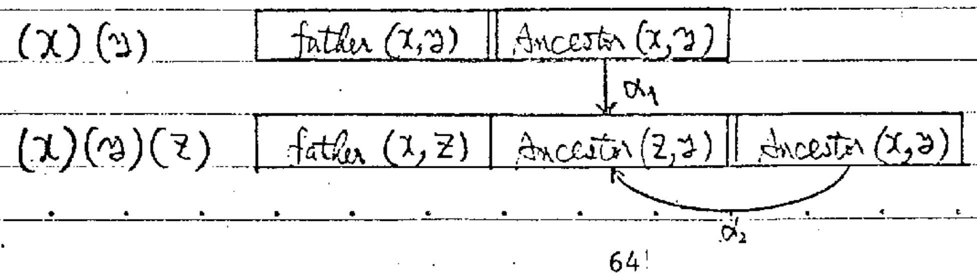


図4.2

□

4.3. 拡張仮想関係の形式

有向連結グラフに次のような節点を許容して，仮想関係を拡張する．

有限個の基底関係 $b_1, \cdots, b_n$ のみからなる wff の冠頭標準形を考える．すなわち量限定 $(Q_1 y_1/\tau_1)\cdots(Q_\beta y_\beta/\tau_\beta)$ を $(Q_y \vec{y}/\tau_y)$（但し $\vec{y} \mathrel{\hat{=}} (y_1, \cdots, y_\beta)$），　，量限定 $(Q'_1 z_1/\tau'_1)\cdots(Q'_r z_r/\tau'_r)$ を $(Q_z \vec{z}/\tau_z)$（但し $\vec{z} \mathrel{\hat{=}} (z_1, \cdots, z_r)$）と略記すると，それは次のように表現できる．

$$(Q_y \vec{y}/\tau_y)(Q_z \vec{z}/\tau_z)\, B(\vec{y}, \vec{z}) \qquad (9)$$

ここに $(Q_y \vec{y}/\tau_y)(Q_z \vec{z}/\tau_z)$ は前置記号，$B(\vec{y}, \vec{z})$ は母式と呼ばれる．式 (9) において量限定 $(Q_y \vec{y}/\tau_y)$ を除去した式を $b(\vec{y})$ とする．

$$b(\vec{y}) \mathrel{\hat{=}} (Q_z \vec{z}/\tau_z)\, B(\vec{y}, \vec{z}) \qquad (10)$$

式 (10) の $\vec{y}$ は自由変数であり，式 (9) から式 (10) を作り出すことを束縛変数の自由化と呼ぶ．

この時，拡張仮想関係 $V(\vec{y})$ は，次式で定義される．

公理：　$(Q_y \vec{y}/\tau_y)\,\{b(\vec{y}) \supset V(\vec{y})\}$，すなわち
$$(Q_y \vec{y}/\tau_y) \sim (Q_z \vec{z}/\tau_z)\,\{B(\vec{y}, \vec{z}) \supset V(\vec{y})\} \qquad (11)$$

節点：　$(Q_y \vec{y}/\tau_y) \sim (Q_z \vec{z}/\tau_z)$

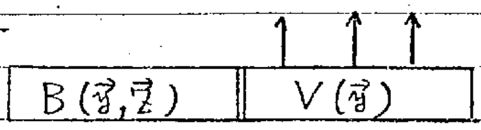


図4.3

仮想関係の場合と同様に，V に出現する全ての変数 $y_1, \cdots, y_\beta$ に対する量記号 $Q_1, \cdots, Q_\beta$ は全称記号 $\forall$ である．

式 (11) を，いわゆる節形式の wff に変換すれば，必ずしも Horn 節にならない節が派生することに注意されたい．

4.4. 拡張仮想関係と書換え規則

次の書換え規則 (a) (b) (c) は, Chang [10] により与えられた.

(a) 全ての左側の句 n について, 右側の句 $m_1,\cdots,m_r$ から直接 n を指す枝をもつならば;

$$W(n) = \alpha_1 W(m_1) \cup \cdots \cup \alpha_r W(m_r) \qquad (12)$$

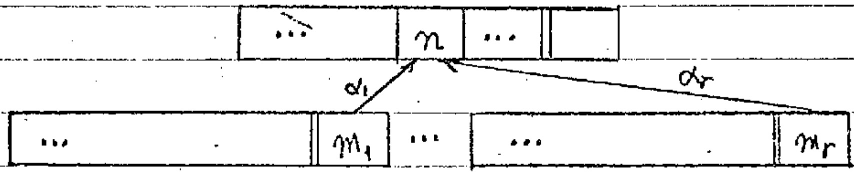


図4.4

(b) 全ての右側の句 n から出る枝がある時;

$$W(n) = P_1 \cdots P_r \qquad (13)$$

$$,\ ここに\quad P_i = \begin{cases} W(m_i) & : m_i が仮想関係を含む句 \\ m_i & : それ以外 \end{cases}$$

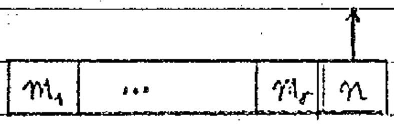


図4.5

(c) 問合せについては その否定形について;

$$T = Q_1 \cdots Q_m \qquad (14)$$

$$,\ ここに\quad Q_i = \begin{cases} W(c_i) & : c_i が仮想関係を含む句 \\ c_i & : それ以外 \end{cases}$$

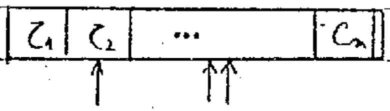


図4.6

(a)(b)(c) の規則は, T および $W(L_i)$ (L_i: 仮想関係の句) を, 形式言語理論における開始(非終端)記号と非終端記号, また基底関係の句 b_i やラベル記号 α_i を終端記号とみなせば, CFG 生成規則と同一の形式をもつ. そこで CFG における文生成と

同様にしてプランが生成され，このプランに統一置換の操作を行って変形問合せが得られる．

さて拡張仮想関係は，前節で与えられた節点も有向連結グラフに許容したものであり，その書換え規則は，(a)(b)(c)に，更に次の(d)を追加したものである．

(d) 仮想関係の句をVとした時；

$$W(V)=B \qquad (15)$$

プラン生成の際には，W(V)は1個の非終端記号，Bは1個の終端記号とみなされる．母式$B(\bar{x},\bar{z})$は，(等号を含む)基底原子句t_j ($j=1..m$)と論理結合記号$\sim, \wedge, \vee, \supset, \equiv$からなるが，母式に対応する終端記号Bの処理は，Extensional Evaluatorレベルで実行される

4.5. 長所と応用

Changは仮想関係をHorn節の公理で定義し，また問合せをReitaのいう肯定的問合せ(positive query)を仮定している．従って彼のアプローチは，彼自身は明記していないが，確定的な(definite)解を与えるという長所がある．

一方拡張仮想関係は，必ずしもHorn節でない公理も含んで定義されるが，問合せを肯定的なものに限定すれば，同様に確定性を保存する．そして従属関係等を含む一般的な仮説に基づく知識フィルタを定義可能にする．それ故，拡張仮想関係の応用として，演繹的QAシステムでの知識フィルタを設定するのに有効である．具体的な利用法は，6章の例題で説明する．

5. 知識フィルタ切断法

5.1 全称束縛の自由化による知識フィルタ切断法

具体的な知識フィルタの定義法を述べる．Reiter[6]らの閉世界仮説とは独立に概念規定しうる所に，その特徴がある．なお本節で得られる結果は，Kowalskiの論文[27]と比較されたい．

直積$\tau_1,\cdots,\tau_m$上のm個の基底関係$b_1(\vec{x}_1),\cdots,b_m(\vec{x}_m)$（$\vec{x}_1,\cdots,\vec{x}_m$：タップル変数）がEDBとして存在したとする．このm個の基底関係の自然結合が満たすべき仮説$H(\vec{x}_1,\cdots,\vec{x}_m)$（基底関係のみからなる任意のwff）と，自然結合される変数同士を等号に関する連言で束縛(binding)するwff $J(\vec{x}_1,\cdots,\vec{x}_m)$とが与えられているとする．この時，次の公理について考察してみる．

$$(\vec{x}_1/\tau_1)\cdots(\vec{x}_m/\tau_m)\{b_1(\vec{x}_1)\wedge\cdots\wedge b_m(\vec{x}_m)\wedge J(\vec{x}_1,\cdots,\vec{x}_m)\supset H(\vec{x}_1,\cdots,\vec{x}_m)\} \qquad (16)$$

式(16)において束縛変数$\vec{x}_1,\cdots,\vec{x}_m$の自由化を行った後，量限定部を除去した式で前提部での連言をとった式を，仮説$H(\vec{x}_1,\cdots,\vec{x}_m)$に基づく知識フィルタと呼び，$V(\vec{x}_1,\cdots,\vec{x}_m)$と表記する．

$$V(\vec{x}_1,\cdots,\vec{x}_m)\triangleq \underbrace{b_1(\vec{x}_1)\wedge\cdots\wedge b_m(\vec{x}_m)\wedge J(\vec{x}_1,\cdots,\vec{x}_m)}_{\text{切断対象wff}}\wedge\underbrace{H(\vec{x}_1,\cdots,\vec{x}_m)}_{\text{切断方法wff}} \qquad (17)$$

全く同様にして，仮説$\sim H(\vec{x}_1,\cdots,\vec{x}_m)$に基づく知識フィルタ$\overline{V}(\vec{x}_1,\cdots,\vec{x}_m)$は，次式で与えられる．

$$\overline{V}(\vec{x}_1,\cdots,\vec{x}_m)\triangleq b_1(\vec{x}_1)\wedge\cdots\wedge b_m(\vec{x}_m)\wedge J(\vec{x}_1,\cdots,\vec{x}_m)\wedge\sim H(\vec{x}_1,\cdots,\vec{x}_m) \qquad (18)$$

式(17)と(18)の"≙"は，論理的には"≡"の意味であるが，その片側の"⊂"のみ使用し，知識フィルタ$V(\vec{x}_1,\cdots,\vec{x}_m)$と$\overline{V}(\vec{x}_1,\cdots,\vec{x}_m)$に基づく仮説的公理を構成すると，次のようになる．

$$(\vec{x}_1/\tau_1)\cdots(\vec{x}_m/\tau_m)\{b_1(\vec{x}_1)\wedge\cdots\wedge b_m(\vec{x}_m)\wedge J(\vec{x}_1,\cdots,\vec{x}_m)\wedge H(\vec{x}_1,\cdots,\vec{x}_m)\supset V(\vec{x}_1,\cdots,\vec{x}_m)\} \qquad (19)$$

$$(\vec{r}_1/R_1)\cdots(\vec{r}_m/R_m)\{b_1(\vec{r}_1)\wedge\cdots\wedge b_m(\vec{r}_m)\wedge J(\vec{r}_1,\cdots,\vec{r}_m)\wedge\sim H(\vec{r}_1,\cdots,\vec{r}_m)\supset\overline{V}(\vec{r}_1,\cdots,\vec{r}_m)\} \quad (20)$$

式(17)と(18)より，知識フィルタの直和性を導くのは易しい．ここに，⊕は排他的論理和である．

$$V(\vec{r}_1,\cdots,\vec{r}_m)\oplus\overline{V}(\vec{r}_1,\cdots,\vec{r}_m)=b_1(\vec{r}_1)\wedge\cdots\wedge b(\vec{r}_m)\wedge J(\vec{r}_1,\cdots,\vec{r}_m) \quad (21)$$

式(21)より $V(\vec{r}_1,\cdots,\vec{r}_m)$ と $\overline{V}(\vec{r}_1,\cdots,\vec{r}_m)$ との直和集合とは，m個の基底関係達を自然結合して作った関係集合に等しい．従って $V(\vec{r}_1,\cdots,\vec{r}_m)$ ($\overline{V}(\vec{r}_1,\cdots,\vec{r}_m)$) は，それらの自然結合された集合に関して閉じており，$H(\vec{r}_1,\cdots,\vec{r}_m)$ ($\sim H(\vec{r}_1,\cdots,\vec{r}_m)$) に出現する量記号の解釈が，ユニークに定まる．

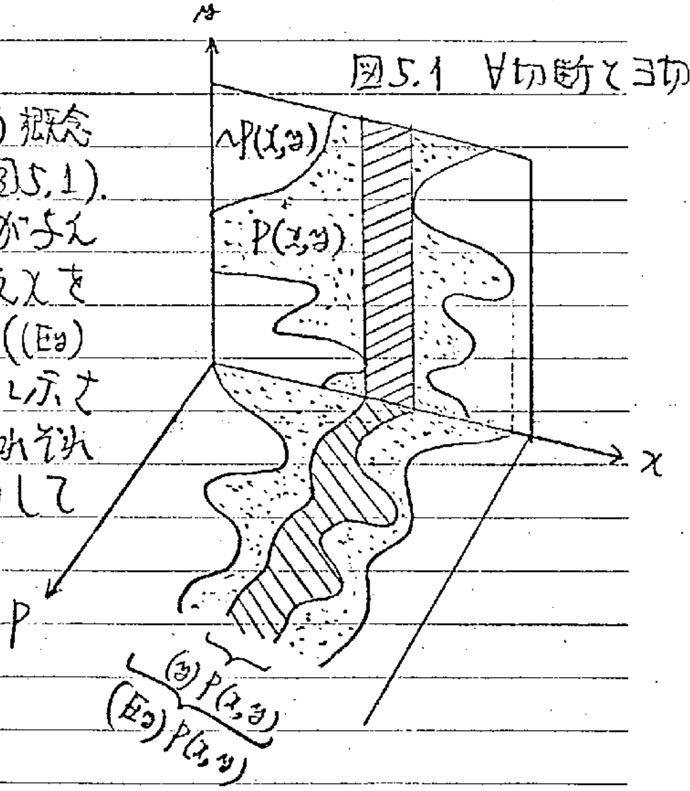


図5.1 ∀切断と∃切断

更に式(17),(18)で"切断"という概念を用いた理由を例示する(参照，図5.1)．wff $(x)(y)P(x,y)$ ($(x)(Ey)P(x,y)$) が与えられていたとする．この時，束縛変数xを自由化した式を $V(x)\triangleq(y)P(x,y)$ ($(Ey)P(x,y)$) と置く．$V(x)$ は，図5.1に示されるように，Pが変更する際，それぞれ斜線部(点線部)のごとく変動していく．従って，それらを∀(∃)切断と呼ぶことにする．

5.2. 従属関係型知識フィルタ

従属関係型の知識フィルタは，前節の結果に $m=1, J(\vec{r}_1)=T$ と置いただけで十分である．

$V_{X\to Y}(\vec{r})$ FDフィルタ

$$\triangleq R(\vec{r})\wedge(\vec{v})[\{R(\vec{v})\wedge(\vec{r}.X=\vec{v}.X)\}\supset(\vec{r}.Y=\vec{v}.Y)] \quad (22)$$

$V_{X\to Y}(\vec{r})$ NFDフィルタ

$$\triangleq R(\vec{r})\wedge(E\vec{v})[R(\vec{v})\wedge(\vec{r}.X=\vec{v}.X)\wedge\sim(\vec{r}.Y=\vec{v}.Y)] \quad (23)$$

$V_{X \twoheadrightarrow Y}(\vec{x})$　　MVDフィルタ

$$\triangleq R(\vec{x}) \wedge (\vec{v})(\vec{w})[\{R(\vec{v}) \wedge (\vec{x}.X \cup Z = \vec{w}.X \cup Z) \wedge (\vec{v}.X \cup Y = \vec{w}.X \cup Y)\} \supset R(\vec{w})] \quad (24)$$

$\overline{V}_{X \twoheadrightarrow Y}(\vec{x})$　　NMVDフィルタ

$$\triangleq R(\vec{x}) \wedge (\exists \vec{v})(\exists \vec{w})[R(\vec{v}) \wedge (\vec{x}.X \cup Z = \vec{w}.X \cup Z) \wedge (\vec{v}.X \cup Y = \vec{w}.X \cup Y) \wedge \sim R(\vec{w})] \quad (25)$$

$V_{X \Leftrightarrow Y}(\vec{x})$　　MDフィルタ

$$\triangleq R(\vec{x}) \wedge (\vec{u})(\vec{v})(\vec{w})[\{R(\vec{v}) \wedge R(\vec{u}) \wedge (\vec{x}.X \cup Z = \vec{w}.X \cup Z) \wedge (\vec{v}.X \cup Y = \vec{w}.X \cup Y) \wedge (\vec{u}.Y \cup Z = \vec{w}.Y \cup Z)\} \supset R(\vec{w})] \quad (26)$$

$\overline{V}_{X \Leftrightarrow Y}(\vec{x})$　　NMDフィルタ

$$\triangleq R(\vec{x}) \wedge (\exists \vec{u})(\exists \vec{v})(\exists \vec{w})[R(\vec{v}) \wedge R(\vec{u}) \wedge (\vec{x}.X \cup Z = \vec{w}.X \cup Z) \wedge (\vec{v}.X \cup Y = \vec{w}.X \cup Y) \wedge (\vec{u}.Y \cup Z = \vec{w}.Y \cup Z) \wedge \sim R(\vec{w})] \quad (27)$$

式(22), (24), (26)に式(19)を，あるいは式(23), (25), (27)に式(20)を適用して得られる公理は，基底関係句に関しては必ずしもHorn節でない．しかしながら（拡張）仮想関係句に関しては，正句（positive literal）が高々ひとつというHorn節の制約を破っている訳ではない．

5.3 視座濾波法

既に筆者ら[11,17]は，関数従属と非関数従属（NFD：Non-Functional Dependency）のみを対象とする視座濾波法（View Filtering Method）の理論を，前処理手続きの実行という形式で提案している．すなわちいわゆる外部スキーマとして表明されたFD・NFDのみよりなる視座定義と，内部スキーマとして保有しているFD・NFDのみからなる統合性制約が与えられているとする．まず両スキーマそれぞれの整合性をはかり，ついで両スキーマ間の不整合性を併合しつつユーザの視座を実現するために，前処理として実行せねばならない必要最小限の濾波手続きは何かを与える理論を示した（参照，図5.2）

ところで本小論ではこの考え方を，(1) 他の従属関係をも扱えるように，(2) 前処理でなく実時間処理の場合に拡張している．その際採用されるシステム構成の概略は，図2.1左図に示してあるが，本節以降では，より詳細な基本シス

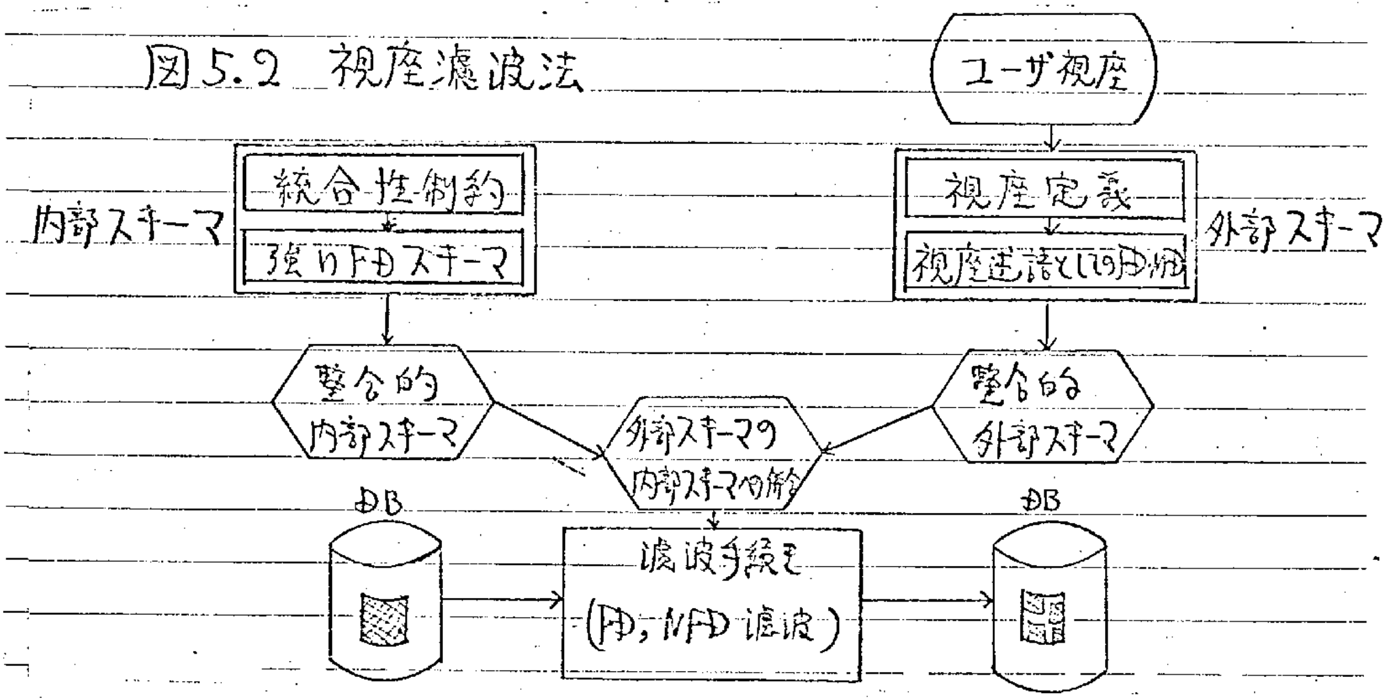


図5.2 視座濾波法

テム構成図を与える．その準備として，従属関係を反映する知識フィルタを構成する際に派生する基本的問題に対する考察を，次節にまとめておく．

5.4 従属関係の推論構造

一般に演繹的QAシステムにおいては，仮説的公理を用いて問合せ変形を行った後，一般前提的公理との整合性検査を行っている．ところがある種の問合せでは，一般前提的公理と仮説的公理の両者に関する公理を，最初から同時に用い解答抽出した方が，性能上有利なことがある．そのような場合の特殊例として，知識ベース上の一般前提的公理として，(C)FD・(C)MVD・(C)MDのみからなる従属関係の古典論理での定義wffsが記述されていたとする

〔観察1〕 整合性問題

FD, MVD, MD, CFD, CMVD, およびCMD達からなる公理は整合的(consistent)である．

∵)

従属関係のwffs表現が，全て唯一個の正の句をもつHorn節

だからである[13]. □

［観察2］　併合問題

上述の従属関係達からなる公理から ある従属関係の公理が推論されるかどうかを決定する妥当性問題は可解(decidable)である. ■

∵)

従属関係のwff(ω)表現が，定義域付の全称記号しか現われない冠頭連言標準形をしていることより，この妥当性問題は可解なクラスに陥る □

［観察1］より，強いFDスキーマを考えた場合[16,17]に必要であった，統合性制約 と 視座定義 それぞれの整合性検査が不要であることが分る また［観察2］より，知識フィルタしたいある従属関係の統合性制約からの妥当性問題が，通常の分解証明法あるいは theorem prover により確認することが，常に可能である. 要約すると，このような従属関係のみを含む一般前提的公理と仮説的公理の全体，それぞれの整合性問題は肯定的であり，また それとの間の妥当性問題は可解である 従って一般前提的公理の全体から，ある仮説的公理が推論されなかった場合のみ，その仮説的公理に対応する知識フィルタを設営する必要がある.

5.5 基本システム構想

前節の考察により 一般前提的公理と仮説的公理とが，帰納的定義を含まない従属関係のみの場合，次のような設計方針を得る.

［設計方針］　従属関係の併合問題

統合性制約をIC，視座定義したい任意の従属関係をĐとする.

(a) "wff(IC) ⊃ wff(Đ)" の場合:

(a.1) D型知識フィルタを仮想的公理として追加することは，基底関係の外延で満足されているので，不要である．

(a.2) not D型知識フィルタは，実行しても基底関係あるいは空関係しか導出されないので，無意味である．

(b) そうでない場合： 視座定義に対応するD型あるいはnot D型の知識フィルタを実行する． □

この設計方針に基づくシステム構成図が，図5.3である

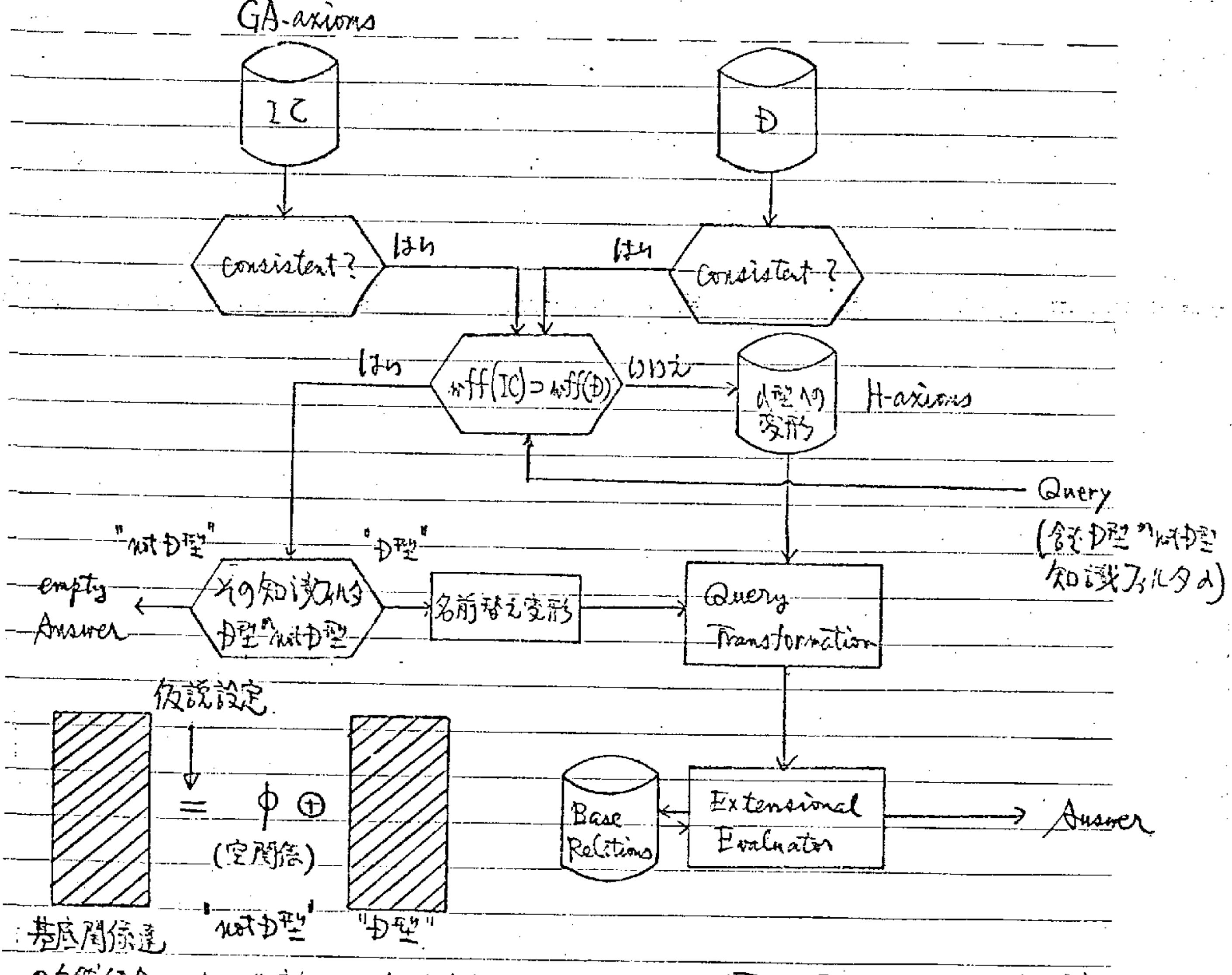


図5.3 システム構成図

6. 応用例

単一の基底関係に対する従属関係型知識フィルタの例を示す.

6.1 FD型知識フィルタ

"マネージャxがプロジェクトyをマネージしている" という関係M(x,y)の外延が, 右図の基底関係で与えられているとする. この時, "xがyの専任マネージャである" という関係 $M_{one}(x,y)$ と "xがyの兼任マネージャである" という関係 $M_{many}(x,y)$ は, それぞれ FD型知識フィルタと NFD型知識フィルタを用いた拡張仮想関係で定義される.

$$(x)(y)\left[M(x,y)\wedge(y')\{M(x,y')\supset(y=y')\}\supset M_{one}(x,y)\right] \quad (28)$$

$$(x)(y)\left[M(x,y)\wedge(Ey')\{M(x,y')\wedge\sim(y=y')\}\supset M_{many}(x,y)\right] \quad (29)$$

Manager	Project
a	A
a	B
b	B
c	A

M

=

Manager	Project
b	B
c	A

M_{one}

⊕

Manager	Project
a	A
a	B

M_{many}

この拡張仮想関係を用いれば, "Project Aの専任(兼任)マネージャは誰か?" という問合せに対し, M_{one} [Manager*, Project=A] (M_{many} [Manager*, Project=A]) というコマンドで, QAシステムへ照会可能である

本例は, DEDUCE 2で許容している numerical quantifier を用いれば, 次のような Horn節型の axiom で記述できる.

$$(x)(E=1,y)\left[M(x,y)\supset M_{one}(x,y)\right] \quad (30)$$

$$(x)(E\geq 2,y)\left[M(x,y)\supset M_{many}(x,y)\right] \quad (31)$$

一般に numerical quantifiers付きで記述された wff を, 通常の quantifiers のみ用いる wff に変換する方法が存在する. Chang[10]は, Horn節型の公理を考察した際, その前置記号の形式を numerical quantifiers に拡張しながら, 上記変換を実施した式が, 必ずしもHorn節でないことを提起しなかった. 本小論でのアプローチは, 確定的な解を得るための手段が numerical quantifiers にあるのでなく, 生成される知識フィルタのHorn

性にあることを明らかにした。従って基底関係のみからなる任意のwffから生成される知識フィルタは，numerical quantifiersでは表現できない公理も許容している。

6.2 MVD型知識フィルタ

Student(x,y)，Course(y,z)をそれぞれ"学生xの取得課目はyである"，"課目yは専攻zの必修課目である"という関係とする。「取得課目＝必修課目」とする両関係のnatural joinによって生成された関係S(x,y,z)は"学生xの取得課目yは，z専攻の必修課目である"ことを意味する。

この時，関係S(x,y,z)を関係Course(y,z)で割った商の関係をMajor(x,y,z)，余りの関係をNMajor(x,y,z)とする。またMajor(x,y,z)の{学生，専攻}方向の射影をとった関係をMajor1(x,z)とする。

Student

学生	取得
A	力
A	熱
A	解
B	力
B	代
C	解
C	代

*

Course

必修	専攻
力	物
熱	物
解	数
代	数

=

S

学生	課目	専攻
A	力	物
A	熱	物
A	解	数
B	力	物
B	代	数
C	解	数
C	代	数

≡

Major

学生	専攻
A	物
C	数

Major 1

(商)

*

専攻	課目
物	力
物	熱
数	解
数	代

Course

⊕

NMajor

学生	課目	専攻
A	解	数
B	力	物
B	代	数

(余り)

するとStudent，Course，S，Major，NMajor，およびMajor1の間には，上図のような関連がある。すなわちもっとも特徴的なのは，

$$\text{Major}[\text{学生，課目，専攻}] = \text{Major1}[\text{学生，専攻}] * \text{Course}[\text{専攻，課目}] \quad (32)$$

であり，関係Majorにおいては多値従属「専攻→→学生｜課目」が成立する。以上の考えは全く一般的であり，商と余りの関係

を求めることと, MVD型知識フィルタを拡張仮想関係で定義することとが対応する.

$$(x)(y)(z)[\xi(x,y,z) \wedge (x')(y')\{\xi(x',y',z) \supset \xi(x',y,z)\} \supset Major(x,y,z)] \quad (33)$$

$$(x)(y)(z)[\xi(x,y,z) \wedge (Ex')(Ey')\{\xi(x',y',z) \wedge \sim\xi(x',y,z)\} \supset NMajor(x,y,z)] \quad (34)$$

このようにすれば"専攻 物理の必須課目を取得した学生は誰か?"という問合せは, Major[学生*, 専攻=物理]で照会可能である. また吉川[25]は, 基底関係 ξを, ここでは仮想関係として定義したMajor1, Course, NMajorに外延的にPMVD分割することを提案している. このことは, MajorやNMajorのような内包的な知識を反映するデータは, 外延として生成せず, あたかも外延として存在するかの如く, ユーザに提供すればよいという著者らの見解と異なる.

6.3. 帰納的に定義された拡張仮想関係

Project

プロジェクト	チーム
I	a
I	b
II	c

Manage

マネージャ	チーム	プロジェクト		
A	a	I	○	X
A	b	I	○	X
A	c	II	○	
B	a	I	○	X
B	b	I	○	X
C	a	I	●	
D	c	II	○	
F	a	I	●	
F	c	II	○	

Sup-d

マネージャ	直属上司	
F	D	x
A	F	x
C	B	

○印: "プロジェクト→→チーム"という MVDフィルタリング

●印: "プロジェクト→→チーム"という NMVDフィルタリング

"チームyはプロジェクトxに属する"という関係 Project(x, y), "マネージャxは, プロジェクトzのチームyをマネージしている"という関係 Manage(x, y, z), "マネージャxの直属上司はyである"という関係 Sup-d(x, y)のEDBが, 上図の如く与えられているとする。

この時，関係 Manage を "プロジェクト$\twoheadrightarrow$チーム" という MVD フィルタリングすれば，"マネージャ x は，プロジェクト z の全てのチーム y をマネージしている" という関係が得られる．そこで "プロジェクト z の全てのチーム y をマネージしているマネージャあるいはその上司 x を表わす" 関係 Sup-all $(x;y,z)$ は，MVD フィルタリングされた拡張仮想関係として，次のように帰納的に定義される．

$$(x)(y)(z)\left[\mathrm{Manage}(x,y,z) \wedge (x')(y')\{\mathrm{Manage}(x',y',z) \supset \mathrm{Manage}(x',y,z)\} \supset \text{Sup-all}(x,y,z)\right] \tag{35}$$

$$(x)(y)(z)(u)\left[\text{Sup-d}(x,u) \wedge \text{Sup-all}(u,y,z) \supset \text{Sup-all}(x,y,z)\right] \tag{36}$$

(35)，(36) のように与えれば，"プロジェクト I の全チームをマネージしているマネージャあるいはその上司は誰か？" という問合せは，Sup-all［人*，プロジェクト＝I］と簡潔に表現できる．しからば，式 (35) と関係 Manage より，"プロジェクト I の全チームをマネージしているマネージャ" を表わす解 {A，B} が求まり，式 (36) と関係 Sup-d より "プロジェクト I の全チームをマネージしているマネージャの上司" を表わす解 {F，D} が，プランにより順次生成される．ここに解になるタップルには，× 印がつけられている．

7. あとがき

本小論で得られた結果を要約する。

第1章では演繹的QAシステムの設計論や関係データベース・モデルのスキーマ設計論の現状について調査した。その結果KBに格納すべき公理として、一般前提的公理と仮説的公理という2種類のものが存在し、かつ両者の相異を積極的に解答抽出に利用すべきであるという見解を述べた。

第2章では関係データベース・モデルの古典論理での解釈を、関係の内包と外延の両側面から述べ、ここで採用した問合せ評価法がevaluational approachであることを指摘した。ついで本小論でのapproachが、仮説的公理と一般前提的公理を順次に評価するのでなく、それらを同時に問合せ評価に利用するものであることを示した。

第3章では1階多類論理の構文法を与え、それを用いて、従来議論されている従属関係・条件付従属関係を定義するwffsを与えた。このwffsは、始終域切断による場の設定により派生する問題を分析するツールなりえることも示した。

第4章では拡張仮想関係の定義法、有向連結グラフ表現法、および書換え規則法によるプランの順次生成法を提案した。ここに拡張仮想関係とは、有限個の基底関係のみから構成される母式を一個の前提節点とみなす公理として定義され、しかもこの公理は必ずしもHorn節でない。ただし従来からあるHorn節で記述される仮想関係の定義法は、そのまま公理として残すものとする。

第5章では、筆者らが視座濾波法として前処理の場合に行った考え方を実時間処理の場合にも拡張し、一般前提的公理と仮説的公理の整合性問題・併合問題の解決法をシステム構成図として示した。特に全称束縛の自由化による知識フィルタ切断法の理論化を行う。この知識フィルタは、閉世界仮説とは独立に概念規定されたところに、その特徴がある。

第6章では拡張仮想関係および知識フィルタの応用として、仮説的公理の一種である従属関係を演繹的QAシステムの解答抽出に利用する方法を確立した。ここにおいては、帰納を含む仮説的公理が知識フィルタとしてQAに利用されている。

なお本報告においてCWAを採用すると，更に多くの問合せの型に対応する定理が成立するが，これに関しては別の機会に報告する．

今後の課題としては，(1) 一般前提的公理および仮説的公理のモデル理論の中での形式的取扱いの確立，(2) 1階多類論理の構文法・意味論とevaluational approachとの関連の理論化，(3) 全称束縛にとらわれない知識フィルタの一般論，(4) 閉世界仮説と開世界仮説の問合せ評価過程の比較研究 等の多くの部分課題が残っている．

〔謝辞〕 本研究の前報[26]に関し，熱心に討論していただき，かつ有益なコメントをいただいた北川敏男所長，加藤昭彦・竹島卓研究員（国際研），大須賀節雄先生（東大），野口正一・増永良文先生（東北大）に感謝します．特に北川所長・大須賀先生の「否定的情報（Negative Information）の取扱いを慎重に」というコメントが，閉世界仮説の再考を促してくれました．また北川所長からは全体構成および所々の概念設定に関し，増永先生からは evaluational approach 全体についての適切なアドバイスを受けました．最後に関連テーマについて討論していただいた古川康一，田中穂積（電総研），後藤滋樹（武蔵野通研）先生に感謝します．

〔参考文献〕

〔1〕 北川敏男：研究動向把握への情報学的接近，文部省特定研「情報システムの形成過程と学術情報の組織化」B-1班報告，1977，pp.1-31.

〔2〕 北川敏男：情報と認識，NHK大学講座，日本放送出版協会，1977.

〔3〕 Feigenbaum，E.A.： The art of artificial intelligence — Themes and case studies of knowledge engineering，NCC 1978，pp.227-240.

〔4〕 大須賀節雄：意味処理と知識利用のシステムについて，日本語情報処理シンポジウム報告集，情報処理学会プログラミング・シンポジウム委員会，1978.7，pp.234-261.

〔5〕 大須賀節雄：知識ベースとデータベース，電子通信学会，信学技報 Vol.79 No.200，AL79-71，pp.17-26.

〔6〕 Reiter，R.：An Approach to Deductive Question-Answering，BBN Rep. No. 3649，1977，pp.1-161.

〔7〕 Shortliffe，E.H.： Computer based medical consultation MYCIN，Elsevier，North-Holland，1976.

〔8〕 Weiss，S.M. and Kulikowski，C.A.：EXPERT - A system for developing consultation models，Proc. of 6th IJCAI，Tokyo，1979.

〔9〕 Kitagawa，T.：Structure formation of scientific brainware as a basis of data base management，Bull. Math. Statis，Vol.17，No.3-4，1977，pp.59-64.

〔10〕 Chang，C.L.：DEDUCE.2：Further Investigations of Deduction in Relational Data Bases，in Logic and Data Bases（Gallaire，H. and Minker，J. eds.），Plenum Press，1977.

〔11〕 Codd，E.F.：Further Normalization of the data base relational model，in Data Base Systems（Rustin，R. ed.），Prentice Hall，1971.

〔12〕 Fagin，R.：Multivalued Dependency and a New Normal Form for Relational Data Bases，ACM Trans. on Data Base Systems，1977.

〔13〕 Nicolas，J.M.：First Order Logic Formalization - functional，multi-valued，and natural dependencies，SIGMOD，1978，pp.40-46.

〔14〕 田中譲，津田孝夫：データベースにおけるアクセス制御，文部省特定研B-19班研究会資料，1978.

〔15〕 増永良文，野口正一：関係データベースにおける意味論に基づいたビューメカニズムの一構成法，電子通信学会 情報・システム部門全国大会，Oct 1979，p.451.

〔16〕 国藤進，若木利子，竹島卓：ユーザのセマンティック・ヴューを反映する関数従属の利用法について，情報処理学会全国大会，1978.

〔17〕 国藤進，若木利子，竹島卓：ユーザ・ヴューに基づくフィルタリング法

について, 電子通信学会, 信学技報 AL78-68, 1978.

[18] Gallaire, H and Minker, J. (eds): Logic and Data Bases, Plenum Press, 1977.

[19] Mendelson, E. : Introduction to Mathematical Logic, Second Edition, D. Van Nostrand Comp., 1979.

[20] Astrahan, M.M. etal. : SYSTEM R : A Relational Approach to Data Base Management, IBM Res. Lab, RJ 1738, Feb. 1976.

[21] 牧之内顕文他: リレーショナルデータベース管理システムRDB/V1のインプリメンテーション, 電子通信学会 情報・システム部門全国大会, Oct. 1979, p.454.

[22] 国藤進: 条件付従属を用いる関係データベース・スキーマに関する"場"の理論について, 電子通信学会, 信学技報 AL78-17, 1978.

[23] 国藤進: 関係データベース・スキーマにおける"場"の理論の役割について, 昭和53年度 情報処理学会 第19回全国大会, Aug. 1978.

[24] 上林弥彦: リレーショナル・データベースに対する論理関数の応用, 電子通信学会 信学技報 AL 77-35, 1977.

[25] Furukawa, K. : Relational Strategy for Processing Universally Quantified Queries to Large Data Bases, Proc. of 6th IJCAI, 1979.

[26] 国藤進, 若木利子: 知識ベースと統合性制約 ― その演繹的質問応答への利用法, 電子通信学会, 信学技報, AL79-72, 1979.

[27] R. Kowalski : Logic for Data Description, in Logic and Data Bases (Gallaire, H. and Minker, J. eds.), Plenum Press, 1977.

[28] 北川敏男: 実験計画法講義 I 基礎編 (1), 培風館, pp.129-130, 1955.

LUP: A Vehicle of Implementing a View Support Subsystem of a Relational DBMS

Yoshifumi Masunaga

Research Institute of Electrical Communication
Tohoku University
Katahira 2-1-1, Sendai, Miyagi 980
Japan

Abstract

In this paper, an architecture of implementing a view support subsystem of a relational DBMS is described by introducing the LUP(Local view Update Processor) concept with the view defining tree. That is, the LUP is a processor handling the translation of update, which initially stays at the root (i.e. the view) and then comes up step by step to a leaf (i.e. a base relation which is used to define the view) of the tree. If all LUP's in the tree succeed in reaching certain leaves, then the update execution will be initiated. The induced structural integrity constraint, the update modification rules and the augmentation rule are introduced as theoretical basis for describing the actions taken by a LUP. The actions of a LUP are described in detail, and the update execution control is also described. It is noted that the formal description of the meaning of a view is essential to define such LUP functions. Two examples are given to demonstlate this architecture. It should be stressed that the behavior of LUP's on the view defining tree just corresponds to a way of implementing the view update translation mechanism.

The Table of Contents

1. Introduction ... 4

2. Views ... 7

2.1 Definition ... 7

2.2 View Defining Tree ... 7

3. View Update Problems ... 9

4. View Support Subsystem ... 12

4.1 LUP ... 12

4.2 Theoretical Foundations ... 14

4.2.1 Induced Structural Integrity Constraint ... 14

4.2.2 Update Modification Rule ... 16

4.2.3 Augmentation Rule ... 17

4.3 LUP Functions ... 17

4.3.1 Preliminaries ... 17

4.3.2 Deletions ... 19

4.3.2.1 The Expanded Direct Product ... 19

4.3.2.2 Union ... 20

4.3.2.3 Difference ... 21

4.3.2.4 Projection ... 22

4.3.2.5 Restriction ... 22

4.3.3 Insertions ... 23

4.3.3.1 The Expanded Direct Product ... 23

4.3.3.2 Union ... 24

4.3.3.3 Difference ... 25

4.3.3.4 Projection ... 26

4.3.3.5 Restriction ... 27

4.4 Update Execution ... 27

4.4.1 Execution Control ... 27

4.4.2 Additive Side-effect Control ... 28

5. Examples ... 30

5.1 Example 1 -Deletion- ... 30

5.2 Example 2 -Insertion- ... 31

6. Concluding Remarks ... 33

1. Introduction

Modern DBMS architecture tends to take the three level construction to provide the physical and the logical data independence as proposed in the ANSI/X3/SPARC report [TSIC78].

In this framework, there are external, conceptual and internal schema which describes the objects in the realms of interest according to the three levels. It is understood that the physical data independence will be achieved by specifying a mapping between the conceptual schema and the internal schema, while the logical data independence will be achieved by specifying a mapping between the external schema and the conceptual schema. However it is said that most of the present commercial or institutional DBMS's hardly provide any external schema views and therefore they provide almost no logical data independence [TSIC77, KIM79].

Usually the external schema consists of the virtual data which are defined in terms of the data in the conceptual schema. It is understood that one of the most serious reasons why the present DBMS's hardly provide any external schema views is that there are many difficulties in defining a mapping between the external schema and the conceptual schema.

In the relational data model, the conceptual schema consists of a set of relations, called base relations, and the external schema consists of relations, called (user's) views, which are derived from the base relations or other views by applying a sequence of the relational algebra operations and the computing functions such as AVERAGE. Therefore the update to a view is only effected if the update to a view is translatable to the updates to the base relations which define the view and results the intended update result without causing any side-effect. There are at least two main issues in this problem. That is, the first one is to make it clear when and only when the update is translatable. The second one is to make it clear how the translation is done. Of course those two problems are closely connected.

In this paper, we try to make the second problem clear and give a solution to the first problem through the investigation to the second one. In the previous work [MASU79], we have investigated the first problem particuraly from the semantic point of view and showed that the meaning of a view should essentially be taken into account. Of course this paper stands on this point of view. However the main interest of this paper is to show how the second problem will be made clear from the algorithmic point of view. The new concept, LUP(Local view Update Processor) is introduced as a vehicle of implementing a view support subsystem, by which we can describe the mapping between the external schema and the conceptual schema. The LUP traverses the view defining tree from the lower level to the higher level and translates the update against the view step by step.

The following of this paper consists of as follows: In section 2, how views are defined is reviewed and the view defining tree is introduced. In section 3, the traditional view update problems are shortly re-examined and see the point why the meaning of a view should be taken into account to discuss those problems. In section 4, the LUP is introduced and an architecture of implementing the view support subsystem is described using the LUP concept with the view defining tree. A few theoretical foundations are given, LUP functions are described in detail, and the update execution under the LUP concept is described. In section 5, two examples of update translation is demonstlated. Section 6 concludes this paper.

2. Views

2.1 Definition

In the relational data model, relationships among attributes of an entity set and the associations between entity sets are represented as relations. As defined in [CODD70], a relation R(A1, A2, ..., An) on n attribute domains dom(A1), dom(A2), ..., dom(An) (where dom(Ai) represents the domain of the attribute Ai,) is a finite subset of the direct product dom(A1) x dom(A2) x ... x dom(An). (We call this derect product the domain of R and denote it by dom(R).)

A relation which is physically realized on a certain storage device such as a disk is called a base relation. A set of base relations with integrity constraints consists of the conceptual schema of a relational database system. A view is a relation derived from the base relations (or other views) by applying a sequence of relational algebra operations and computing functions such as AVARAGE. (However, in this paper, we exclude views derived by computing functions to make our discussion simpler.) A view is a virtual relation and a set of views consists of an external schema of a relational database system.

2.2 View Defining Tree

In order to introduce the view defining tree, let us here review more precisely how views are defined: Originally the following eight operations are introduced as elements of the relational algebra, i.e. four traditional set operations (the expanded direct product, union, intersection, difference) and other four less traditional operations on relations (projection, join, division, restriction) [CODD72]. However as it is known those eight operations are not mutually independent. Among them we deduce five operation, i.e. the expanded direct product($\otimes$), union($\cup$), difference($-$), projection($R[A]$) and restriction($R[A\,\theta\,B]$), as a minimal set of operations. The reason of this selection is that the expanded direct product operation is essential to expand a relation, the projection and the restriction operations are essential to restrict a relation in the virtical and the horizontal direction respectively. It is clear that in this framework the θ-join of relation R on domain A with relation S on domain B is defined by $R[A\,\theta\,B]S = (R \otimes S)[A\,\theta\,B]$ and the division of R on A by S on B is defined by $R[A \div B]S = R[\bar{A}]-((R[\bar{A}]\otimes S[B])-R)[\bar{A}]$ [CODD72].

Now, the view defining tree is defined according to the defining expression of the view. For example, let us suppose that there are two base relations ED(EMP, DEPT) and DM(DEPT, MGR). In our framework, the natural join of ED on DEPT with DM on DEPT is defined by $ED \bowtie DM = ((ED \otimes DM)[DEPT^1 = DEPT^2])[EMP^\frown DEPT^{2\frown} MGR]$, where the superscripted number 1 and 2 are used to distinguish that $DEPT^1$ belongs to ED and $DEPT^2$ belongs to DM.

The view defining tree of this view is shown in Fig.1. Notice that the notion of tree was found in [OSMA79].

3. View Update Problems

In this section we shortly review the traditional view update problems and see why the meaning of a view (or a relation) should be introduced to treate those problems. We do it by taking a simple but typical example:

Let the extensions of the base relation ED, DM and the natural join view EDM be as shown in Fig.2. (Those are denoted by ed, dm and edm respectively.)

(a) Suppose one wants to delete a tuple (e1, d1, m1) from the view edm. Then three alternatives are considerable for this update translation: (1) delete the pair (e1, d1) from ed, (2) delete the pair (d1, m1) from dm, (3) delete (e1, d1) and (d1, m1) from ed and dm respectively. But no alternative is adoptable without causing the side-effect.

(b) Suppose one wants to insert a tuple (e5, d3, m4) to edm. To effect this update, one might translate it into two insert statements, each of which inserts the pair (e5, d3) to ed and the pair (d3, m4) to dm respectively. But this translation causes the additive side-effect.

(c) Suppose one wants to delete a tuple (e3, d2, m3) from edm. In this case, any one of the following three alternatives is adoptable: (1) delete the pair (e3, d2) from ed, (2) delete the pair (d2, m3) from dm, (3) delete (e3, d2) and (d2, m3) from ed and dm respectively. But one can not decide uniquely which alternative shoulc be chosen. This is one of the uniqueness problems.

Many investigations have been done to those problems [DATE71, CODD74, CHAM75, STON75, FERN76, PAOL77, DAYA78, BANC79, OSMA79, MASU79]. Among them, a semantic aspect of those problems was investigated in [PAOL77 , BAMC79 and MASU79]. Particularly, in [MASU79] the "meaning" of a relation was introduced and it played an essential role to characterize the translatable updates. (For example, the meaning of the base relation ED is a semantic nature of it by which we can see that "the employee e1 works in the department d1" as long as the pair (e1, d1) belongs to ED. This is formally denoted by M_{ED}.) The main results there were shown taking EDM as an example:

(a) Let M_{ED} and M_{DM} be the formal descriptions of the meaning of the base relations ED and DM respectively. Then the meaning of the natural join EDM is defined by

(E1) $((\forall t \in \mathrm{dom}(R) \times \mathrm{dom}(S))(M_{EDM}(t) \equiv M_{ED}(t[EMP^\frown DEPT]) \wedge$

$M_{DM}(t[DEPT^\frown MGR]))$.

Now suppose one wants to delete a tuple (e1, d1, m1) from edm. Then the next statement holds:

(E2) "M_{EDM}(e1, d1, m1) is false if and only if either M_{ED}(e1, d1) is false or M_{DM}(d1, m1) is false or both."

If the intention of deleting the tuple (e1, d1, m1) from edm were comming from the fact that the pair (e1, d1) had lost the meaning (i.e. M_{ED}(e1, d1)) is false), then the correct delete statement to be issued against edm should be a delete statement, which is capable of deleting all tuples having e1 and d1 as the EMP value and the DEPT value respectively. The similar arguments hold for other two cases. Therefore the tuple delete requirement of deleting a tuple (e1, d1, m1) from edm was nonsense. As a result, this enables us to characterize the set of all translatable delete statements against EDM.

(b) So far as we are concern with the meaning of a view, the uniqueness problem stated previously does not arise.

(c) However, there exists yet another aspect of the view update problem. (This means that the previous additive side-effect comes from the nature of the natural join operation.) We note here that the meaning of a relation approach is still essential in the following investigations.

4. View Support Subsystem

4.1 LUP

LUP(Local view Update Processor) is a vehicle of implementing a view support subsystem of a relational DBMS. Generally, as observed in the previous section, the view update problems are complicated and therefore it seems very difficult to present a simple view update translation mechanism. However, it seems that one can possibly implement a view support subsystem if we first observe what happens when an update against a view is translated, and then identify some principles which rule the translation.

In our approach, a view is defined as a relation derived from the base relations by applying a sequence of five relational algebra operations and the derivation is shown as a tree which root represents the view, which leaves represent the base relations and which intermediate nodes represent certain intermediate relations. (We call the root is in the lowest level and therefore others are in higher level.) Therefore we can follow faithfully how the update against the view will be translatable to the updates to leaves by identifying when the translation is possible and how the translation is done, where we can find the concept of a processor which handles the translation. That is, the LUP is a processor which is allocated to a node, it governs the update translation at this node and can move on the tree. In our architecture, obviously an update is first issued against the root where a LUP stays initially. The general form of (an) input to a LUP is a quadruple, the precise definition of which is given in section 4.3.1. The LUP output is the translated update statement(s) against its one level higher node(s) if the translation is possible and if not the announcement of the impossibility of translation. The LUP (if necessary the LUP is duplicated.) comes up to the higher node(s) when the translation was succeeded. The same action will be taken until all LUP's in the tree have reached to certain leaves. The precise description of the LUP actions is given in section 4.3.2 and 4.3.3. Next, if all LUP's in leaves succeed in executing updates, then they come down to the root where the expected update result will be obtained. This update execution is described in section 4.4 in detail. It should be noted here that the formal description of the meaning of a view is essentially taken into account in characterizing the LUP function.

As a summary, the LUP concept with the view defining tree is valid particuraly from the following points of view:

(a) The LUP concept with the view defining tree enables us to handle the view update translation just by looking at relations of one level higher and lower. This means that at most a finite number of actions of LUP's are definable (because only five relational operations are used to define the view defining tree and only two types of updates (deletion, insertion) are considered), while those actions are capable enough to handle the translation.

(b) LUP's process the view update translation step by step, which in turn means that the LUP concept with the view defining tree can handle the translation in a unified manner.

(c) The behavior of LUP's on the view defining tree just corresponds to a way of implementing the view update translation mechanism.

4.2 Theoretical Foundations

Before describing the function of LUP's in the following sections, we state a few theoretical foundations which are necessary to describe the function.

4.2.1 Induced Structural Integrity Constraint

Let us suppose that the expanded direct product of relation R(A1, A2, ..., An) and S(B1, B2, ..., Bp) is defined. Then the following integrity constraint should hold for $R \otimes S$:

$$\text{(E3)}\quad (\forall t, t'' \in \mathrm{dom}(R) \times \mathrm{dom}(S))((t, t'' \in R \otimes S)) \supset ((t[A1^\frown A2^\frown \ldots^\frown An]^\frown t''[B1^\frown B2^\frown \ldots^\frown Bp]) \in R \otimes S)).$$

Obviously this is induced from the syntactic nature among tuples of the expanded direct product view. Therefore we call this the induced structural integrity constraint of the view.

Now, let us suppose that a tuple delete statement D is issued against the view. By res(D, $R \otimes S$), we denote the expected result relation. By diff(D, $R \otimes S$), we denote the set of all tuples of the view which should be deleted by D, i.e. diff(D, $R \otimes S$) = $R \otimes S$ - res(D, $R \otimes S$). To effect this tuple deletion, D should be translated into two tuple delete statements D_R and D_S (one of those may be empty) against R and

S respectively. (Here we do not want to say anything about how the translation will be done.) However, the point is that if D were effected, then the update result, res(D, $R \otimes S$), should again satisfy the induced structural integrity constraint. That is, the following should hold:

$$\text{(E4)} \quad (\forall t, t' \in dom(R) \times dom(S))((t, t' \in res(D, R \otimes S)) \supset ((t[A1^\frown A2^\frown \ldots ^\frown An]^\frown t''[B1^\frown B2^\frown \ldots ^\frown Bp]) \in res(D, R \otimes S)))).$$

Now, the following is almost obvious:

<u>Theorem</u>

A delete statement D against $R \otimes S$ is translatable to the delete statement(s) against R and/or S (without causing any side-effect) if and only if (E4) holds.

We should note here that there is an exact correspondence between this theorem and the result of the characterization of the translatable delete statement D against $R \otimes S$ done from the semantic point of view [MASU79]. Because this investigation is valid in the following sections, a short summary is given below:

First, the meaning of $R \otimes S$ is formally defined by

$$\text{(E5)} \quad (\forall t \in dom(R) \times dom(S))(M_{R \otimes S}(t) \equiv M_R(t[A1^\frown A2^\frown \ldots ^\frown An]) \wedge M_S(t[B1^\frown B2^\frown \ldots ^\frown Bp])).$$

Second, the tuple delete statement D is issued against $R \otimes S$, because every tuple of diff(D, $R \otimes S$) has lost the meaning of $R \otimes S$.

Then the following is obtained by (E4) and the particularization rule of the quantification theory:

(E6) "$M_{R \otimes S}$(diff(D)) is false if and only if either (M_R(diff (D, R $\otimes$ S) [A1⌢A2⌢...⌢An]) is false or M_S(diff(D, R $\otimes$ S) [B1⌢B2⌢...⌢Bp]) is false or both."

Therefore the valid delete statement D against R $\otimes$ S should intend to delete either (a) any tuple of R $\otimes$ S which projection on A1⌢A2⌢...⌢An belongs to diff(D, R $\otimes$ S) [A1⌢A2⌢...⌢An], (b) any tuple of R $\otimes$ S which projection on B1⌢B2⌢...⌢Bp belongs to diff(D, R $\otimes$ S)[B1⌢B2⌢...⌢Bp], or (c) both. It is now clear that a delete statement D against R $\otimes$ S satisfies the induced structural integrity constraint if and only if D is either one of the above three statements. Notice that if a delete statement satisfies this constraint, then no side-effect occurs.

The same argument holds for insert statements.

4.2.2 Update Modification Rule

Let V be a view which is associated with a certain intermediate node of a view defining tree. Moreover, let us suppose that the one level lower relation of V is defined as a restriction V[A θ B] of it for certain A, B and θ.

Now, suppose a delete statement against V is D. Then one can modify D to D' such that (a) diff(D', V) $\supseteq$ diff(D, V) and (b) diff(D', V) $\cap$ V[A θ B]) = diff(D, V). This is called a tuple delete statement modification rule. This modification is possible because V is an intermediate (therefore virtual) relation. However, in order not to cause any side-effect, the modification should be minimal (see section 4.3.2.1). An example of applying this rule is seen in example 5.1.

For a tuple insert statement I, one can modify I whenever V is a direct product view. The rule is called the insert statement modification rule. (The modification should also be minimal by the same reason.) A simple example of this case is given in Fig.3, where the tuple insert statement I of inserting the pair (3,3) to R $\otimes$ S is modified to the statement of inserting a set of pairs $\{(1,3), (2,3), (3.3)\}$, by which modification the insertion requirement of inserting a pair (3,3) to the view R[A=B]S is effected.

4.2.3 Augmentation Rule

This rule is used to modify an insert statement against a projection view. Suppose an insert statement I is issued against the projection view R[A]. As will be shown in section 4.3.3.4, in principle, I is not translatable to the insert statement against R because of the semantic ambiguity. However, when it is possible to determine $u[\bar{A}]$(where $u \in$ dom(R)) from res(I, R[A]) for any t in diff(I, R[A]) such that t = u[A], then we can translate I to an insert statement against R which realizes the insertion in R[A].

For example, the statement of inserting a tuple (e5, d3, m4) is translatable to the insertion against E[D=D]M because the DEPT^1 value is always determined by the DEPT^2 value. (That is, those two values are always equal.)

4.3 LUP Functions

In this section we describe how LUP's behave.

4.3.1 Preliminaries

We associate a distinguished non negative integer with each node of a view defining tree (in a certain order). (An example is shown in Fig.1, where 0 is associated with the root and so on.) Generally, node n has one ancestor (i.e. the node of one level lower) except the root, and at most two descendant nodes (i.e. the nodes of one level higher). By anc(n), we denote the ancestor of node n, and by des(n), we denote the descendant of node n. (If there are two descendants, then by $\mathrm{des}_L(n)$ and $\mathrm{des}_R(n)$, we denote the upper left and the upper right node of node n.) By rel(n), we denote the relation defined at node n of the view defining tree. By reldef(n), we denote the defining expression of rel(n) in terms of its descendant relation(s). By U(n), we denote the update statement against rel(n), which is initially given by the user to the root and may be given by LUP's to the higher node(s).

General form of an input to a LUP is a quadluple (rel (anc(n)), rel(n), reldef(n), U(n)). For example, suppose a LUP stays at the root of the view EDM defining tree (Fig.1) and a delete statement D is issued against the view. Then the input to the LUP is (ϕ , edm, $\mathrm{EDM}=(\mathrm{E}[\mathrm{D}=\mathrm{D}]\mathrm{M})[\mathrm{EMP}\frown\mathrm{DEPT}^{1}\frown\mathrm{MGR}]$, D), where ϕ denotes an empty relation, i.e. the relevant node is the root.

The output of the LUP at node n is the translated update statement(s) against its one level higher node(s) if the translation is possible and if not the announcement of the impossibility of translation.

4.3.2 Deletions

Suppose a LUP stays at node n and U(n) is a delete statement D. The following states how the LUP functions in this case.

4.3.2.1 The Expanded Direct Product

Suppose rel(n)= rel(des_L(n)) $\otimes$ rel(des_R(n)). Now, two types of inputs to LUP are considerable corresponding to (i) anc(n)=ϕ (i.e. n is the root) and (ii) anc(n)$\neq\phi$ (i.e. n is an intermediate node).

(i) Case of anc(n)=ϕ

In this case, the input to the LUP is a quadruple (ϕ, rel(n), rel(des_L(n)) $\otimes$ rel(des_R(n)), D).

<u>Action D-1-1</u>

"Check whether (E4) holds. If so, then output the translated update statements D_{Lout} and D_{Rout} to des_L(n) and des_R(n) respectively. (One of the outputs may be empty.) D_{Lout} and D_{Rout} are determined as stated in section 3. In this case the LUP moves to des_L(n) (des_R(n)) whenever D_{Lout} (D_{Rout}) is not empty. (If both are non empty then the LUP is duplicated and those move to des_L(n) and des_R(n). Otherwise the LUP announces that the translation is impossible."

(ii) Case of and(n)$\neq\phi$

The imput to the LUP is a quadruple (rel(anc(n)), rel(n), rel(des_L(n)) $\otimes$ rel(des_R(n)), D).

<u>Action D-1-2</u>

"Check whether (E4) holds. If so, then do the same as stated in Action D-1-1. Otherwise, begin to apply the update modification rule (in section 4.2.2) to D so that the LUP may possibly find out D' which is a minimal modification of D. (Here the term "minimal" means that there does not exist any other modification D" of D such that res(D", rel(n)) satisfies (E4) and diff(D', rel(n)) $\supsetneq$ diff(D", rel(n)) $\supsetneq$ diff(D, rel(n)).) This condition is searched exhaustively. If such D' is found, then do the same as stated in Action D-1-1. Otherwise the LUP announces the impossibility of translation."

4.3.2.2 Union

Suppose reldef(n)= rel(des_L(n))$\cup$rel(des_R(n)). The formal description of rel(n) is defined by

(E7) $(\forall t \in \mathrm{dom}(\mathrm{rel}(\mathrm{des}_L(n))) \times \mathrm{dom}(\mathrm{rel}(\mathrm{des}_R(n))))(M_{\mathrm{rel}(n)}(t) \equiv M_{\mathrm{rel}(\mathrm{des}_L(n))}(t) \cup M_{\mathrm{rel}(\mathrm{des}_R(n))}(t))$.

Then the following holds:

(E8) "For any tuple t, $M_{\mathrm{rel}(n)}(t)$ is false if and only if both $M_{\mathrm{rel}(\mathrm{des}_L(n))}(t)$ and $M_{\mathrm{rel}(\mathrm{des}_R(n))}(t)$ are false."

This means that to effect D against rel(n), all tuples of diff(D, rel(n)) should be deleted both from rel(des_L(n)) and rel(des_R(n)). Let D_{Lout} and D_{Rout} be the tuple delete statements against the left and the right descendant relation which deletes diff(D, rel(n)). Those two tuple delete statements are always definable.

Action D-2

"Output D_{Lout} and D_{Rout} to des_L(n) and des_R(n) respectively. The LUP is duplicated and those move to corresponding descendant nodes."

4.3.2.3 Difference

Suppose $rel(n) = rel(des_L(n)) - rel(des_R(n))$.

The meaning of rel(n) is formally defined by

(E9) $(\forall t \in dom(rel(des_L(n))) \times dom(rel(des_R(n))))(M_{rel(n)}(t) \equiv M_{rel(des_L(n))}(t) \wedge \sim M_{rel(des_R(n))}(t))$.

Then,

(E10) "For any tuple t, $M_{rel(n)}(t)$ is false if and only if $M_{rel(des_L(n))}(t)$ is false or $M_{rel(des_R(n))}(t)$ is true or both."

This means that to delete a tuple t from rel(n), one can delete t from $rel(des_L(n))$ or insert t to $rel(des_R(n))$ or doing both simultaneously. There is no mathematical reason to decide which alternative should be chosen. However, notice that those have different meanings. That is, the first one means that t has lost the meaning of $rel(des_L(n))$, while the second one means that t becomes to satisfy the meaning of $rel(des_R(n))$. Therefore, essentially the LUP here can not choose arbitrarily and should ask to the user which one should be chosen.

Action D-3

"Ask to the user which alternative should be chosen. According to the answer, the LUP (if necessary it is duplicated) moves to the descendant node(s)."

However, if we do not want to have a LUP-user conversation, then we should give up to translate the delete statement:

Action D-3'

"Announce that the translation is impossible."

Notice again that the LUP should not be allowed to choose an alternative arbitrarily because it may cause semantic inconsistency.

4.3.2.4 Projection

Suppose rel(n)= rel(des(n))[A], where A is a list of attributes.

The meaning of rel(n) is formally defined by

(E11) $(\forall t \in \mathrm{dom}(\mathrm{rel}(n)))(M_{\mathrm{rel}(n)}(t) \equiv (\exists u \in \mathrm{dom}(\mathrm{rel}(\mathrm{des}(n))) u[A]= t \wedge M_{\mathrm{rel}(\mathrm{des}(n))}(u)))$.

Then,

(E12) "For any tuple t, $M_{\mathrm{rel}(n)}(t)$ is false if and only if for every u of dom(rel(n)), if u[A]= t then $M_{\mathrm{rel}(\mathrm{des}(n))}$ (u) is false."

This means that to delete a tuple t from rel(n), it is sufficiently enough to delete all tuple u of rel(des(n)) such that u[A]= t. In this case D is always translatable to D_{out} against rel(des(n)) straightforwardly so that it deletes all desired tuples. The LUP takes the following action:

Action D-4

"Output D_{out} as the delete statement against rel(des(n)) and the LUP moves to des(n)."

4.3.2.5 Restriction

Suppose rel(n)= rel(des(n))[$A \theta B$], providing A and B are union-compatible.

The meaning of it is formally defined by

(E13) $(\quad t \in \mathrm{dom}(\mathrm{rel}(n)) \; (M_{\mathrm{rel}(n)}(t) \equiv (t[A]\,\theta\, t[B]) \wedge$

$M_{\mathrm{rel}(\mathrm{des}(n))}(t))$.

Then,

(E14) "For any tuple t, $M_{\mathrm{rel}(n)}(t)$ is false if and only if,

if $t[A]\,\theta\, t[B]$, then $M_{\mathrm{rel}(\mathrm{des}(n))}(t)$ is false.

This indicates the translation of D to the delete statement D_{out} against rel(des(n)), which is directly obtained by using the query modification method of [STON75]. The LUP action here is stated as follows:

Action D-5

"Output D_{out} as the delete statement against rel(des(n)) and the LUP moves to des(n)."

4.3.3 Insertions

Suppose a LUP stays at node n and U(n) is an insert statement I. The following states how the LUP functions in this case.

4.3.3.1 The Expanded Direct Product

Suppose $\mathrm{rel}(n) = \mathrm{rel}(\mathrm{des}_L(n)) \otimes \mathrm{rel}(\mathrm{des}_R(n))$.

(i) Case of $\mathrm{anc}(n) = \phi$

In this case, the input to the LUP is a quadruple (ϕ, rel(n), $\mathrm{rel}(\mathrm{des}_L(n)) \otimes \mathrm{rel}(\mathrm{des}_R(n))$, I). From the meaning point of view, the following is observed.

(E15) "For any t, $M_{\mathrm{rel}(n)}(t)$ is true if and only if both

$M_{\mathrm{rel}(\mathrm{des}_L(n))}(t[\alpha])$ and $M_{\mathrm{rel}(\mathrm{des}_R(n))}(t[\beta])$ are true."

This means that if we want to insert a tuple t to rel(n), then we should insert $t[\alpha]$ and $t[\beta]$ to $\mathrm{rel}(\mathrm{des}_L(n))$ and $\mathrm{rel}(\mathrm{des}_R$

(n)) respectively, where α and β denote the list of all attributes of rel(des_L(n)) and rel(des_R(n)) respectively.

Action I-1-1

"Check whether (E3) holds for res(I, rel(n)). If so, output I_{Lout} and I_{Rout} to des_L(n) and des_R(n) respectively, where I_{Lout} and I_{Rout} are the statements of inserting diff(I, rel(n)) [α] and diff(I, rel(n))[β] to rel(des_L(n)) and rel(des_R(n)) respectively. The LUP is duplicated and each moves to des_L(n) and des_R(n) respectively. If not, announce that the translation is impossible."

(ii) Case of anc(n)$\neq \phi$

In this case, the update modification rule is applicable.

The action of LUP is as follows:

Action I-1-2

"Check whether (E3) holds for res(I, rel(n)). If so, do the same as stated in Action I-1-1. Otherwise, begin to apply the update modification rule (in section 4.2.2) to I and find out I' which is the minimal modification of I. (The term minimal is defined analogously as did in Action D-1-2. In this case, such I' is always found.) Then do the same as stated in Action I-1-1."

4.3.3.2 Union

Suppose rel(n)= rel(des_L(n))$\cup$rel(des_R(n)).

In this case, the following holds from the meaning point of view:

(E16) "For any t, $M_{rel(n)}(t)$ is true if and only if either $M_{rel(des_L(n))}(t)$ or $M_{rel(des_R(n))}(t)$ or both are true."

This means that to effect I against rel(n), diff(I, rel(n)) should be inserted in either rel(des_L(n)) or rel(des_R(n)) or both. But notice that there is no mathematical reason which alternative should be chosen. This is completely a semantic issue as discussed already in the case of deletion against the difference view. (See section 4.3.2.3.)

Action I-2

"Ask to the user which alternative should be chosen. According to the answer, the LUP (if necessary duplicated) outputs the translated insert statement(s) and moves to the relevant descendant node(s)."

If we do not want to have such LUP-user conversation, the following is taken:

Action I-2'

"Announce that the translation is impossible."

Notice here that the LUP should not be allowed to choose an alternative arbitrary because it may cause semantic inconsistency.

4.3.3.3 Difference

Suppose reldef(n)= rel(des_L(n))- rel(des_R(n)).

By (E9) we obtain the following:

(E17) "For any t, $M_{rel(n)}(t)$ is true if and only if $M_{rel(des_L(n))}(t)$ is true and $M_{rel(des_R(n))}(t)$ is false."

This means that to effect I against rel(n), diff(I, rel(n)) should be inserted in rel(des_L(n)) and it should be deleted from rel(des_R(n)). The insert and the delete statement against those two relations respectively are definable straightforwardly. (We denote those by I_{Lout} and I_{Rout} respectively.)

Action I-3

"Output I_{Lout} and I_{Rout} to des_L(n) and des_R(n) respectively. The LUP is duplicated and each moves to the corresponding descendant node."

4.3.3.4 Projection

Suppose rel (n)= rel(des(n))[A], where A is a list of attributes. By (E11) we have the following:

(E18) "For any t, $M_{rel(n)}$(t) is ture if and only if there exists a tuple u of dom(rel(des(n))) such that u[A]= t and $M_{rel(des(n))}$(u) is true."

This means that to insert a tuple t in rel(n), the LUP should find out a tuple u satisfying (E18). However, there may not be possible to find out an unique u. The uniqueness is essential because different tuple represents different occurance of the entities and the relationships among them. Therefore, in principle, insert statements against the projection view is not translatable except the statement to which the augmentation rule is applicable. (See section 4.2.3.) In order to check whether the augmentation rule is applicable, the LUP should see the definition (i.e. intention) of rel(des(n)) in this case. Now the action of the LUP is made clear.

Action I-4

"Check whether the augumentation rule is applicable. If so, output the augmented insert statement I_{out} and moves to des(n). Otherwise, announce that the translation is impossible."

The problem here deeply relates to the null value issue in a relational data model (CODD75, ZANI77) which is another aspect of the view update problems (MASU79). But here we do not discuss this problem further.

4.3.3.5 Restriction

Suppose rel(n)= rel(des(n))(Aθ B), providing A and B are union-compatible. By (E13), we obtain the following:

(E19) "For any t, $M_{rel(n)}$(t) is true if and only if t(A) θ t(B) and $M_{rel(des(n))}$(t)are true."

This indicates that we can translate I into the inserte statement I_{out}, which inserts diff(I, rel(n)) to rel(des(n)), according to the query modification method of (STON75).

Action I-5

"Output I_{out} to rel(des(n)) and moves to des(n)."

4.4 Update Execution

4.4.1 Execution Control

In section 4.1 and 4.3, we have described how LUP's behave. After a certain period of time, if all update translations are succeeded, then all LUP's stay at certain leaves. By the "LUP orbit", we mean a set of all paths, each of which begins at the root and ends at a certain leaf where a LUP reached. Then the LUP orbit consists a subtree of the view defining tree. (For example, as shown in Fig.4(a), the LUP orbit is the straight line from node 0 to node 4, i.e. the set (0, 1), (1, 2)), (2, 4) , in Example 1 of section 5.1.) If the LUP orbit is a straight line, then the update execution is straightforwardly done in such a way that first execute the update statement against the leaf relation, and then compute the extension of the view except using the new value of the leaf relation. However, if the LUP orbit is not a straight line but a proper subtree, then LUP's which stay at the upper nodes of a branching node should communicate each other to synchronize the execution of the update statement. For example, in Example 2 of section 5, LUP's at node 3 and 4 should be synchronized in the sense that the restriction operation EDDM $\left(\mathrm{DEPT}^1 = \mathrm{DEPT}^2\right)$ is executable after both LUP's come down to node 2. In order to realize the synchronization, we associate a "milestone" at each branching node of the LUP orbit. (In Example 2, as shown in Fig.4(b), node 2 has a milestone.) When a LUP first comes down to the node with a milestone, then the LUP should wait the pair LUP to come down the node. Except synchronization, the execution is done in ordinary manner.

4.4.2 Additive Side-effect Control

The additive side-effect may occur when one wants to insert a set of tuples to a certain view. In our framework, it may happen if there exists a LUP which took Action I-1-2 with the insert statement modification rule. However, we can observe that the rule is essentially necessary to effect a tuple insertion to a certain view which is derived from an expanded direct product view. The modified insert statement inserts more tuples than those which are inserted by the original statement. The problem here is to investigate how the additionally inserted tuples interact with the view which is derived from the expanded direct product view. (In Example 2 of section 5.2, the quadruple (e4, d3, d3, m4) should be distinguished among the additionally inserted tuples, which causes the additive side-effect because it can pass the restriction EDDM[$DEPT^1 = DEPT^2$].) As investigated in [MASU79], the additive side-effect issue is rather a structural issue than a semantic one in the sense that the additionally inserted tuple has correct meaning. Therefore, whether the use of the insert statement modification rule causes any additive side-effect should be checked exhaustively. That is, when a LUP uses the insert statement modification rule, the LUP associates a star mark(*) to the node (of the LUP orbit) to indicate the use of it. In update execution, LUP's should check whether the additive side-effect occur or not if they come down below the star marked node.

5. Examples

Let us now demonstrate how the update translations are done under the LUP concept with the view defining tree.

5.1 Example 1 -Deletion-

Suppose the view EDM is defined as shown in Fig.1 and the extensions are as given in Fig.2. Suppose a delete statement D_0 is issued against edm:

(E20) D_0: "Delete every tuple from edm having d1 and m1 as $DEPT^2$ and MGR value respectively."

Initially, a LUP stays at node 0 and then translate D_0 to D_1, which is the delete statement against e(d= d)m, and moves to node 1 (Action D-4):

(E21) D_1: "Delete every tuple from d(d= d)m having d1 and m1 as $DEPT^2$ and MGR value respectively."

Next, the LUP at node 2 decides to translate D_1 to D_2, which is the delete statement against eddm, and moves to node 3 (Action D-5):

(E22) D_2: "Delete every tuple from eddm having d1 and m1 as $DEPT^2$ and MGR value respectively and having the same $DEPT^1$ and $DEPT^2$ value."

The LUP at node 3, first examine whether (E4) holds for res(D_2, eddm). However the LUP see that it does not hold in this case. Then the LUP tries to apply the update modification rule (section 4.2.2) to D_2. In this case, the LUP succeeds in finding out such a statement which is D_3. (Notice that D_3 is obtained just by loosing the qualification part of D_2 such that the

condition of "having the same $DEPT^1$ and $DEPT^2$ value" is omitted)
(Update modification rule):

(E23) D_3: "Delete every tuple from eddm having d1 and m1 as $DEPT^2$ and MGR value respectively."

Then the LUP examines D_3 and translates it to the delete statement D_4 against dm and moves to node 4 (Action D-1-2):

(E24) D_4: "Delete every tuple from dm having d1 and m1 as $DEPT^2$ and MGR value respectively."

Now, the LUP recognized that it is in a leaf. Therefore it begin to execute D_4. As stated in section 4.4, the LUP comes down to the root where it can show the desired result.

5.2 Example 2 -Insertion-

Suppose the view EDM is defined as shown in Fig.1 and the extensions are as given in Fig.2. Suppose an insert statement I_0 is issued against edm(cf. section 3).

(E25) I_0: "Insert a tuple (e5, d3, m4) to edm."

The LUP stayed initially at node 0 translates it to I_1, which is an insert statement against e[d= d]m. This is possible because of the augumentation rule (section 4.2.3) and moves to node 1 (Action I-4):

(E26) I_1: "Insert a tuple (e5, d3, d3, m4) to e[d= d]m."

Then the LUP at node 1 translates I_1 to I_2, which is an insert statement against eddm, and moves to node 2 (Action I-5):

(E27) I_2: "Insert a tuple (e5, d3, d3, m4) to eddm."

Now the LUP first sees that (E4) does not hold for res(I_2, eddm). Therefor the insertion modification rule is used so that

I_2 is translated into two insert statements I_3 against ed and I_4 against dm:

(E28) I_3: "Insert a tuple (e5, d3) to ed."

(E29) I_4: "Insert a tuple (d3, m4) to dm."

As stated in section 4.2, node 2 is associated with a star mark. The LUP is duplicated and each of which comes up to node 3 and 4.

Now the LUP at node 3 and the LUP at node 4 begin to execute I_3 and I_4 respectively. Both LUP's come down and synchronized at node 2. Then two LUP's are marged into one, and execute the restriction EDDM$[\text{DEPT}^1 = \text{DEPT}^2]$. The LUP comes down to node 1 and because the upper node 2 is associated with the star mark, the LUP begins to check whether there exists an additionally inserted tuple which can pass the restriction. If there does not, then proceed execution, otherwise announce that the translation is impossible (The additive side-effect occurs.). In this case, as stated in section 4.4.2, the additionally inserted quadruple (e4, d3, d3, m4) passed the restriction. Therefore the translation of our insertion is impossible because of the additive side-effect.

6. Concluding Remarks

The LUP Which is a vehicle of implementing a view support subsystem is introduced and the architecture of implementing the subsystem is described in detail. Through the investigation, the followings are observed with relation to the update translatability problem.

(a) In characterizing the actions taken by a LUP, the meaning of a view played an essential role. The ambiguity of the update translations is also characterized under it.

(b) The translatability and the translation mechanism of tuple delete statements are completely characterized from the meaning point of view. While this is not true for those of tuple insert statements. That is, the additive side-effect can not be controlled from this point of view, because this comes from the structural nature of the relational algebra. The null value issue closely relates to this problem.

(c) The LUP concept provides a very strong tool to distinguish such semantic and structural aspects of the view update problems.

Acknowledgement

The author is thankful to Prof. S. Noguchi who gave him an opportunity to work on this subject. This work is partly supported by the Science Foundation Grant of the Ministry of Education of Japan, Grant No. 468008.

Reference

(BANC79) Bancilhon, F., "Supporting View Updates in relational data base," Proc. IFIP TC-2 Working Conference on Data Base Architecture, 1979, pp. 198-219.

(CHAM75) Chamberlin, D.D., J.N.Gray and I.L. Traiger, "Views, authorization, and lockingin a relational database system," Proc. AFIPS NCC, 1975, pp. 425-430.

(CODD70) Codd, E.F., "A relational model of data for large shared data banks," CACM 13, 6, 1970, pp. 377-387.

(CODD72) Codd, E.F., "Relational completeness of database sublanguages," In Data Base Systems, Courant Computer Sci. Symp. 6, R. Rustin, Ed., Prentice-Hall, Englewood Cliffs, 1972, pp. 65-97.

(CODD74) Codd, E.F., "Recent investigations in a relational database system," Information Processing 74, North-Holland Pub. Co., Amsterdam, 1974, pp. 1017-1021.

(CODD75) Codd, E.F., "Understanding Relations," IBM Res. Report, July 10, 1975.

(DATE71) Date, C.J.and P.Hopewell, "File definition and logical data independence," Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access and Control, pp. 117-138 (1971).

(DAYA78) Dayal, U. and P.A.Bernstein, "On the updatability of relational views," Proc. Fourth VLDB Conf., 1978, pp. 368-377.

(FERN76) Fernandez, E.B. and R.C.Summers, "Integrity aspects of a shared data base," Proc. AFIPS NCC, 1976, pp. 819-827.

(KIM79) Kim, W., "Relational database systems," Computing Surveys 11, 3, pp. 185-211 (1979).

(MASU79) Masunaga, Y., "On a semantic aspect of view updates in a relational database system," submitted to the 8th World Computer Congress (1979).

(OSMA79) Osman, I.M., "Updating defined relations," Proc. AFIPS NCC, 1979, pp. 733-740.

(PAOL77) Paolini, P., and G.Pelagatti, "Formal definition of mappings in a database," ACM SIGMOD, Proc. of the Intl. Conf. on Management of Data, 1977, pp. 40-46.

[STON75] Stonebraker, M., "Implementation of integrity constraints and views by query modification," Proc. 1975 SIGMOD Conf., ACM, N.Y., pp. 65-78.

[TSIC77] Tsichritzis, D.C. and F.H.Lochovsky, "Data Base Management Systems (book)," Academic P., 1977.

[TSIC78] Tsichritzis, D.C. and A.Klug (Ed.)," The ANSI/X3/SPARC DBMS framework, report of the study group on database management systems," Inform. Systems 3, pp. 173-191 (1978).

[ZANI77] Zaniolo, C., "Relational views in a data base system support for queries, "IEEE COMPSAC 77, 1977, pp. 267-275.

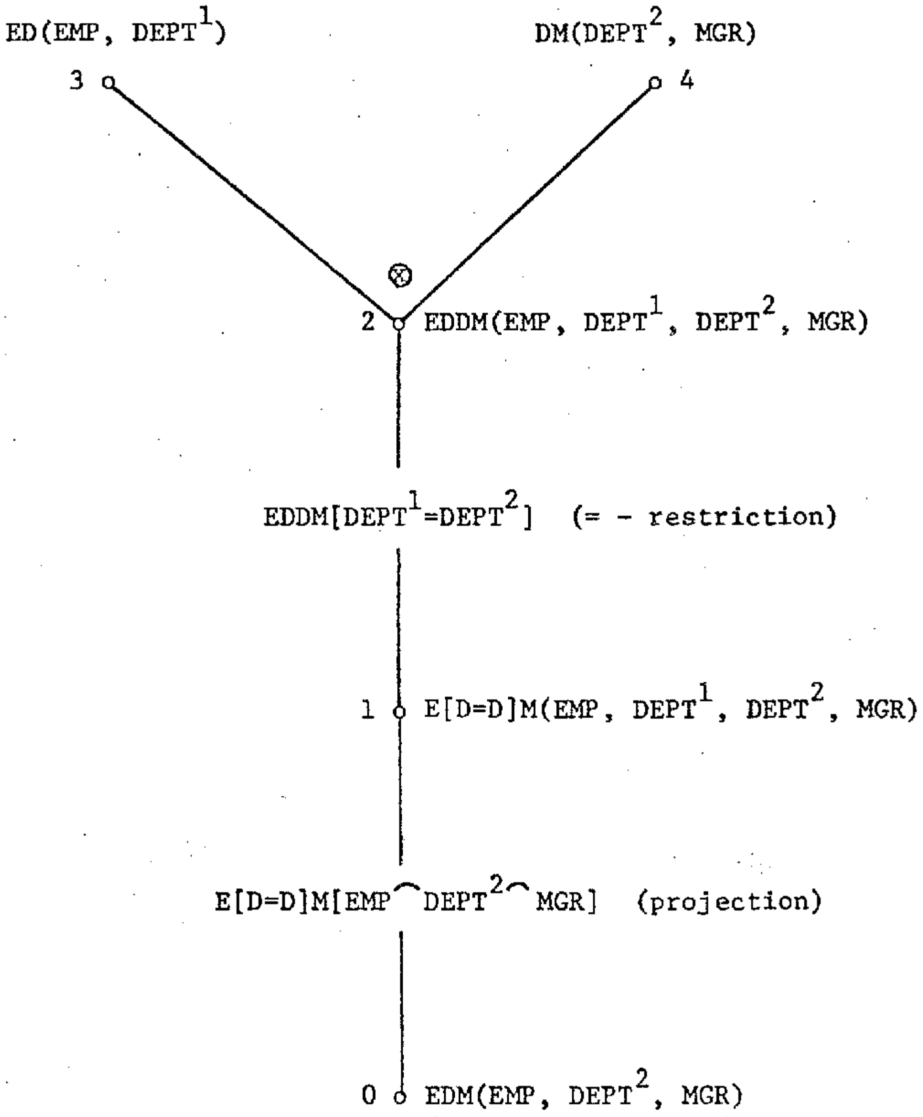

ED(EMP, DEPT^1)
DM(DEPT^2, MGR)
3
4
⊗
2
EDDM(EMP, DEPT^1, DEPT^2, MGR)
EDDM[DEPT^1=DEPT^2] (= - restriction)
1
E[D=D]M(EMP, DEPT^1, DEPT^2, MGR)
E[D=D]M[EMP⌢DEPT^2⌢MGR] (projection)
0
EDM(EMP, DEPT^2, MGR)

Fig. 1. The view defining tree of the view EDM.

ED

ed:	EMP	DEPT^1
	e1	d1
	e2	d1
	e3	d2
	e4	d3

DM

dm:	DEPT^2	MGR
	d1	m1
	d1	m2
	d2	m3

EDDM

eddm:	EMP	DEPT^1	DEPT^2	MGR	
	e1	d1	d1	m1	*
	e1	d1	d1	m2	*
	e1	d1	d2	m3	
	e2	d1	d1	m1	*
	e2	d1	d1	m2	*
	e2	d1	d2	m3	
	e3	d2	d1	m1	
	e3	d2	d1	m2	
	e3	d2	d2	m3	*
	e4	d3	d1	m1	
	e4	d3	d1	m2	
	e4	d3	d2	m3	

e[d=d]m: This consists of five tuples of eddm marked *.

EDM

edm:	EMP	DEPT^2	MGR
	e1	d1	m1
	e1	d1	m2
	e2	d1	m1
	e2	d1	m2
	e3	d2	m3

Fig. 2. Extensions of ED, DM, EDDM, E[D=D]M and EDM.

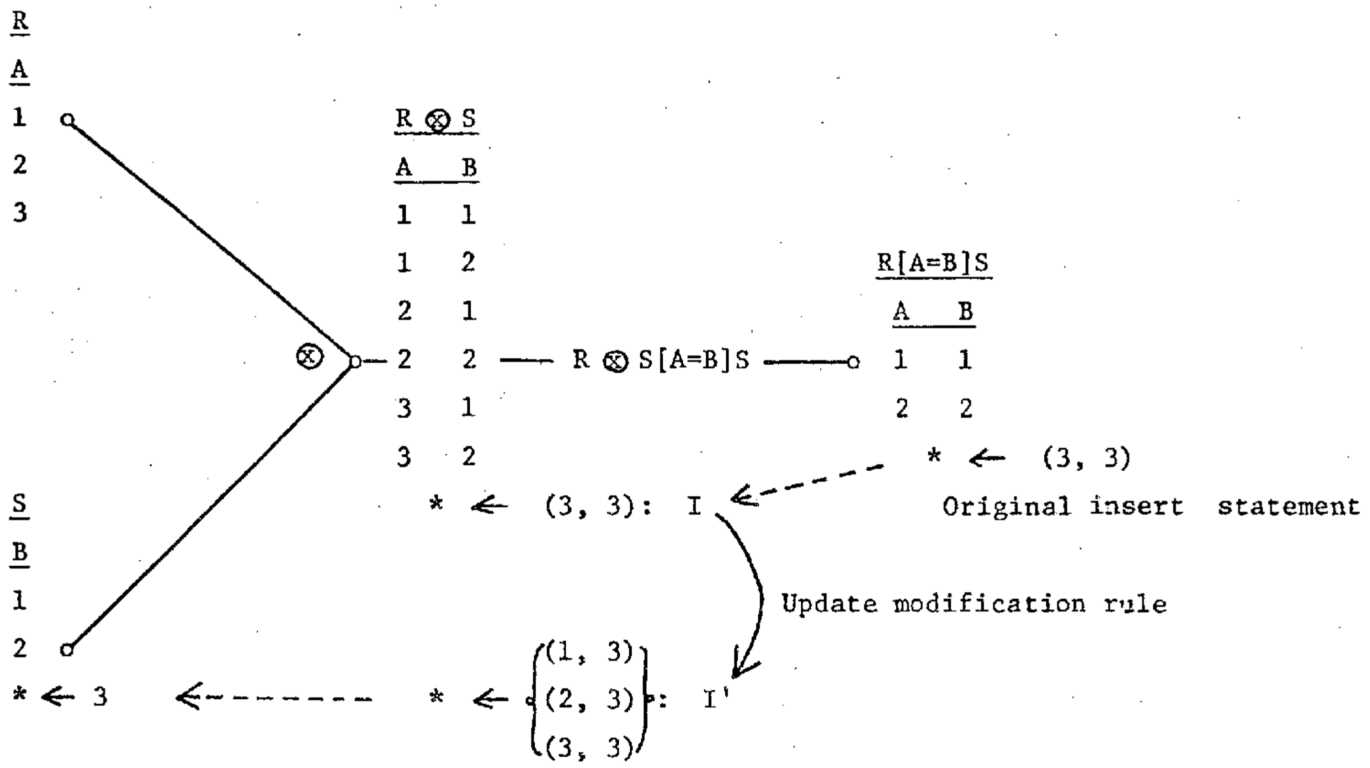
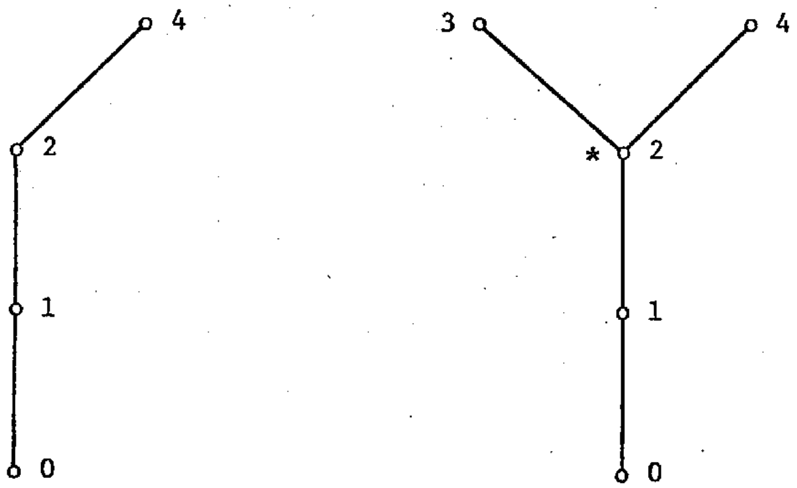


Fig. 3. Update modification rule.



(a) Case of Example 1.

(b) Case of Example 2.

Fig. 4. The LUP orbits.

関係データベースの4NF D-tree スキーマ

北海道大学・工学部
田中 譲

1. 序論

関係データベースのスキーマの論理設計法は設計基準の違いによって，独立成分法と分解法に大別される．一般に独立成分法では，3NF 以上の正規形スキーマを設計することはできない。例えば，R(A,B,C) において AB→C，C→B が成立するとき，これに対するスキーマは {R} となり，R を R_1(AC) と R_2(BC) の BCNF の関係に分解することは許されない。さらに，独立成分法で得られたスキーマでは，1つの情報が重複して何箇所にも格納されることがあり得る。例えば S(A,B,C,D) において，AB→C，A→D，B→D のとき，独立成分法で得られるスキーマは {S_1(ABC)，S_2(A,D)，S_3(B,D)} となるが，A と D の間の関係 S[A,D] は，S_2(A,D) と (S_1(ABC) * S_3(B,D))[A,D] の 2 通りの方法で得ることができる．

これに対し，分解法で得られたスキーマは，もとの関係における従属関係をすべてスキーマ中に反映することが一般に不可能である．例えば，先の例 R(A,B,C) (AB→C，C→B) に対する分解法による設計結果は {R_1(AC)，R_2(BC)} となるが，このスキーマに反映されている従属関係は C→B のみで，AB→C は反映されていない。この結果，R(A,B,C) によって格納し得る情報の型と，{R_1(A,C)，R_2(B,C)} によって格納し得る情報の型は異なることになり，これを一致させるためには，後者のスキーマにおいて，AB→C を意味制約として，付加的に課す必要がある．

このように，両方の設計法には，一長一短があるが，2つの設計法は，その設計基準が各々独立性，統合性という近代データベースの2大概念に基づいているといえる。すなわち，独立成分法では，{R_1(A,C)，R_2(BC)} における AB→C のように，関係にまたがる意味制約を避け，スキーマの各構成要素の独立性を高めることを目標にしている．これに対し，分解法では，{S_1(A,B,C)，S_2(A,D)，S_3(B,D)} におけるような情報の冗長性も避け，さらに，構成要素を単純な形にすることを目指している．

著者は，意味制約が従属関係だけに限らないこと，従って，たとえ従属関係がすべてスキーマに反映できたとしても，スキーマの構成要素間にまたがる意味制約は依然として残ると考えられることから，統合性を独立性よりも重視し，分解法の立場をとっている．例えば，先の $S(A,B,C,D)$ では $\{S_1(A,B,C), S_2(A,D), S_3(B,D)\}$ と独立成分に分解したとしても，$S_2(A,D) \subseteq (S_1(A,B,C) * S_3(B,D))[A,D]$ 等の制約があり，S_1, S_2, S_3 を独立に更新することはできない。

従来の分解法アルゴリズムは，結果として得られるスキーマの正規性のみを設計条件として課していたので［FAGI77］，著者による研究［TANA77］［TANA79］を除いて，統合成に関する考察はされていない．本研究では，［TANA79］では 4NF に関して不完全にしか解かれていなかった情報の非冗長性の問題を解決すると共に，独立性の問題をも考慮し，この問題に対する最適近似解を与える．

2. 関係モデルとスキーマ設計

2.1 関係モデル．

有限集合 Ω を属性集合と名付け，集合 D を値集合と名付ける。全関数 $\mu: \Omega \to D$ をタプルと名付ける．D^{Ω} で $\mu: \Omega \to D$ なるすべてのタプルの集合を表わす．D^{Ω} の部分集合 R を Ω 上の関係と名付け，$R(\Omega)$ と表わす．$\Omega' \subset \Omega$ に対し μ の Ω' への制限を $\mu|_{\Omega'}$ で表わす。条件

$$\forall \mu\ \forall \nu \in R(\Omega)\ ((\mu|_X = \nu|_X) \supset (\mu|_Y = \nu|_Y))$$

が成立するとき，$X \to Y$ なる関数従属関係（FD）が成立するという。$\Omega' \subset \Omega$ に対し，$\{\mu|_{\Omega'} \mid \mu \in R(\Omega)\}$ を $R(\Omega)$ の Ω' へのプロジェクションと呼び $R(\Omega)[\Omega']$ で表わす．2つの関係 $R(\Omega_1) \subset D^{\Omega_1}$ と $S(\Omega_2) \subset D^{\Omega_2}$ に対し，$D^{\Omega_1 \cup \Omega_2}$ 上の関係 $\{\rho \mid \exists \mu \in R, \exists \nu \in S, (\rho|_{\Omega_1} = \mu) \wedge (\rho|_{\Omega_2} = \nu)\}$ を R と S のジョインといい $R * S$ で表わす．ジョインは可換かつ結合律を満足する．$\mathcal{R} = \{R_1, R_2, \cdots, R_n\}$ に対し

$$*\mathcal{R} = R_1 * R_2 * \cdots * R_n$$

と定義する．

条件.

$$R[\Omega'] = R[\Omega_1] * R[\Omega_2]$$

が $\Omega' \subset \Omega$ に対して成立するとき. Rにおいて $\Omega_1 \cap \Omega_2 \twoheadrightarrow \Omega_1$ in Ω' なる多値従属関係(MVD)が成立するという。

条件

$$R[\Omega'] = R[\Omega_1] * R[\Omega_2] * \cdots * R[\Omega_n]$$

が成立するとき. Rは ジョイン従属関係(JD)を満足するといい, $*\{\Omega_1, \Omega_2, \cdots, \Omega_n\}$ でこのJDを表わす。

MVDとJDの定義において. Ω'は 場(context)と呼ばれ 従属関係fの場は cont(f)で表わされる。 cont(f) = Ω のとき. fはRにおいて 大域的であるという。有限集合Γを 従属関係の集合とする. Γのすべての要素が 関係Rにおいて成立することを

$$R(\Omega) \quad \text{sat} \quad \Gamma$$

と表わす. ΓとΓ'が

$$\forall R \quad (R \text{ sat } \Gamma) \supset (R \text{ sat } \Gamma')$$

を満足するとき. $\Gamma \vdash \Gamma'$と表わす。 $\Omega' \subset \Omega$とし. ΓのΩ'への制限を

$$\Gamma|_{\Omega'} = \{ f \mid \Gamma \vdash \{f\}, \text{cont}(f) \subset \Omega' \}$$

と表わす.

2.2. スキーマ設計の設計基準

Ω, Γを各々. 属性集合, 従属関係集合とする. Ω, Γに対するスキーム[Ω, Γ]を

$$[\Omega, \Gamma] = \{ R \mid R \subset D^{\Omega}, R \text{ sat } \Gamma \}$$

と定義する。 $[\Omega', \Gamma']$ が $\Omega' \subset \Omega$ かつ $\Gamma' = \Gamma|_{\Omega'}$ のとき、$[\Omega', \Gamma']$ は $[\Omega, \Gamma]$ の部分スキームであるという。

$\mathcal{S}(\Omega_0, \Gamma_0)$ を $[\Omega_0, \Gamma_0]$ の部分スキームの集合とし スキーム集合 $\mathcal{S}(\Omega_0, \Gamma_0)$ のジョインを

$$* \mathcal{S}(\Omega_0, \Gamma_0) = \{ *_{[\Omega', \Gamma'] \in \mathcal{S}} R \mid R \in [\Omega', \Gamma'] \}$$

と定義する。

スキーム集合 $\mathcal{S}(\Omega_0, \Gamma_0)$ が、汎関係 $[\Omega_0, \Gamma_0]$ のスキーマになるためには、

$$\forall R \ (R \in [\Omega_0, \Gamma_0]) \supset (R \in * \mathcal{S}(\Omega_0, \Gamma_0))$$

が少くとも成立しなければならない。この条件は、

（情報格納可能性）

$$[\Omega_0, \Gamma_0] \subseteq * \mathcal{S}(\Omega_0, \Gamma_0)$$

と書ける。

定理 2.1.

$$[\Omega_0, \Gamma_0] \subseteq * \mathcal{S}(\Omega_0, \Gamma_0)$$

$$\text{iff} \ \ \forall R \in [\Omega_0, \Gamma_0], \ \ R = *_{[\Omega', \Gamma'] \in \mathcal{S}(\Omega_0, \Gamma_0)} R[\Omega']$$

（証明）

1. if 部は自明。
2. only if 部。

$\mathcal{S}(\Omega_0, \Gamma_0) = \{ [\Omega_1, \Gamma_1], [\Omega_2, \Gamma_2], \cdots, [\Omega_n, \Gamma_n] \}$ とする。$[\Omega_0, \Gamma_0] \subseteq * \mathcal{S}$ より、

$$\forall R \in [\Omega, \Gamma] \ \ \forall i \ \ \exists R_i \in [\Omega_i, \Gamma_i] \quad R = R_1 * R_2 * \cdots * R_n$$

が成立する。性質

$$\forall i \quad R * R[\Omega_i] = R$$

より.

$$\forall i \quad R = R_1 * R_2 * \cdots * R_n * R[\Omega_i] = R_1 * R_2 * \cdots * R_n$$

が成立する. このことから.

$$\forall i \quad R_i \supset R[\Omega_i]$$

である. よって

$$\forall i \quad R = R_1 * R_2 * \cdots * R_{i-1} * R[\Omega_i] * R_{i+1} * \cdots * R_n$$

となる. $R[\Omega_i]$を R_i とみなして同様の議論をくりかえすことにより.

$$R = R[\Omega_1] * R[\Omega_2] * \cdots * R[\Omega_n]$$

が得られる. (証明終).

次に部分スキームの集合の間に冗長性に関する順序関係を導入することにする. $\mathcal{S}_1$, $\mathcal{S}_2$ を $[\Omega_0, \Gamma_0]$ の部分スキームの集合とする. 半順序関係を

$$\mathcal{S}_1 \leq \mathcal{S}_2 \ \text{iff} \ \forall [\Omega_1, \Gamma_1] \in \mathcal{S}_1 \ \exists [\Omega_2, \Gamma_2] \in \mathcal{S}_2 \ \text{s.t.} \ \Omega_1 \subset \Omega_2$$

と定義し, $\mathcal{S}_1 \leq \mathcal{S}_2$ かつ $\mathcal{S}_2 \leq \mathcal{S}_1$ のとき, $\mathcal{S}_1 \sim \mathcal{S}_2$ と定義する.
$\mathcal{S}_1 \leq \mathcal{S}_2$ で $\mathcal{S}_1 \nsim \mathcal{S}_2$ のとき, $\mathcal{S}_1 < \mathcal{S}_2$ と書く.
部分スキーム集合の族 $\mathcal{C}(\Omega_0, \Gamma_0)$ を

$$\mathcal{S}(\Omega_0, \Gamma_0) \in \mathcal{C}(\Omega_0, \Gamma_0)$$

$$\begin{aligned}
\text{iff} \quad & [\Omega_0, \Gamma_0] \subseteq *\mathcal{S}(\Omega_0, \Gamma_0) \\
& \text{and} \\
& \neg \exists \mathcal{S}'(\Omega_0, \Gamma_0) \\
& \qquad \text{s.t.} \quad 1)\ [\Omega_0, \Gamma_0] \leq *\mathcal{S}'(\Omega_0, \Gamma_0) \\
& \qquad \text{and} \quad 2)\ \mathcal{S}'(\Omega_0, \Gamma_0) \leq \mathcal{S}(\Omega_0, \Gamma_0) \\
& \qquad \text{and} \quad 3)\ \mathcal{S}'(\Omega_0, \Gamma_0) \nsim \mathcal{S}(\Omega_0, \Gamma_0).
\end{aligned}$$

と定義する。$\mathcal{C}(\Omega_0, I_0)$ は情報格納可能な極小スキーム集合の族である。

定理 2.2.

$$\mathcal{S}(\Omega, I) \in \mathcal{C}(\Omega_0, I_0)$$

iff

1) $\forall R \in [\Omega, I] \quad R = *_{[\Omega', I'] \in \mathcal{S}(\Omega, I)} R[\Omega']$

and

2) $\forall [\Omega', I'] \in \mathcal{S}(\Omega, I)$, $\mathrm{non}([\Omega', I'])$,

ただし.

$\mathrm{non}([\Omega', I']) = \exists R \in [\Omega', I']$, R は分解不能.

and

3). $\neg \exists [\Omega', I'] \in \mathcal{S}(\Omega, I)$, $\exists A \in \Omega'$,

$\forall R \in [\Omega, I]$

$$R = (*_{[\Omega'', I''] \in \mathcal{S} - \{[\Omega', I']\}} R[\Omega'']) * R[\Omega' - \{A\}].$$

(証明).

1). only if 部は自明.

2). if 部.

$\mathcal{S}(\Omega, I)$ は. 1). 2). 3) を満足するとする. 条件 1) より.

$$* \mathcal{S}(\Omega, I) \geq [\Omega, I]$$

である. よって. $\mathcal{S}(\Omega, I)$ の極小性を証明すればよい. 今, 仮に. $\mathcal{S}(\Omega, I)$ が極小でないと仮定する。つまり,

$$\exists \mathcal{S}', \; \mathcal{S}' \leq \mathcal{S}, \; \mathcal{S}' \neq \mathcal{S}, \; * \mathcal{S}' \geq [\Omega, I]$$

とする. すると,

$$\mathcal{S}' = \{[\Omega'_1, I'_1], \cdots, [\Omega'_m, I'_m]\}$$

$$\mathcal{S} = \{[\Omega_1, I_1], \cdots, [\Omega_n, I_n]\}$$

に対して.

$$\forall_i \quad {}^\exists j \quad \Omega_i' \subset \Omega_j$$

が成立する。このような Ω_j の1つを Ω_i' に対応させて $\Omega_{(i)}$ で表わす。
$n \geq m$ と仮定すると,

$$\forall R \in [\Omega, \Gamma]$$

$$R = *_{[\Omega_i', \Gamma_i'] \in \mathcal{S}'} R[\Omega_i']$$

$$= *_{[\Omega_{(i)}, \Gamma_{(i)}] \in \mathcal{S}} R[\Omega_{(i)}]$$

となり．一方．

$$R = *_{[\Omega_j, \Gamma_j] \in \mathcal{S}} R[\Omega_j]$$

となる．これより $\mathcal{S}(\Omega, \Gamma) - \{[\Omega_{(i)}, \Gamma_{(i)}] \mid [\Omega_i', \Gamma_i'] \in \mathcal{S}'(\Omega, \Gamma)\}$ は冗長である．よって $n = m$ である。このとき

$$\exists j \quad \Omega_j' \subsetneq \Omega_{(j)}$$

なる j が存在すると．

$$\forall R \in [\Omega, \Gamma]$$

$$R = *_{[\Omega_i', \Gamma_i'] \in \mathcal{S}'} R[\Omega_i']$$

$$= \left(*_{[\Omega_{(i)}, \Gamma_{(i)}] \in \mathcal{S} - \{[\Omega_{(j)}, \Gamma_{(j)}]\}} R[\Omega_{(i)}] \right) * R[\Omega_j']$$

で $\Omega_j' \subsetneq \Omega_j$ となり，条件(3)に矛盾する。よって．(1)，(2)，(3)を満足する $\mathcal{S}$ に対して．

$$\mathcal{S}' \leq \mathcal{S}, \quad \mathcal{S}' \neq \mathcal{S}, \quad *\mathcal{S}' \supseteq [\Omega, \Gamma]$$

なる $\mathcal{S}'$ は存在しない．つまり

$$\mathcal{S} \in \mathcal{C}(\Omega, \Gamma)$$

となる．（証明終）.

以上の議論とは別に、もし

$$[\Omega_0, \Gamma_0] = *\mathcal{S}(\Omega_0, \Gamma_0)$$

が成立すれば、このスキーマは、スキーム$[\Omega_0, \Gamma_0]$と同じ情報格納能力を持つことになる。このことを格納完備性と名付ける。部分スキームの集合の族$\mathcal{D}(\Omega, \Gamma)$を、

$$\mathcal{S}(\Omega, \Gamma) \in \mathcal{D}(\Omega, \Gamma)$$

$$\text{iff} \quad [\Omega, \Gamma] = *\mathcal{S}(\Omega, \Gamma)$$

$$\text{and} \quad \neg \exists\, \mathcal{S}'(\Omega, \Gamma)$$

$$\text{s.t.} \quad 1)\ [\Omega, \Gamma] = *\mathcal{S}'(\Omega, \Gamma)$$

$$\text{and} \quad 2)\ \mathcal{S}'(\Omega, \Gamma) \leqslant \mathcal{S}(\Omega, \Gamma)$$

$$\text{and} \quad 3).\ \mathcal{S}'(\Omega, \Gamma) \neq \mathcal{S}(\Omega, \Gamma)$$

と定義する。

スキーマ設計を複雑にしている理由の1つは、

$$\exists\, [\Omega, \Gamma] \quad \text{s.t.} \quad \mathcal{C}(\Omega, \Gamma) \cap \mathcal{D}(\Omega, \Gamma) = \phi$$

となることである。

$\mathcal{S} \in \mathcal{C}(\Omega_0, \Gamma_0)$なる1つの$\mathcal{S}$を見つける立場を分解法といい、$\mathcal{S} \in \mathcal{D}(\Omega_0, \Gamma_0)$なる1つの$\mathcal{D}$を見つける立場を独立成分法という。

ここでは、分解法の立場に立ち、$\mathcal{C}(\Omega_0, \Gamma_0)$の要素$\mathcal{S}$で、$*\mathcal{S}$が最も$[\Omega_0, \Gamma_0]$に近いものを見つけることを目標にする。つまり$[\Omega_0, \Gamma_0]$を最も良く近似する$\mathcal{S}$を$\mathcal{C}(\Omega_0, \Gamma_0)$中に見つけることにする。

そこで、近似の良さを表わす半順序"$\sqsubseteq$"を導入する。$\mathcal{S}_1, \mathcal{S}_2 \in \mathcal{C}(\Omega, \Gamma)$とし、

$$\mathcal{S}_1 \sqsubseteq \mathcal{S}_2 \quad \text{iff} \quad *\mathcal{S}_1 \subseteq *\mathcal{S}_2$$

と定義する。最良近似解の集合$\mathcal{C}_0(\Omega_0, \Gamma_0)$は、

$$\mathcal{S}(\Omega,\Gamma) \in \mathcal{C}_0(\Omega,\Gamma)$$

$$\text{iff} \quad \mathcal{S}(\Omega,\Gamma) \in \mathcal{C}(\Omega,\Gamma)$$

and

$$\neg \exists\, \mathcal{S}'(\Omega,\Gamma) \in \mathcal{C}(\Omega,\Gamma)$$

$$\text{s.t.} \quad \mathcal{S}'(\Omega,\Gamma) \subsetneq \mathcal{S}(\Omega,\Gamma)$$

と定義できる。

部分スキームの集合 $\mathcal{S}(\Omega,\Gamma)$ に対して $\Gamma(\mathcal{S})$ を

$$[\Omega,\Gamma(\mathcal{S})] = *\mathcal{S}(\Omega,\Gamma)$$

と定義すると，

$$\Gamma(\mathcal{S}) = \left(\cup_{[\Omega',\Gamma']\in\mathcal{S}} \Gamma'\right) \cup \left\{*\left(\cup_{[\Omega',\Gamma']\in\mathcal{S}} \{\Omega'\}\right)\right\}$$

となる。

補題 2.1.

$$[\Omega,\Gamma] \subseteq [\Omega,\Gamma'] \quad \text{iff} \quad \Gamma \vdash \Gamma'$$

（証明）　自明.

定理 2.3

$$\forall \mathcal{S}_1(\Omega,\Gamma) \in \mathcal{C}(\Omega,\Gamma),\ \forall \mathcal{S}_2(\Omega,\Gamma) \in \mathcal{C}(\Omega,\Gamma)$$

$$\mathcal{S}_1(\Omega,\Gamma) \sqsubseteq \mathcal{S}_2(\Omega,\Gamma)$$

$$\text{iff} \quad \Gamma(\mathcal{S}_1) \vdash \Gamma(\mathcal{S}_2)$$

（証明）. 補題 2.1 と $\Gamma(\mathcal{S})$ の定義より明らか.

3、従属関係間の半順序.

3.1. 従属関係の集合の正準表現.

従属関係の集合Γとして、以下では FDと大域的MVDのみからなる集合を考える。

$\{X, Y_0, Y_1, \cdots, Y_n\}$がΩの分割であるとき、

$$X : [Y_0]\ Y_1 \mid Y_2 \mid \cdots \mid Y_n$$

で 集合

$$\{\ {}^{\forall}A \in Y_0\ X \to A,\ X \twoheadrightarrow Y_1,\ X \twoheadrightarrow Y_2, \cdots, X \twoheadrightarrow Y_n \}.$$

を表わす。集合 $\tilde{\Gamma}^*$ を

$$X : [Y_0]\, Y_1 \mid Y_2 \mid \cdots \mid Y_n \in \tilde{\Gamma}^*$$

$$\begin{aligned} \text{iff} \quad & \Gamma \vdash X : [Y_0]\, Y_1 \mid Y_2 \mid \cdots \mid Y_n \\ \text{and} \quad & \neg\, {}^{\exists} U : [V_0]\, V_1 \mid V_2 \mid \cdots \mid V_m \\ & \quad \text{s.t.} \\ & \qquad \Gamma \vdash U : [V_0]\, V_1 \mid V_2 \mid \cdots \mid V_m \vdash X : [Y_0]\, Y_1 \mid Y_2 \mid \cdots \mid Y_n . \end{aligned}$$

と定義する。

このような $\tilde{\Gamma}^*$ を求めるには、以下の推論規則

r. $X : [Y_0]\, Y_1 \mid Y_2 \mid \cdots \mid Y_n ,\ U : [V_0]\, V_1 \mid V_2 \mid \cdots V_m$

$$\vdash W : [Z_0]\, Z_1 \mid Z_2 \mid \cdots \mid Z_m \mid Z_{m+1} .$$

$$\begin{aligned} W &= X\,(Y_i \cap U) \\ Z_0 &= Y_0\,(Y_i \cap V_0) - W \\ Z_j &= Y_i \cap V_j \\ Z_{m+1} &= \Omega - W - \textstyle\bigcup_{1 \le j \le m} Z_j . \end{aligned}$$

と、次の性質

p. $\Gamma \vdash \overline{U} : [V_0]\, V_1 \mid V_2 \mid \cdots \mid V_m \vdash X : [Y_0]\, Y_1 \mid Y_2 \mid \cdots \mid Y_n$

$$
\begin{aligned}
\text{iff} \quad & \overline{U} \subseteq X \\
\text{and} \quad & V_0 \supseteq Y_0 \\
\text{and} \quad & \forall i \quad \exists I \subset \{1, 2, \cdots, m\}. \\
& \quad X Y_i = \overline{U}\left(\cup_{j \in I} V_j \right)
\end{aligned}
$$

を用いればよい。

example 3.1.

$\Omega_0 = \{A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q\}$

$\Gamma_0 = \{ G \to DK,\ AC \to OP,\ H \to AB,$

$AB \twoheadrightarrow CDEFGKLM,\ C \twoheadrightarrow DEFGKN,\ D \twoheadrightarrow AHKLN,\ F \twoheadrightarrow ABG,$

$I \twoheadrightarrow JQ \}$

のとき、$\tilde{\Gamma}_0$ を

$\tilde{\Gamma}_0$:

$G : [DK]\ ABCEFHIJLMNOPQ$

$AC : [OP]\ DEFGKN \mid BHIJLMQ$

$H : [AB]\ CDEFGIJKLMNOPQ$

$AB : [\ ]\ CDEFGKLM \mid HIJNOPQ$

$C : [\ ]\ DEFGKN \mid ABHIJLMOPQ$

$D : [\ ]\ AHKLN \mid BCEFGIJMOPQ$

$F : [\ ]\ ABG \mid CDEHIJKLMNOPQ$

$I : [\ ]\ JQ \mid ABCDEFGHKLMNOP$

と定義し、規則 r を使って、新しい従属関係を見つけ、性質 P を用いて不用なものを消去すると次のように $\tilde{\Gamma}_0^*$ が計算される。

$\tilde{\Gamma}_0^*$: G:[ABDKOP] H|L|N|IJQ|CEFM

H:[ABOP] N|IJQ|CDEFGKLM

AB:[OP] H|N|IJQ|CDEFGKLM

C:[ABOP] H|N|IJQ|DEFGK|L|M

D:[ABKOP] H|L|N|IJQ|CEFGM

F:[ABDKOP] G|H|L|N|IJQ|CEM

I: JQ|ABCDEFGHKLMNOP

$\tilde{\Gamma}^*$の定義より、次の定理が成立する。

定理 3.1.

$$\forall g \ \exists \mathfrak{f} \in \tilde{\Gamma}^* \quad (\Gamma \vdash g) \supset (\mathfrak{f} \vdash g)$$

$\tilde{\Gamma}^*$をΓの正準表現と名付ける。

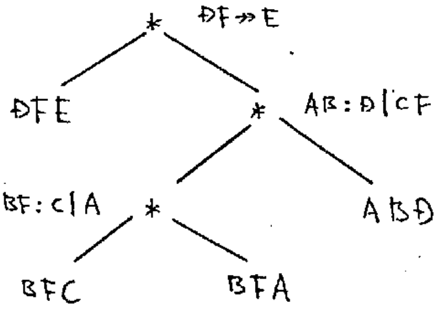
3.2. 分解過程.

FDとMVDを用いての関係の分解過程は、分解木と呼ばれる木構造に表現できる。

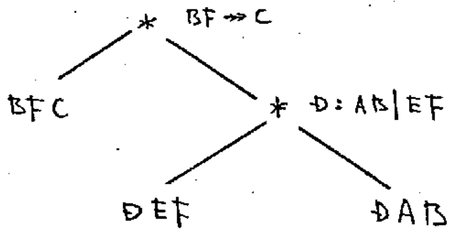
example 3.2.

Ω_0 : {A, B, C, D, E, F}

Γ : DF ↠ E , AB ↠ CF , BF ↠ C , D ↠ EF



分解木 (a)



分解木 (b)

従属関係 $f : X \Rightarrow Y$ $(X \rightarrow Y \text{ or } X \twoheadrightarrow Y)$ に対し atom(f, Σ) を,

$$\text{atom}(f, \Sigma) = (\neg\, {}^{\exists} g : U \twoheadrightarrow V \text{ in } XY \text{ s.t. } \Sigma \vdash g$$
$$\text{and. } U \supset Y \quad (g \text{ は nontrivial}))$$
$$\text{and } \text{cont}(f) = \Omega$$

と定義する.

$$U \supset X \text{ or } \Gamma \vdash U \rightarrow V$$

非形式的な解釈をすると, atom(f, Σ) が成立するような f を優先的に分解過程に使用すればよいと考えられる。

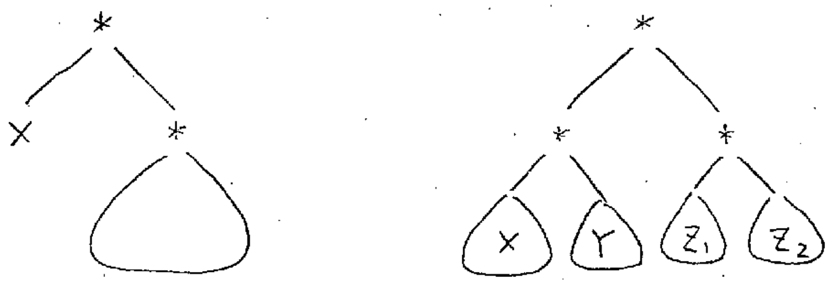
定理 3.2.

従属関係として, FD と MVD のみを考える。

$${}^{\forall}\Sigma \quad {}^{\exists} f \quad \Sigma \vdash f, \text{atom}(f, \Sigma)$$

(証明).

$\Sigma \vdash f$ なる f に対して $\neg$ atom(f, Σ) とする. $[\Omega; \Sigma]$ に対する分解木には, 図 3.1 (a) のような形をしたものは存在せず, 必らず 図 3.1 (b) のような形になる.



(a)

(b)

図 3.1.

$Z = Z_1 Z_2$ とし, XYZ が 図 3.2 のような関係にあるとする.

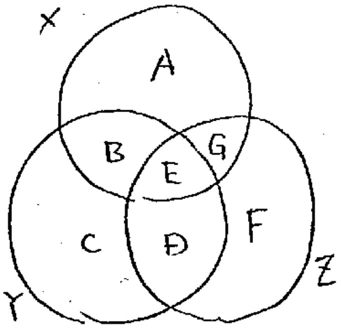


図 3.2.

図3.1 (b) より、$f_1 : (XY) \cap Z \twoheadrightarrow XY|Z$, $f_2 : X \cap Y \twoheadrightarrow X|Y$ となるが、これより、

$$f_1 : DEG \twoheadrightarrow ABC|F$$

$$f_2 : BE \twoheadrightarrow AG|CD \text{ in } ABCDEG$$

となる。

$$f_1, f_2 \ \vdash \ BEG \twoheadrightarrow A|CDF,\ BED \twoheadrightarrow C|AG$$

より、AかCのいずれかが空でないとき、このスキームは、図3.3のように分解できることがわかる。

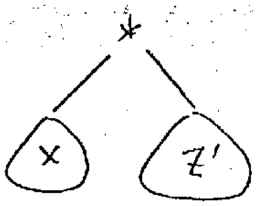


図 3.3

$A = C = \phi$ のときは、

$$f_1 : DEG \twoheadrightarrow B|F$$

$$f_2 : BE \twoheadrightarrow G|D \text{ in } BEGD$$

となり、atom(f_1, I) となり矛盾する。$A \neq \phi$ か $C \neq \phi$ のときも、X が non$([X, I|x])$ ならば、これを新しく図3.1 (b) のXYと考えて、同じ議論をくり返すと、最終的には、atom(f, I) なる大域的な f が必ず見つかり、仮定に矛盾する。（証明終）

3. 3. 従属関係の間の半順序.

$[\Omega_0, \Gamma_0]$ を スキーマ設計を行うデータベースとする。Γ_0^* を

$$\Gamma_0^* = \{ f \mid f \in \Gamma_0', \neg \exists g \in \Gamma_0' \ \ g \vdash f \}$$

$$\Gamma_0' = \bigcup_{\tilde{f} \in \tilde{\Gamma}_0^*} \tilde{f}$$

と定義する。

従属関係の集合 σ と $X \subset \Omega_0$ に対して

$$\begin{aligned} \varepsilon(X, \sigma) = \ & \text{if } \exists g : X' \Rightarrow Y' \in \sigma \text{ s.t. } X' \subset X, \ Y' - X \neq \phi \\ & \quad \text{then } \varepsilon(XY', \sigma) \\ & \quad \text{else } X \end{aligned}$$

と定義する。また、後に定義する σ_i に対して

$$\omega_i(\tilde{f}) = \varepsilon(X, \sigma_i)$$

と定義する。ただし $\tilde{f}$ は $X : [Y_0] Y_1 | \ldots | Y_n$ とする。$\tilde{f}, \tilde{g} \in \tilde{\Gamma}_0^*$ に対して

$$\tilde{f} \underset{i}{\leq} \tilde{g} \quad \text{iff} \quad \omega_i(\tilde{f}) \subseteq \omega_i(\tilde{g})$$

と定義する。

Γ_i^*, σ_i, $\tilde{\sigma}_i$, γ_i を

Γ_0^* : 上述の通り.

$$\sigma_0 = \{ f \mid f \in \Gamma_0^*, \ \text{atom}(f, \Gamma_0^*) \}$$

$$\tilde{\sigma}_0 = \{ \tilde{f} \cap \sigma_0 \mid \tilde{f} \in \tilde{\Gamma}_0^* \}$$

$$\gamma_0 = \{ \tilde{f} \cap \sigma_0 \mid \omega_0(\tilde{f}) \neq \phi, \neg \exists \tilde{g} \ \ (\omega_0(\tilde{g}) \neq \phi) \wedge (\tilde{g} \underset{0}{\leq} \tilde{f}) \}$$

$$\widetilde{\Gamma}_i^* = \{ \mathfrak{F}|_{\Omega'} \mid \mathfrak{F} \in \widetilde{\Gamma}_0^* \quad \mathfrak{F} = X:[Y_0]Y_1|\cdots|Y_n,$$
$$\Omega' = (\Omega - \bigcup_{0\le j\le i-1} \omega_j(\mathfrak{F}))\cup X \}$$

$$\Gamma_i^* = \{ f \mid f\in\Gamma_i', \neg\exists g\in\Gamma_i', g\vdash f \}$$

$$\Gamma_i' = \bigcup_{\mathfrak{F}\in\widetilde{\Gamma}_0^*} \mathfrak{F}$$

$$\sigma_i = \{ f \mid f\in\Gamma_i^*,\ \mathit{atom}(f,\Gamma_i^*) \}$$

$$\widetilde{\sigma}_i = \{ \mathfrak{F}\wedge\sigma_i \mid \mathfrak{F}\in\widetilde{\Gamma}_0^* \}$$

$$\gamma_i = \{ \mathfrak{F}\wedge\sigma_i \mid \omega_i(\mathfrak{F})\neq\phi, \neg\exists\widetilde{g}\,(\omega_i(\widetilde{g})\neq\phi)\wedge(\widetilde{g}\underset{i}{\leq}\mathfrak{F}) \}$$

と定義する。

定理 3.3.

$$\forall i \quad \sigma_i\neq\phi \quad , \quad \forall i \quad \gamma_i\neq\phi$$

(証明)

定理 3.2 より自明.

定理 3.4.

$$\exists n \quad \Gamma_{n+1}^* = \phi$$

(証明).

$$\forall i, \quad \gamma_i \notin \widetilde{\Gamma}_{i+1}^*$$

で,かつ

$$\forall i \quad \gamma_{i+1}\neq\phi$$

が成り立ち, さらに,

$$\forall i \quad \Gamma_0^* \vdash \Gamma_{i+1}^*$$

が成り立ち、$\{f \mid \Gamma_0^* \vdash f\}$なる集合は有限であるから、あるnが存在して

$$\Gamma_{n+1}^* = \phi$$

となる。（証明終）

定理 3.5.

$$\gamma_0 \cup \gamma_1 \cup \cdots \cup \gamma_n \vdash \Gamma$$

この定理を証明するために次の補題を用いる。

補題 3.1.

$$\forall i \quad \tilde{\Gamma}_i^* \cup \gamma_0 \cup \gamma_1 \cup \cdots \cup \gamma_{i-1} \vdash \Gamma$$

（証明）
数学的帰納法により証明する。

$i=0$のとき、

$$\tilde{\Gamma}_0^* \vdash \Gamma$$

となり明らか。γが$(i-1)$個以下のとき、この式が成立するとして、$\Gamma = \Gamma_1^*$とすると、

$$\tilde{\Gamma}_i^* \cup \gamma_1 \cup \gamma_2 \cup \cdots \cup \gamma_{i-1} \vdash \tilde{\Gamma}_1^*$$

が成り立つ。この式より、

$$\tilde{\Gamma}_i^* \cup \gamma_1 \cup \cdots \cup \gamma_{i-1} \cup \gamma_0 \vdash \tilde{\Gamma}_1^* \cup \gamma_0$$

となり、

$$\tilde{\Gamma}_1^* \cup \gamma_0 \vdash \tilde{\Gamma}_0$$

は、γは1個のときの式であるから、仮定より成立する。（証明終）

(定理 3.5の証明)

補題 3.1 において $i=n+1$ とおくと $\tilde{\Gamma}^*_{n+1}=\phi$ より定理が証明される.

(証明終).

example 3.2.

example 3.1 の $[\Omega_0, \Gamma_0]$ に対して $r_0, r_1, \cdots$ を求める.

$\tilde{\sigma}_0$: G : [ABDKOP] H|L|N

H : [ABOP] N

AB : [OP] N

C : [ABOP] H|N|L|M

D : [ABKOP] H|L|N

F : [ABDKOP] G|H|L|N

I : JQ

$r_0 = \{\{AB \rightarrow O, AB \rightarrow P, AB \twoheadrightarrow N\}, \{I \twoheadrightarrow JQ\}\}$.

$\tilde{\sigma}_1$: G : [ABDKOP] H|L|N|IJQ

H : [ABOP] N|IJQ

AB : IJQ in Ω-OPN

C : [ABOP] H|N|L|M|IJQ

D : [ABKOP] H|L|N|IJQ

F : [ABDKOP] G|H|L|N|IJQ

$r_1 = \{\{AB \twoheadrightarrow IJQ \text{ in } \Omega\text{-}OPN\}\}$.

$\tilde{\sigma}_2$: $\tilde{\sigma}_1 - \{$AB : IJQ in Ω-OPN$\}$

$r_2 = \{\{H \rightarrow A, H \rightarrow B, H \rightarrow O, H \rightarrow P, H \twoheadrightarrow N, H \twoheadrightarrow IJQ\}\}$

$\tilde{\sigma}_3$: $\tilde{\sigma}_2 - \{$H : [ABOP] N|IJQ$\}$

$r_3 = \{\{D \rightarrow A, D \rightarrow B, D \rightarrow K, D \rightarrow O, D \rightarrow P, D \twoheadrightarrow H, D \twoheadrightarrow L, D \twoheadrightarrow N, D \twoheadrightarrow IJQ\},$
$\{C \rightarrow A, C \rightarrow B, C \rightarrow O, C \rightarrow P, C \twoheadrightarrow H, C \twoheadrightarrow N, C \twoheadrightarrow L, C \twoheadrightarrow M, C \twoheadrightarrow IJQ\}\}$.

$\tilde{\sigma}_4$: $G : [ABDKOP]\ H|L|N|IJQ$

$\quad F : [ABDKOP]\ G|H|L|N|IJQ|CEM$

$\gamma_4 = \{\{G \to A, G \to B, G \to D, G \to K, G \to O, G \to P, G \twoheadrightarrow H, G \twoheadrightarrow L, G \twoheadrightarrow N, G \twoheadrightarrow IJQ\}\}$

$\tilde{\sigma}_5$: $F : [ABDKOP]\ G|H|L|N|IJQ|CEM$

$\gamma_5 = \{\{F \to A, F \to B, F \to D, F \to K, F \to O, F \to P, F \twoheadrightarrow G, F \twoheadrightarrow H, F \twoheadrightarrow L, F \twoheadrightarrow N$
$\quad F \twoheadrightarrow IJQ, F \twoheadrightarrow CEM\}\}$

$\tilde{\Gamma}_6^* = \phi$

δ_i を次のように定義する。

$$\delta_0 = \gamma_0$$

$$\delta_i = \{\mathfrak{F}_i / \gamma_{i-1}^* \mid \mathfrak{F}_i \in \gamma_i\}$$

$$\gamma_i{}^* = \cup_{0 \le j \le i-1} \gamma_j$$

$$\mathfrak{F}/\gamma_i{}^* = \{f \mid \mathfrak{F} - (\{f\} \cup \gamma_i{}^*)^+ H f\}$$

ただし $\Gamma^+ = \{f \mid \Gamma \vdash f\}$ とする。

example 3.3.

example 3.2 の結果より δ_i を求めると

$$\delta_0 = \{\{AB \to O, AB \to P, AB \twoheadrightarrow N\}, \{I \twoheadrightarrow JQ\}\}$$

$$\delta_1 = \{\{AB \twoheadrightarrow IJQ \text{ in } \Omega - OPN\}\}$$

$$\delta_2 = \{\{H \to A, H \to B\}\}$$

$$\delta_3 = \{\{D \to H, D \to K, D \twoheadrightarrow L\}, \{C \twoheadrightarrow H, C \twoheadrightarrow L, C \twoheadrightarrow M\}\}$$

$$\delta_4 = \{\{G \to D\}\}$$

$$\delta_5 = \{\{F \twoheadrightarrow G, F \twoheadrightarrow CEM\}\}$$

となる。

定理. 3.6.

$$\forall \mathfrak{F}_0 \in \gamma_0 , \quad \Gamma_0^* - \mathfrak{F}_0^\dagger \nvdash \mathfrak{F}_0$$

(証明).

もし,

$$\Gamma_0^* - \mathfrak{F}_0^\dagger \vdash \mathfrak{F}_0$$

とすると,

$$\forall f \in \mathfrak{F}_0 \ \exists h, g \in \Gamma^* - \mathfrak{F}_0 \quad h, g \vdash f$$

が成立する. $f: X \to Y$, $h: V \to W$, $g: U \to Z$ とすると.

$$(h, g)^\dagger \vdash \quad V(U \wedge W) \twoheadrightarrow W \wedge Z,\ U(V \wedge Z) \twoheadrightarrow Z \wedge W$$

となる. いま.

$$X = V(U \wedge W)$$
$$Y = W \wedge Z$$

とおくと.

$$\omega_0(\mathfrak{h}) = \varepsilon(V, \sigma_0)$$
$$\omega_0(\mathfrak{f}) = \varepsilon(V(U \wedge W), \sigma_0)$$

より

$$\omega_0(\mathfrak{f}) \geq \omega_0(\mathfrak{h})$$

となる. これと $\mathfrak{f} \in \gamma_0$ より. $\omega_0(\mathfrak{f})$ の極小性から.

$$\omega_0(\mathfrak{f}) \geq \omega_0(\mathfrak{h}) \geq \omega_0(\mathfrak{f})$$

となり. $\omega_0(\mathfrak{f}) = \omega_0(\mathfrak{h})$ である. しかし. $V \subset X$ であるから,
$V = X$ でなければならず.

$$\mathfrak{f} = \mathfrak{h}$$

となり, $W \wedge Z = Y$ より $h \in \mathfrak{F}_0$ となり矛盾する. (証明終)

定理 3.7.

$$\forall i,\ \forall \tilde{f}_i \in \gamma_i \quad \Gamma_i^* - \tilde{f}_i^{\,+} \nvdash \tilde{f}_i$$

（証明）

定理3.6において Γ_0^* を Γ_i^* で置き換えればよい。

定理 3.8.

$\delta_0 \cup \delta_1 \ \cdots \cup \delta_n$ は Γ_0 の極小被覆である.

（証明）.

1). 被覆であることの証明.

$$\delta_0 \cup \delta_1 \cup \cdots \cup \delta_n$$

$$\vdash \gamma_0 \cup \delta_1 \cup \cdots \cup \delta_n$$

$$\vdash \gamma_0 \cup \{\tilde{f}_1/\gamma_0 \mid \tilde{f}_1 \in \gamma_1\} \cup \delta_2 \cup \cdots \cup \delta_n$$

$$\vdash \gamma_0 \cup \gamma_1 \cup \{\tilde{f}_2/\gamma_1^* \mid \tilde{f}_2 \in \gamma_2\} \cup \delta_3 \cdots \cup \delta_n$$

$$\vdash \gamma_0 \cup \gamma_1 \cup \gamma_3 \cup \delta_3 \cup \cdots \cup \delta_n$$

$$\vdots$$

$$\vdash \gamma_0 \cup \gamma_1 \cup \cdots \cup \gamma_n$$

$$\vdash \Gamma_0$$

2). 極小であることの証明.

極小性は γ_i, δ_i の構成法から明らか。　　　（証明終）

Δ を

$$\Delta = \cup_i \cup_{f \in \delta_i} f$$

と定義する. Γ_0 が FD と大域的な MVD のみからなる場合には. Δ の要素も FD と大域的な MVD のみからなるようにできる.

example 3.4.

example 3.3 の δ_i に対して,

Δ : $AB \to O$, $AB \to P$, $AB \twoheadrightarrow N$

$I \twoheadrightarrow JQ$,

$AB \twoheadrightarrow IJQ$,

$H \to A$, $H \to B$,

$D \to H$, $D \to K$, $D \twoheadrightarrow L$,

$C \twoheadrightarrow H$, $C \twoheadrightarrow L$, $C \twoheadrightarrow M$,

$G \to D$, $F \twoheadrightarrow G$, $F \twoheadrightarrow CEM$

となる。δ_1 中の $AB \twoheadrightarrow IJQ$ in Ω-OPN は 場の制約 "in Ω-OPN" を削除できる. このことは常に成立する性質である.

定理 3.9

$$\forall f \in \Delta \qquad \Delta - \{f\}^+ \nvdash f$$

(証明).

Δ が極小被覆であることから明らか.

定理 3.10

f を $X \Rightarrow Y$ とするとき Γ / f で $\Gamma|_{\Omega - Y}$ を意味することにする。

$$\forall f \in \Delta \qquad \Delta / f \vdash \Gamma_0 / f$$

(証明)

$$\Delta \vdash \Gamma_0$$

より $f : X \rightarrow Y$ とすると

$$\Delta|_{\Omega-Y} \vdash \Gamma_0|_{\Omega-Y}$$

が成り立ち.

$$\Delta/f \vdash \Gamma_0/f$$

となる.　(証明終)

定理 3.10

$$\neg\, {}^{\exists}f\ {}^{\exists}g \in \Delta \quad \Delta/f \cup \{f\} \vdash \Delta/g \cup \{g\}$$

(証明)

$$\Delta/f \cup \{f\} \vdash \Delta/g \cup \{g\}$$

とすると

$$ {}^{\exists}h \in \Delta\ {}^{\exists}f \in \Delta,\ {}^{\exists}g \in \Delta \quad h, f \vdash g$$

となる. これは矛盾する. (証明終)

4 最良近似非冗長4NFスキーマの設計

$\mathcal{F}\in\Gamma_0^*$ に対し，$\omega(\mathcal{F})$ を

$$\omega(\mathcal{F}) = \cup_i\ \omega_i(\mathcal{F})$$

と定義する．

$$\mathcal{R} = \cup_i \cup_{\mathcal{F}\in\gamma_i} \mathcal{F}$$

とし，$\mathcal{S}\in\mathcal{C}_0(\Omega_0,\Gamma_0)$ なるスキーマで，$\mathcal{R}$ 中の従属関係のみによる分解過程の結果得られるものの集合を $\mathcal{E}_R(\Omega_0,\Gamma_0)$ で表わす．

定理

$$\neg\ \exists\mathcal{S}\in\mathcal{C}_0(\Omega_0,\Gamma_0)\ \forall\mathcal{S}'\in\mathcal{E}_R \quad \mathcal{S}\fallingdotseq\mathcal{S}'$$

(証明)

$$\neg\ \exists\mathcal{S}\in\mathcal{C}_0(\Omega_0,\Gamma_0)\ \forall\mathcal{S}'\in\mathcal{E}_R \quad \Gamma(\mathcal{S})\vdash\Gamma(\mathcal{S}')$$

を証明すればよい。

$\Gamma\backslash g$ を

$$\Gamma\backslash g = \Gamma_{XY} \quad (g: X\Rightarrow Y)$$

と定義すると，$\mathcal{S}$ の分解過程では Γ_0^* が使われるとしてよいので，

$$\Gamma(\mathcal{S}) = \Gamma(\mathcal{S}(\Omega_0/g, \Gamma_0^*/g)) \cup \Gamma(\mathcal{S}(\Omega_0\backslash g, \Gamma_0^+\backslash g)) \cup g$$

となる．一方 $\mathcal{S}'$ の分解過程では最初に $\omega(\mathcal{F})$ が極小の $\mathcal{F}$ に対し，$\mathcal{F}^*=\mathcal{F}\cap(\cup_i\sigma_i)$ なる $\mathcal{F}^*$ を用いるとすると，

$$\Gamma(\mathcal{S}') = \Gamma(\mathcal{S}'(\Omega_0/\mathcal{F}^*, \mathcal{R}/\mathcal{F}^*)) \cup \Gamma(\mathcal{S}'(\Omega_0\backslash\mathcal{F}^*, \mathcal{R}\backslash\mathcal{F}^*)) \cup \mathcal{F}^*$$

である。 $\Gamma(\mathcal{S})\vdash\Gamma(\mathcal{S}')$ なら，特に $\Gamma(\mathcal{S})\vdash\mathcal{F}^*$ であり，したがって，

$\exists\, \tilde{\mathfrak{X}} \in \tilde{\Gamma}_0^*/g$ (or $\tilde{\mathfrak{X}} \in \tilde{\Gamma}_0^* \backslash g$) $\tilde{\mathfrak{X}}, g \vdash \tilde{\mathfrak{F}}^*$

となる。定理3.6と同様にして $\tilde{\mathfrak{X}} = \mathfrak{F}^*$ が得られ、$\tilde{\Gamma}_0^*/g$ に対応する分解に $\mathfrak{F}^*$ が使われねばならない。この場合には、

$$\begin{aligned}\Gamma(\delta) = \mathfrak{F}^* &\cup \Gamma(\delta(\Omega_0/g/\mathfrak{F}^*, \Gamma_0^*/g/\mathfrak{F}^*))\\ &\cup \Gamma(\delta(\Omega_0/g\backslash\mathfrak{F}^*, \Gamma_0^*/g\backslash\mathfrak{F}^*))\\ &\cup \Gamma(\delta(\Omega_0\backslash g, \Gamma_0^*\backslash g))\\ &\cup \{g\}\end{aligned}$$

となり、δ'' が存在して

$$\begin{aligned}\Gamma(\delta) = \mathfrak{F}^* &\cup \Gamma(\delta''(\Omega_0/\mathfrak{F}^*, \Gamma_0^*/\mathfrak{F}^*))\\ &\cup \Gamma(\delta''(\Omega_0\backslash\mathfrak{F}^*, \Gamma_0^*\backslash\mathfrak{F}^*))\end{aligned}$$

となる。~~同様の議論をサブスキーマに適用すると定理が証明される~~

(証明終)

スキーマ設計のアルゴリズムは次のようになる。

1. それぞれの $\mathfrak{F} \in \tilde{\Gamma}_0^*$ に対し、$\omega(\mathfrak{F})$ と $\mathfrak{F}^*$ を求める。

$$\omega(\mathfrak{F}) = \cup_i\, \omega_i(\mathfrak{F})$$
$$\mathfrak{F}^* = \mathfrak{F} \cap (\cup_i\, \sigma_i)$$

2. $\omega(\mathfrak{F})$ に関して $\mathfrak{F}$ の Hasse 図を書く。
3. BCNF の D-tree スキーマ設計アルゴリズムで、各ノードのリレーションを求め、冗長属性を除去する。
4. $\tilde{\Gamma}_0^*$ を参照して、各ノードのリレーションを 4NF に分解する。

設計プロセスの例を次頁以後に示す。

example 4.1.

例として example 3.1 から 3.2 で示した例を用いる.

$\tilde{T}_0^*$: $\mathfrak{F}^1$: G [ABDKOP] H|L|N|IJQ|CEFM

$\mathfrak{F}^2$: H [ABOP] N|IJQ|CDEFGKLM

$\mathfrak{F}^3$; AB [OP] H|N|IJQ|CDEFGKLM

$\mathfrak{F}^4$: C [ABOP] H|N|IJQ|DEFGK|L|M

$\mathfrak{F}^5$: D [ABKOP] H|L|N|IJQ|CEFGM

$\mathfrak{F}^6$: F [ABDKOP] G|H|L|N|IJQ|CEM

$\mathfrak{F}^7$: I : JQ|ABCDEFGHKLMNOP

			ω	key
γ_0 :	$\mathfrak{F}^3$:	AB [OP] N	ABOPN	AB
	$\mathfrak{F}^7$:	I : JQ	IJQ	I
γ_1 :	$\mathfrak{F}^3$:	AB : IJQ	ABIJQ	AB
γ_2 :	$\mathfrak{F}^2$:	H [ABOP] N|IJQ	HABOPNIJQ	H
γ_3 :	$\mathfrak{F}^5$:	D [ABKOP] H|L|N|IJQ	DABKOPHLNIJQ,	D
	$\mathfrak{F}^4$:	C [ABOP] H|N|L|M|IJQ	CABOPHNLMIJQ,	C
γ_4 :	$\mathfrak{F}^1$:	G [ABDKOP] H|L|N|IJQ,	GABDKOPHLNIJQ,	G
γ_5 :	$\mathfrak{F}^6$:	F [ABDKOP] G|H|L|N|IJQ|CEM	FABDKOPGHLNIJQCEM	F

$\omega(\mathfrak{F}^1)$ = GABDKOPHLNIJQ

$\omega(\mathfrak{F}^2)$ = HABOPNIJQ

$\omega(\mathfrak{F}^3)$ = ABOPNIJQ

$\omega(\mathfrak{F}^4)$ = CABOPHNIJQLM

$\omega(\mathfrak{F}^5)$ = DABKOPHLNIJQ

$\omega(\mathfrak{F}^6)$ = FABDKOPGHLNIJQCEM

$\omega(\mathfrak{F}^7)$ = IJQ

$\mathfrak{F}^{1*}$ = G : [ABDKOP] H|L|N|IJQ

$\mathfrak{F}^{2*}$ = H : [ABOP] N|IJQ

$\mathfrak{F}^{3*}$ = AB : [OP] N|IJQ

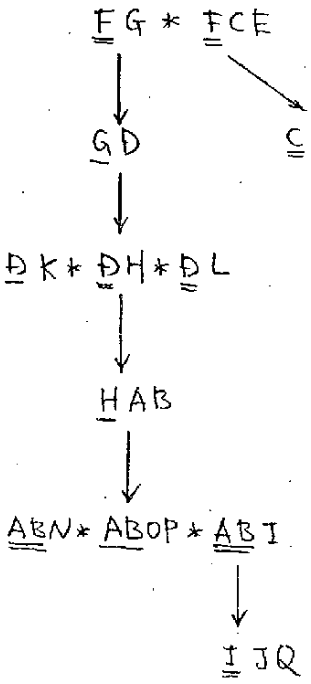
$\mathfrak{F}^{4*}$ - C : [ABOP] H|N|IJQ|L|M

$\mathfrak{F}^{5*}$ = D : [ABKOP] H|L|N|IJQ

$\mathfrak{F}^{6*}$ - F : [ABDKOP] G|H|L|N|IJQ|CEM

$\mathfrak{F}^{7*}$ = I : JQ

最適近似非冗長4NF D-tree スキーマ



参考のために、設計入力となった従属関係を再度示しておく。

$G \rightarrow DK$

$AC \rightarrow OP$

$H \rightarrow AB$

$AB \twoheadrightarrow CDEFGKLM$

$C \twoheadrightarrow DEFGKN$

$D \twoheadrightarrow AHKLN$

$F \twoheadrightarrow ABG$

$I \twoheadrightarrow JQ$

入力と出力は決して自明な関係にはない。

5. 結論.

この研究では，著者のD-treeスキーマに理論的根拠を与えると共にこのアルゴリズムを4NFへ拡張した。さらに，最良近似の概念を導入し，独立成分法の考えもとり入れた。本研究で示したアルゴリズムは，独立成分をみつけるのにも好都合である。

得られたアルゴリズムを用いたスキーマ設計の例は，このアルゴリズムが決して自明な働きをしているのではないことを示している。実際，ここに示した例に対し，4章で示したような都合のいい性質を持ったスキーマを見つけることは非常に困難である。著者等は，現在，このようなアルゴリズムの実際問題への応用を目指している。

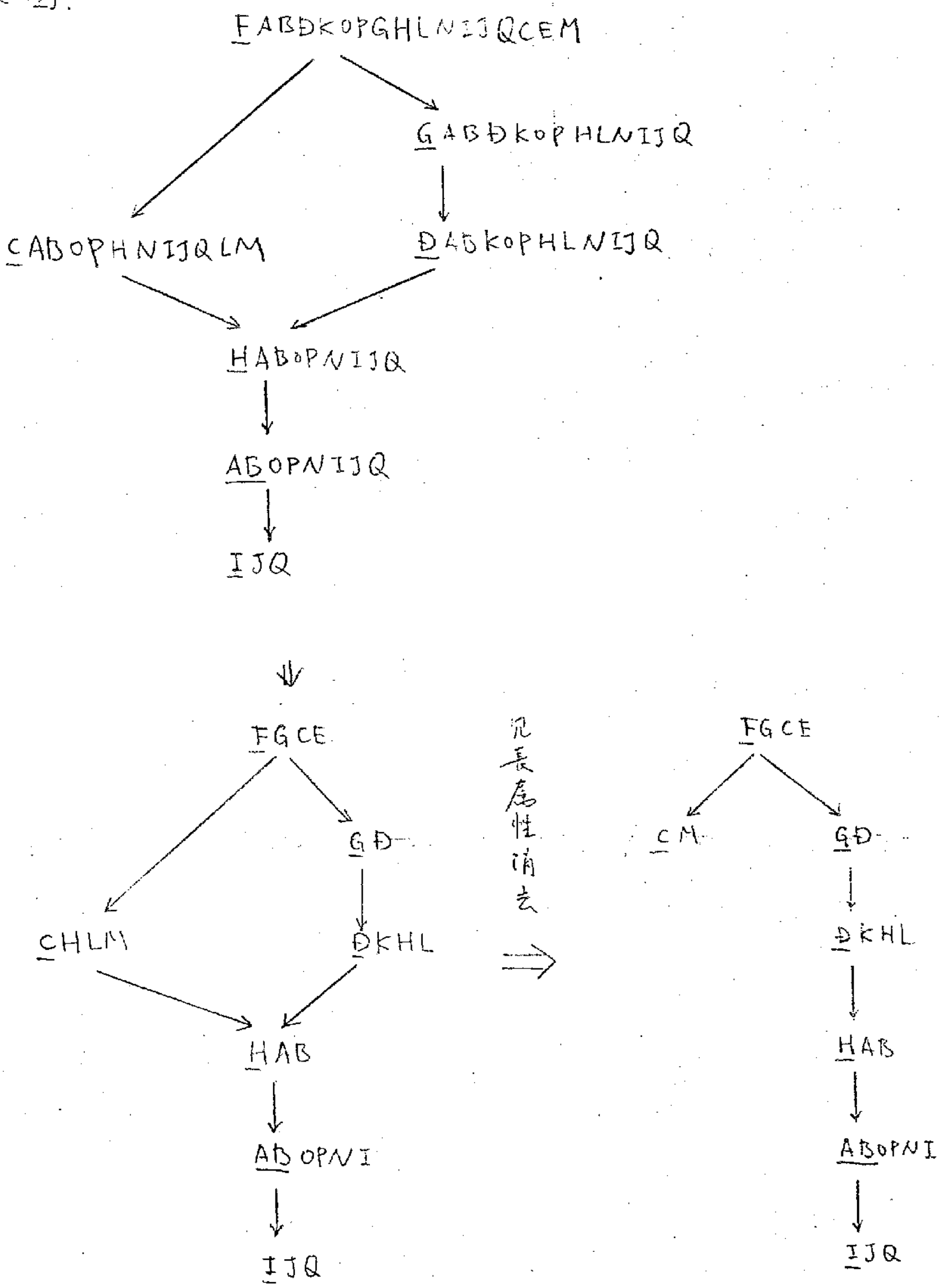
参考文献.

[FAGI77] R. Fagin, "Multivalued Dependencies and a New Normal Form for Relational Data Bases," ACM TODS, Vol. 2 No. 3 (Sept. 1977) pp. 262-278.

[TANA77] Y. Tanaka et al., "Decomposition and Composition of a Relational Database," Proc. 3rd VLDB, Tokyo, Oct. 1977, pp. 454-461.

[TANA79] Y. Tanaka, "Logical Design of a Relational Schema and Integrity of a Data Base," in "Database Architecture" North-Holland, Bracchi ed., New-York, 1979, pp. 297-317.

Hasse 図.


FABDKOPGHLNIJQCEM
GABDKOPHLNIJQ
CABOPHNIJQLM
DABKOPHLNIJQ
HABOPNIJQ
ABOPNIJQ
IJQ
FGCE
GD
CHLM
DKHL
HAB
ABOPNI
IJQ
冗長属性消去
FGCE
CM
GD
DKHL
HAB
ABOPNI
IJQ

Presented at JIPDEC Information
Systems Seminar on Semantic
Aspects of Databases, JIPDEC,
Tokyo, Japan, Feb. 21-22, 1980

Preservability of Data Dependencies for Relational Database Operations

Katsumi Tanaka*

Yahiko Kambayashi**

Shuzo Yajima**

* College of Liberal Arts, Kobe University
Nada, Kobe 657, Japan

** Dept. of Information Science, Kyoto University
Sakyo, Kyoto 606, Japan

1. Introduction

In relational model of data [CODD7006], data dependencies are useful tools to describe the semantics of data. Data dependencies are used as integrity constraints to specify a set of meaningful relations in the database. Various kinds of data dependencies have been introduced such as functional [CODD7105], multivalued [ZANI7606][FAGI7709], mutual [NICO7809], join [RISS7809], Boolean [SAGIF7903] and generalized mutual dependencies [MENDM7910].

In this paper, we mainly discuss the preservability of data dependencies (especially, functional, multivalued and generalized mutual dependencies) for the well-known relational database operation called natural joins.

A central problem of relational database design is how to select a relational database scheme consisting of relation schemes (sets of attributes) and data dependencies. It is often the case that we start the design from giving an initial relation scheme R (a set of all the attributes in the database) and a set D of data dependencies defined on the initial relation scheme. In order to reduce the redundancy of data and so called "storage anomalies" [DATE77], it is often useful to decompose R into a set of relation schemes $R_1,\ldots,R_n$. According to the decomposition, the set D of data dependencies defined on R is transformed into $\{D_1,\ldots,D_n\}$, where each D_i denotes a set of data dependencies defined on R_i, that are implied by D. The contents of relations (instances) may vary due to several storage operations, such as insertions, deletions and updates. In order to guarantee the given set D by $R_1,\ldots,R_n$, if each storage operation is allowed to be performed for any instance r_i of R_i, then it must transform r_i into r_i', where both r_i and r_i' satisfy D_i. Therefore, it is necessary to decompose R into $R_1,\ldots,R_n$ so that the given set D may hold in $r_1*r_2*\ldots*r_n$ (* denotes a natural join operator) for any instance r_i of R_i $(i=1,\ldots,n)$.

In this paper, the preservability of a data dependency for a natural join is defined as follows: Consider two relation schemes R_1 and R_2, where two sets D_1 and D_2 of data dependencies hold in R_1 and R_2, respectively. According to the notation in [BEERM7904], $SAT(D_i)$ (i=1,2) is assumed to denote a set of possible instances of R_i that satisfy D_i. A data dependency d in D_1 is said to be preserved in R_1*R_2 if and only if d holds in r_1*r_2 for any r_1,r_2 that belong to $SAT(D_1)$ and $SAT(D_2)$, respectively.

For example, let $R_1=\{A,B,C,D\}$ and $R_2=\{B,C,E\}$ be two relation schemes with $D_1=\{A\twoheadrightarrow B|CD\}$ and $D_2=\{BC\rightarrow E\}$. Here, $A\twoheadrightarrow B|CD$ is a multivalued dependency on R_1 [FAGI7709] and $BC\rightarrow E$ is a functional dependency on R_2. Fig.1 shows example instances r_1 and r_2 such that r_1 and r_2 belong to $SAT(D_1)$ and to $SAT(D_2)$, respectively. If we take a natural join for these example instances r_1 and r_2, then we can find that the multivalued dependency $A\twoheadrightarrow B|CD$ in D_1 does not hold in the instance r_1*r_2. That is, the data dependency $A\twoheadrightarrow B|CD$ in D_1 is not preserved in R_1*R_2. On the other hand, from the results of this paper, we can find that the functional dependency $BC\rightarrow E$ is preserved in R_1*R_2.

The preservability of data dependencies for natural joins is a useful criterion for designing a relational database scheme. The concept of our "preservability" is different from the various well-known criteria as follows:

(a) Lossless join [AHO-B7909],

(b) Independent components [RISS7712], and

(c) "Preservation of data dependencies" in [BEERM7904].

Fig. 2 (a), (b),(c) and (d) illustrate these three well-known criteria and our "preservability", respectively.

As shown in Fig.2 (a), the lossless join property guarantees only the fact that it is possible to reconstruct any instance I of an initial relation scheme

R without loss of information by joining the projections of I on R_i (i=1,...,n). That is,

$$I=I[R_1]*I[R_2]*...*I[R_n],$$

where $I[R_i]$ denotes a projection of I on R_i. Since the lossless join property is concerned with only the projections of instances of R, it does not always imply the fact that every dependency in D holds in $r_1*r_2*...*r_n$ for any instance r_i that belongs to $SAT(D_i)$.

As shown in Fig.2 (b), the concept of independent componenets by Rissanen implies the lossless join property and the fact that the given set D of funtional dependencies is equivalent to the union of $D_1,...,D_n$ (sets of functional dependencies implied by D). The preservability in this paper is implied by the concept of the independent components in special cases. In fact, as shown later, if $R_1,...,R_n$ are the independent components, then any functional dependency in D is preserved in $R_1*R_2*...*R_n$. When D includeds multivalued dependencies, however, Rissanen's definition of the independent components is insufficient [BEERB7809], [ZANIM78][TANAK7908].

Recently, Beeri et al. discussed the preservability of data dependencies under the universal relation (instance) assumption [BEERM7904]. The universal relation assumption enforces that each instance r_i of R_i in any database should be a projection of a same instance I of R on R_i. As shown in Fig.2 (c), for a given set $R_1,...,R_n$, they examined the problem whether the joins of these projections

$$I[R_1]*I[R_2]*...*I[R_n]$$

belongs to SAT(D) for any instance I in SAT(D).

We believe that the universal relation assumption is too strict since it leads to the introduction of more interrelation constraints: If an attribute A belongs to two relation schemes R_1 and R_2, then this assumption enforces that the two sets of A-values in r_1 and r_2 should be the same. Even if we have a tuple t to be inserted into r_1, we cannot insert it until we have some tuple of r_2 whose A-value is the same as the A-value of t. This may cause further insertion/deletion anomalies shown in [DATE77].

As shown in Fig.2 (d), in this paper, we do not have the universal relation assumption. We are concerned with the problem whether each data dependency in D of an initial relation sheme R is preserved in $r_1 * r_2 * \ldots * r_n$, where each r_i is any instance belonging to SAT(D_i).

2. Basic Concepts

A relational database scheme consists of a finite set of relation schemes, sets of data dependencies defined on each relation scheme and a set of inter-relation dependency constraints. Each relation scheme is a finite set of attributes. An initial relation scheme is a set of all the attributes appearing in the database. Uppercase letters from the beginning of the alphabet are used to denote names for attributes, and those from the end of the alphabet are names for sets of attributes. In this paper, concatenation of sets of attributes denotes union, such as R=XY.

An instance (or a relation) r of a relation scheme R with n attributes is a finite mathematical n-ary relation, which is a subset of the Cartesian product of n corresponding domains. Let R be an initial relation scheme and $R_1,\ldots,R_n$ be relation schemes such that $\bigcup_{i=1}^{n} R_i=R$. A database for the relation schemes $R_1,\ldots,R_n$ is a set of instances $r_1,\ldots,r_n$ such that each r_i is an instance of R_i.

For an instance r of a relation scheme R, the projection of r on a set X of attributes ($X \subseteq R$) is defined as $\{t[X];\ t \in r\}$ denoted by r[X]. Here, t[X] denotes a subtuple of a tuple t, which consists of values associated with each attribute in X. Let r ans s be instances of relation schemes R=XY and S=YZ such that X, Y and Z are mutual ly disjoint sets of attributes. The (natural) join of r and s, denoted by r*s, is a relation $\{(x,y,z);\ (x,y) \in r \text{ and } (y,z) \in s\}$. r[x,Y] denotes a set $\{t[Y];\ t[X]=x,\ t \in r \text{ and } X,Y \subseteq R\}$.

A functional dependency (FD) $X \rightarrow Y$ holds in a relation scheme R ($R \supseteq XY$) iff every two tuples of r that have the same X-value also have the same Y-value for any instance r of R.

A multivalued dependency (MVD) $X \to\to Y$ holds in R iff for every instance r of R, $r[x,YZ]=r[x,Y] \times r[x,Z]$ holds for any X-value x in r[X]. Here, X, Y and Z are subsets of R and Z=R-XY. When $X \to\to Y$ holds in R, we also denote the MVD by $X \to\to Y|Z$ (Z=R-XY) since $X \to\to Y$ implies the other MVD $X \to\to Z$ to complementarily hold in R. If Y or Z is an empty set or $X \supseteq Y$, then $X \to\to Y$ is called a trivial MVD; otherwise a nontrivial MVD.

Let r[XYZ] be a projection of r of R=XYZW, where $XYZ \cap W=\phi$, $W \neq \phi$ and $XY \cap Z=\phi$. An embedded multivalued dependency (EMVD) $X \to\to Y|Z$ holds on R=XYZW iff for every instance r of R, r[XYZ] is equal to the natural join of r[XY] and r[XZ]. If Y or Z is an empty set or $X \supseteq Y$, then the EMVD $X \to\to Y|Z$ is called a trivial EMVD; otherwise a nontrivial EMVD.

A mutual dependency (MD) m {X, Y, Z} holds in R ($R \supseteq XYZ$) iff the following condition is satisfied [MENDM7910]: Whenever any instance r of R contains three tuples t_1, t_2, t_3 (not necessarily distinct) such that $t_1[X]=t_2[X]$, $t_2[Y]=t_3[Y]$, and $t_3[Z]=t_1[Z]$, then threr must be some tuple t in r such that $t[X]=t_1[X]$, $t[Y]=t_2[Y]$ and $t[Z]=t_3[Z]$.

Let S be a set of S_{ij} $(1 \leq i<j \leq n)$ such that each S_{ij} is an arbitrary set of attributes. A generalized mutual dependency (GMD) m(S) of order n holds in R iff for every instance r of R, the following condition holds [MENDM7910]:
Whenever r contains n tuples $t_1,\ldots,t_n$ such that

$$t_i[S_{ij}]=t_j[S_{ij}] \text{ for all } 1 \leq i<j \leq n \qquad (*)$$

and there exists a tuple t (not necessarily in r) satisfying

$$t[S_{ij}]=t_i[S_{ij}] \text{ for all } 1 \leq i<j \leq n, \qquad (**)$$

then there must be some tuple in r that satisfies (**).

A multiboolean dependency (MBD) X → Y + Z holds in R iff for every instance r of R, every pair of tupless of r that agree in all attributes in X either agree in each of Y, or in each of Z [SAGIF7903].

3. Projections of Data Dependencies

When we decompose a given initial relation scheme R with a set D of data dependencies into a set of relation schemes $R_1, \ldots, R_n$, it is often necessary to compute the projections of D on each R_i.

The projectability of FDs and MVDs have been already well studied in [FAGI7709][ZANI7606][AHO-B7909][ULLM80]. Especially, Aho, Beeri and Ullman showed how to compute the projection of a set of FDs and MVDs as follows:

Theorem 1: [ULLM80]

Let D be a set of FDs and MVDs that hold in R. For any subset S of R, the set of FDs and MVDs that hold in S is computed as follows:

(1) Compute the closure D^+ of D, that is a set of all the FDs and MVDs implied by D.

(2) For each $X \rightarrow Y$ in D^+, if $X \subseteq S$, then $X \rightarrow Y \cap S$ holds in S.

(3) For each $X \rightarrow\rightarrow Y$ in D^+, if $X \subseteq S$, then $X \rightarrow\rightarrow Y \cap S$ holds in S.

(4) No other FDs or MVDs for S may be deduced from the fact that D holds in R.

Several complete sets of inference rules for FDs and MVDs are known , which are useful to compute the closure D^+ of D in [ARMS7408][BEERF7708]. As for the case of mutual dependencies, however, any similar projection rule as Theorem 1 does not always hold as shown below.

Theorem 2: Assume that only a mutual dependency $m\{X,Y,Z\}$ holds in $R=XYZ$. For any proper subset S of R with $X \cap S \neq \phi$, $Y \cap S \neq \phi$, and $Z \cap S \neq \phi$, there exists an instance I of R such that I satisfies $m\{X,Y,Z\}$, but not $m\{X \wedge S, Y \wedge S, Z \wedge S\}$.

Proof: It is sufficient to construct such an instance I for any subset S. Let $X=X_1X_2$, $Y=Y_1Y_2$, $Z=Z_1Z_2$, $X\cap S=X_1$, $Y\cap S=Y_1$ and $Z\cap S=Z_1$.

(Case 1) When S does not include any one of X, Y and Z, the instance I_1 in Fig.3 (a) is an example in which $\dot{m}\{X_1X_2,Y_1Y_2,Z_1Z_2\}$ holds, but $\acute{m}\{X\cap S,Y\cap S,Z\cap S\}=\acute{m}\{X_1,Y_1,Z_1\}$ does not hold.

(Case 2) When only one of X, Y and Z, for example X, is included by S, the instance I_2 in Fig.3 (b) is an example in which $m\{X_1X_2,Y_1Y_2,Z_1Z_2\}$ holds, but $\acute{m}\{X_1X_2,Y_1,Z_1\}$ does not hold.

(Case 3) When only two of X,Y and Z, for example X and Y, are included by S, the instance I_1 in Fig.3 (a) is also an example in which $\dot{m}\{X_1X_2, Y_1Y_2,Z_1Z_2\}$ holds, but $\acute{m}\{X_1X_2,Y_1Y_2,Z_1\}$ does not hold.

For other cases, such instances are easily constructed in the similar manner.

Q.E.D.

Example 1: Let D be a set of the MVD $A\twoheadrightarrow D$ and the MD $\dot{m}\{A,B,CD\}$ that hold in R=ABCD. Since $m\{A,B,CD\}$ is a mutual dependency, any instance I of R can be decomposed without loss of information into three of its projections. That is,

$$I=I[AB]*I[ACD]*I[BCD]$$

holds for any instance I of R. Furthermore, since $A\twoheadrightarrow D|BC$ also holds in R, any instance I of R can be decomposed without loss of information into two of its projections. That is,

$$I=I[AD]*I[ABC]$$

holds for any instance I of R. Fig.4 (a) shows an example instance I of R. It should be noted that $m\{A,B,C\}$ does not hold in the instance I. That is,

$$I[ABC]\neq I[AB]*I[AC]*I[BC].$$

As shown in Fig.4 (b), if we decompose R=ABCD into {AD,ABC} by the MVD $A\twoheadrightarrow D$, any information of $m\{A,B,CD\}$ is not transferred to the relation scheme ABC.

On the other hand, if we first decompose R into $\{AB, ACD, BCD\}$ by $m\{A, B, CD\}$ as shown in Fig.4 (c), then the MVD $A \twoheadrightarrow D|BC$ is projected on ACD as $A \twoheadrightarrow C|D$, and further decomposition is possible. That is, D implies

$$I=I[AB]*I[AC]*I[AD]*I[BCD]$$

holds for any instance I of R.

4. Preservability of Data Dependencies

In [RISS7809], Rissanen introduced the concept of join dependencies. The concept of join dependencies is closely related to the concept of lossless join property in [AHO-B7909]. A join dependency provides a necessary and sufficient condition for a relation to be decomposable without loss of information into n projections of the relation. Recently, Mendelzon and Maier showed that any join dependency can be represented by a generalized mutual dependency (GMD) and some MVDs [MENDM7910]. In this section, we show various conditions for several types of data dependencies (FDs, MVDs, MDs and GMDs) to be preserved for natural join operations. We also discuss the roles of multiboolean dependencies (MBDs) by Sagiv and Fagin in testing the preservability of data dependencies.

Consider two relation schemes R_1 and R_2, where two sets D_1 and D_2 of data dependencies (FDs, MVDs and GMDs) hold in R_1 and R_2, respectively. For each type of data dependencies (FDs, MVDs and GMDs), we define its preservability as follows:

(1) <u>FD-preservation</u>: An FD $X \rightarrow Y$ in D_1 is preserved in $R_1{*}R_2$ if $X \rightarrow Y$ holds in $r_1{*}r_2$ such that r_1 and r_2 are arbitrary instances belonging to SAT(D_1) and to SAT(D_2), respectively.

(2) <u>MVD-preservation</u>: An MVD $X \rightarrow\rightarrow Y$ in D_1 is preserved in $R_1{*}R_2$ if $X \rightarrow\rightarrow Y | R_1 - XY$ holds in $r_1{*}r_2$ such that r_1 and r_2 are arbitrary instances belonging to SAT(D_1) and SAT(D2), respectively.

(3) <u>GMD-preservation</u>: A GMD m(S) in D_1 is preserved in $R_1{*}R_2$ if m(S) holds in $r_1{*}r_2$ such that r_1 and r_2 are arbitrary instances belonging to SAT(D_1) and SAT(D_2).

Here, SAT(D_i) (i=1,2) denotes a set of all the instances that satisfy D_i.

It should be noted that the definition of the MVD-preservation does not necessrily require the fact that $X \twoheadrightarrow Y$ holds in $r_1 * r_2$ for any instance $r_1 \varepsilon$ SAT(D_1) and $r_2 \varepsilon$ SAT(D_2). It only requires that $X \twoheadrightarrow Y | R_1 - XY$ holds in $r_1 * r_2$, which is an EMVD of $R_1 R_2$. If we extend the definition of the MVD-preservation into the one described above, the following property of EMVDs [TANAK7908] is useful to check the extended MVD-preservation: Let $X \twoheadrightarrow Y$ be an MVD defined on R_1 ($R_1 \supseteq XY$). Suppose that R_2 is a relation scheme such that $R_2 \supseteq R_1$. Then, the MVD $X \twoheadrightarrow Y$ holds in R_2 if and only if $R_1 - Y \twoheadrightarrow Y$ holds in R_2.

Example 2: Let R_1=ABCD and R_2=CDE with $D_1 = \{A \rightarrow B, A \twoheadrightarrow C\}$ and $D_2 = \phi$. Example instances $r_1 \varepsilon$ SAT(D_1) and $r_2 \varepsilon$ SAT(D_2) are shown in Fig.5 (a) and (b), respectively. The instance $r_1 * r_2$ is shown in Fig.5 (c). Obviously, in this example, the FD $A \rightarrow B$ holds in $r_1 * r_2$ and the (EMVD) $A \twoheadrightarrow C | BD$ does not hold in $r_1 * r_2$. Therefore, the $\underbrace{\text{MVD in}}_{A \twoheadrightarrow C}$ D_1 is not preserved in $R_1 * R_2$. Consider another relation scheme R_3=CF with $D_3 = \phi$. Fig.5 (d) shows an example instance r_3 in SAT(D_3) and Fig.5 (e) shows the instance $r_1 * r_3$. Note that the MVD $A \twoheadrightarrow C$ does not hold in $r_1 * r_3$, but that $A \twoheadrightarrow C | BD$ (EMVD) holds in $r_1 * r_3$. From the results shown later, we can show that $A \twoheadrightarrow C | BD$ holds in $r_1 * r_3$ for any $r_1 \varepsilon$ SAT(D_1) and $r_3 \varepsilon$ SAT(D_3). Therefore, the MVD $A \twoheadrightarrow C$ in D_1 is preserved in $R_1 * R_3$.

The problem of testing the preservability differs from the problem of testing implications of data dependencies in the following respects: Consider two relation schemes R_1 and R_2 with D_1 and D_2, respectively. Here, we assume that D_1 and D_2 are sets of FDs and MVDs. Even if a data dependency $X \twoheadrightarrow Y | Z$ ($R_1 R_2 \supseteq XYZ$) is implied by $D_1 \cup D_2 \cup \{R_1 \cap R_2 \twoheadrightarrow R_1 - R_2 | R_2 - R_1\}$, $X \twoheadrightarrow Y | Z$ does not always hold in some $r_1 * r_2$ such that $r_1 \varepsilon$ SAT(D_1) and $r_2 \varepsilon$ SAT(D_2). Because some data dependencies in D_1 or D_2 may be violated in r_1*r_2 for some r_1 in $SAT(D_1)$ and r_2 in $SAT(D_2)$.

The following theorem shows that any FD in D_1 of a relation scheme R_1 is always preserved in R_1*R_2.

Theorem 3: Let R_1 and R_2 be two relation schemes. D_1 and D_2 are sets of FDs, MVDs and GMDs that hold in R_1 and R_2, respectively. Then, any FD $X \rightarrow Y$ in D_1 or D_1^+ is preserved in R_1*R_2.

Proof: Assume that some FD $X \rightarrow Y$ does not hold in some r_1*r_2 such that $r_1 \varepsilon$ $SAT(D_1)$ and $r_2 \varepsilon$ $SAT(D_2)$. There must be two distinct tuples t_1 and t_2 in r_1*r_2 such that $t_1[X]=t_2[X]$ and $t_1[Y]\neq t_2[Y]$. Since r_1*r_2 contains such tuples t_1 and t_2, r_1 must contain a tuple t_1' and t_2' such that $t_1'[X]=t_2'[X]$ and $t_1'[Y]\neq t_2'[Y]$. This leads to the fact that $X \rightarrow Y$ does not hold in r_1. A contradiction. Q.E.D.

The following theorem provides a sufficient condition for an MVD to be preserved in R_1*R_2.

Theorem 4: [TANAK7901]

Let R_1 and R_2 be two relation schemes. D_1 and D_2 are sets of FDs, MVDs and GMDs that hold in R_1 and R_2, respectively. Any MVD $X \twoheadrightarrow Y$ in D_1 is preserved in R_1*R_2 if

(1) either the FD $X \rightarrow Y \cap R_2$ or $X \rightarrow Z \cap R_2$ holds in R_1, or

(2) $X \cap R_2 \twoheadrightarrow Y \cap R_2 | Z \cap R_2$ holds in R_2, where $Z=R_1-XY$.

Proof: Without loss of generality, we can assume that $R_1=X_1X_2Y_1Y_2Z_1Z_2$ and $R_2=WX_1Y_1Z_1$, where W,X_1,X_2,Y_1,Y_2,Z_1 and Z_2 are disjoint sets of attributes.

It is sufficient to show that $X_1X_2 \twoheadrightarrow Y_1Y_2|Z_1Z_2$ holds in $r_1{*}r_2$ for any instance $r_1 \varepsilon$ SAT(D_1) and $r_2 \varepsilon$ SAT(D_2). Let $X=X_1X_2$, $Y=Y_1Y_2$ and $Z=Z_1Z_2$.

(Case 1) Assume that the FD $X \rightarrow Y \cap R_2$, that is, $X_1X_2 \rightarrow Y_1$ holds in R_1. From Theorem 3, the FD $X_1X_2 \rightarrow Y_1$ in D_1 (or D_1^+) is always preservedin $R_1{*}R_2$. Since the FD $X_1X_2 \rightarrow Y_1$ in R_1R_2 implies the MVD $X_1X_2 \twoheadrightarrow Y_1|Y_2Z_1Z_2W$. From Theorem 1, we can find that the EMVD $X_1X_2 \twoheadrightarrow Y_1|Z_1$ always holds in any $r_1{*}r_2$. r_1 and r_2 are joined on only $X_1Y_1Z_1$ and $X_1X_2 \twoheadrightarrow Y_1Y_2|Z_1Z_2$ holds in any r_1. Therefore, in any $r_1{*}r_2$, $X_1X_2 \twoheadrightarrow Y_1Y_2|Z_1Z_2$ holds.

(Case 2) Assume that $X \cap R_2 \twoheadrightarrow Y \cap R_2|Z \cap R_2$, that is, $X_1 \twoheadrightarrow Y_1|Z_1$ holds in R_1. Let r be an instance $r_1{*}r_2$ such that r_1 and r_2 are arbitrary instances in SAT(D_1) and in SAT(D_2), respectively. Let t be an arbitrary tuple in r such that $t[X_1X_2]=x_1x_2$. Since r_1 contains a tuple whose X_1X_2-value is x_1x_2, and $X_1X_2 \twoheadrightarrow Y_1|Z_1$ holds in R_1 by Theorem 1,

$$r_1[x_1x_2,Y_1Z_1]=r_1[x_1x_2,Y_1] \times r_1[x_1x_2,Z_1].$$

From the assumption of that $X_1 \twoheadrightarrow Y_1|Z_1$ holds in R_2, we have

$$r_2[x_1,Y_1Z_1]=r_2[x_1,Y_1] \times r_2[x_1,Z_1].$$

In $r=r_1{*}r_2$, we have

$$\begin{aligned} r[x_1x_2,Y_1Z_1] &= r_1[x_1x_2,Y_1Z_1] \cap r_2[x_1,Y_1Z_1] \\ &= (r_1[x_1x_2,Y_1] \cap r_2[x_1,Y_1]) \times (r_1[x_1x_2,Z_1] \cap r_2[x_1,Z_1]) \\ &= r[x_1x_2,Y_1] \times r[x_1x_2,Z_1]. \end{aligned}$$

Therefore, $X_1X_2 \twoheadrightarrow Y_1|Z_1$ holds in any $r_1{*}r_2$. Since r_1 and r_2 are joined on only $X_1Y_1Z_1$, $X_1X_2 \twoheadrightarrow Y_1Y_2|Z_1Z_2$ holds in any $r_1{*}r_2$. Q.E.D.

The following theorem provides a sufficient condition for a GMD m(S) to be preserved in $R_1{*}R_2$.

Theorem 5: Let R_1 and R_2 be two relation schemes with D_1 and D_2 (sets of FDs, MVDs GMDs), respectively. A GMD $m(S)$ in D_1 is preserved in R_1*R_2 if a GMD $m(S')$ holds in R_2 such that $S'=\{S_{ij} \cap R_1;\ 1 \leq i < j \leq n$ and $S_{ij} \in S\}$.

Proof: For simplicity, let $m(S)$ be a mutual dependency $m\{X,Y,Z\}$. Let $R_1= VX_1X_2Y_1Y_2Z_1Z_2$, $R_2=WX_1Y_1Z_1$, $X=X_1X_2$, $Y=Y_1Y_2$ and $Z=Z_1Z_2$, where $V \cap XYZ=\phi$; $V \cap W=\phi$, $W \cap XYZ=\phi$, $X_1 \cap X_2=\phi$, $Y_1 \cap Y_2=\phi$ and $Z_1 \cap Z_2=\phi$. Suppose that $m(S)=m\{X,Y,Z\}$ is not preserved in R_1*R_2. That is, some $r=r_1*r_2$ contains three tuples t_1, t_2 and t_3 such that $t_1[X]=t_2[X]$, $t_2[Y]=t_3[Y]$, $t_3[Z]=t_1[Z]$, and it is possible to construct a tuple t such that $t[X]=t_1[X]$, $t[Y]=t_2[Y]$, $t[Z]=t_3[Z]$, but r does not contain t. Since $m\{X,Y,Z\}$ holds in R_1, r_1 must contain a tuple t' such that $t'[X]=t_1[X]$, $t'[Y]=t_2[Y]$, $t'[Z]=t_3[Z]$. From the assumption that $m\{X_1,Y_1,Z_1\}$ holds in R_2, r_2 contains a tuple t'' such that $t''[X_1]=t_1[X_1]$, $t''[Y_1]=t_2[Y_1]$, $t''[Z_1]=t_3[Z_1]$. r_1 and r_2 are joined on only $X_1Y_1Z_1$, t must be in r_1*r_2. A contradiction. It is possible to prove the general case in the similar manner.
Q.E.D.

Next, we discuss the roles of multiboolean dependencies (MBDs) for the MVD-preservation. Let us consider the case when $Y_2=Z_2=\phi$ in Theorem 4. Obviously, we have the following theorem concerned with MBDs:

Theorem 6: Let $R_1=X_1X_2Y_1Z_1$ and $R_2=WX_1Y_1Z_1$ be two relation schemes with D_1 and D_2 (sets of FDs, MVDs, GMDs), respectively. An MVD $X_1X_2 \twoheadrightarrow Y_1$ in D_1 is preserved in R_1*R_2 if $X_1X_2 \rightarrow Y_1 + Z_1$ (MBD) holds in R_1.

This theorem implies that there are cases such that $X \twoheadrightarrow Y$ in D_1 is preserved in R_1*R_2 even if none of $X \rightarrow Y \cap R_2$, $X \rightarrow Z \cap R_2$ and $X \cap R_2 \twoheadrightarrow Y \cap R_2 | Z \cap R_2$ holds.

Consider the following four statements:

(1) $X_1X_2 \rightarrow Y_1$ or $X_1X_2 \rightarrow Z_1$ holds.

(2) $X_1X_2 \rightarrow Y_1 + Z_1$ holds.

(3) $X_1 \rightarrow Y_1 + Z_1$ holds.

(4) $X_1 \rightarrow\rightarrow Y_1 | Z_1$ holds.

The statements (1) and (4) appeared in Theorem 4, and the statement (2) appeared in Theorem 6. It should be noted that (2) implies neither (1) nor (4), while (1) implies (2), (3) implies both (2) and (4). Fig.6 (a) shows example instances r_1 and r_2 which satisfy the condition stated in Theorem 6 and do not satisfy either (1) or (4). The problem here is whether or not some MBDs may be deduced from some set of FDs, MVDs and GMDs.

It should be also noted that $X_1X_2 \rightarrow Y_1 + Z_1$ does not imply $X_1X_2 \rightarrow Y_1Y_2 + Z_1Z_2$ $(Y_2 \neq \phi$ and $Z_2 \neq \phi)$ (see Fig.6(b)). However, when $Y_2 \neq \phi$ or $Z_2 \neq \phi$ in Theorem 4, it is easy to extend the result of Theorem 6 as follows:

<u>Theorem 7</u>: Let R_1 and R_2 be two relation schemes with D_1 and D_2 (sets of FDs, MVDs, GMDs), respectively. An MVD $X \rightarrow\rightarrow Y$ in D_1 is preserved in $R_1 * R_2$ if $X \rightarrow Y \cap R_2 + Z \cap R_2$ holds in R_1.

References

[AHO-B7909] Aho, A. V., Beeri, C. and Ullman, J. D., "The Theory of Joins in Relational Databases", ACM Trans. Database Systems, Vol.4, No.3, pp.297-314, Sept. 1979.

[ARMS7408] Armstrong, W. W., "Dependency Structures of Data Base Relationships", Proc. IFIP 74 Congress, pp.580-583, Aug. 1974.

[BEERF7708] Beeri, C., Fagin, R. and Howard, J. H., "A Complete Axiomatization for Functional and Muitlvalued Dependencies in Database Relations", Proc. ACM-SIGMOD 1979 International Conference on Management of Data, pp.47-61, Aug. 1977.

[BEERB7809] Beeri, C., Bernstein, P. A. and Goodman, N., "A Sophisticate's Introduction to Database Normalization Theory", Proc. 4th International Conference on VLDB, pp.113-124, Sept. 1978.

[BEERM7904] Beeri, C., Mendelzon, A. O., Sagiv, Y. and Ullman, J. D., "Equivalence of Relational Database Schemes", Proc. 11th Ann. ACM Symp. on Theory of Computing, pp.319-329, April 1979.

[BERN7612] Bernstein, P. A., "Synthesizing Third Normal Form Relations from Functional Dependencies", ACM Trans. Database Systems, Vol.1, No.4, pp.277-298, Dec. 1976.

[BISKD7905] Biskup, J., Dayal, U. and Bernstein, P. A., "Synthesizing Independent Database Schemas", Proc. ACM-SIGMOD International Conference on Management of Data, pp.143-151, May 1979.

[CODD7006] Codd, E. F., "A Relational Model of Data for Large Shared Data Banks", Comm. ACM, Vol.13, No.6, pp.377-387, June 1970.

[CODD7105] Codd, E. F., "Further Normalization of the Data Base Relational Model", Proc. Courant Computer Science Symposium 6, Data Base Systems, pp.34-64, May 1971.

[DATE77] Date, C. J., "An Introduction to Database Systems", 2nd ed., Addison-Wesley, 1977.

[FAGI7709] Fagin, R., "Multivalued Dependencies and a New Normal Form for Relational Databases", ACM Trans. Database Systems, Vol.2, No.3, pp.262-278, Sept. 1977.

[FAGI7905] Fagin, R., "A Normal,Form for Relational Databases That is Based on Domains and Keys", IBM Res. Rep., RJ2520, May 1979.

[KAMB7906] Kambayashi, Y., "A New Synthetic Approach for Relational Database Design", presented at AFIPS NCC, June 1979.

[KAMBT7911] Kambayashi, Y., Tanaka, K. and Yajima, S., "Semantic Aspects of Data Dependencies and Their Application to Relational Database Design", Proc. COMPSAC'79 , pp.398-403, Nov. 1979.

[MENDM7910] Mendelzon, A. O. and Maier, D., "Generalized Mutual Dependencies and the Decomposition of Database Relations", Proc. 5th International Conference on VLDB, pp.75-82, Oct. 1979.

[NICO7809] Nicolas, J. M., "Mutual Dependencies and Some Results on Undecomposable Relations", Proc. 4th International Conference on VLDB, pp.360-367, Sept. 1978.

[PARKD7910] Parker, D. S. and Delobel, C., "Algorithmic Applications for a New Result on Multivalued Dependencies", Proc. 5th International Conferen on VLDB, pp.67-74, Oct. 1979.

[RISS7712] Rissanen, J., "Independent Components of Relations", ACM Trans. Database Systems, Vol. 2, No.4, pp.317-325, Dec. 1977.

[RISS7809] Rissanen, J., "Theory of Relations for Databases - A Tutorial Survey", Proc. 7th Symp. on Math. Found. of Computer Science, Lecture Notes in Computer Science 64, pp.536-5-1, Sept. 1978.

[SAGIF7903] Sagiv, Y. and Fagin, R., "An Equivalence between Database Dependencies and a Subclass of Propositional Logic", IBM Res. Rep., RJ2500, March 1979.

[TANAK7901] Tanaka, K., Kambayashi, Y. and Yajima, S., "On the Representability of Decompositional Schema Design with Multivalued Dependencies", Kyoto Univ., Yajima Lab. Res. Rep., ER79-01, Jan. 1979.

[TANAK7908] Tanaka, K., Kambayashi, Y. and Yajima, S., "Properties of Embedded Multivalued Dependencies in Relational Databases", Trans. of Institute of Electronics and Communication Engineers of Japan, Section E, Vol.E62, No.8, pp.536-543, Aug. 1979.

[ULLM80] Ullman, J. D., "Principles of Database Systems", Computer Science Press, Inc., 1980.

[ZANI7606] Zaniolo, C., "Analysis and Design of Relational Schemata for Database Systems", Computer Methodology Group Report, Computer Science Dept., UCLA, UCLA-ENG-7669, June 1976.

[ZANIM78] Zaniolo, C. and Melkanoff, M. A., "Relational Schemas for Database Systems", Technical Report, UCLA-ENG-7801, UCLA, 1978.

Fig.1 Example instances and the preservability of functional and multivalued dependencies.

r_1

A	B	C	D
1	0	1	0
1	0	0	1
1	1	1	0
1	1	0	1

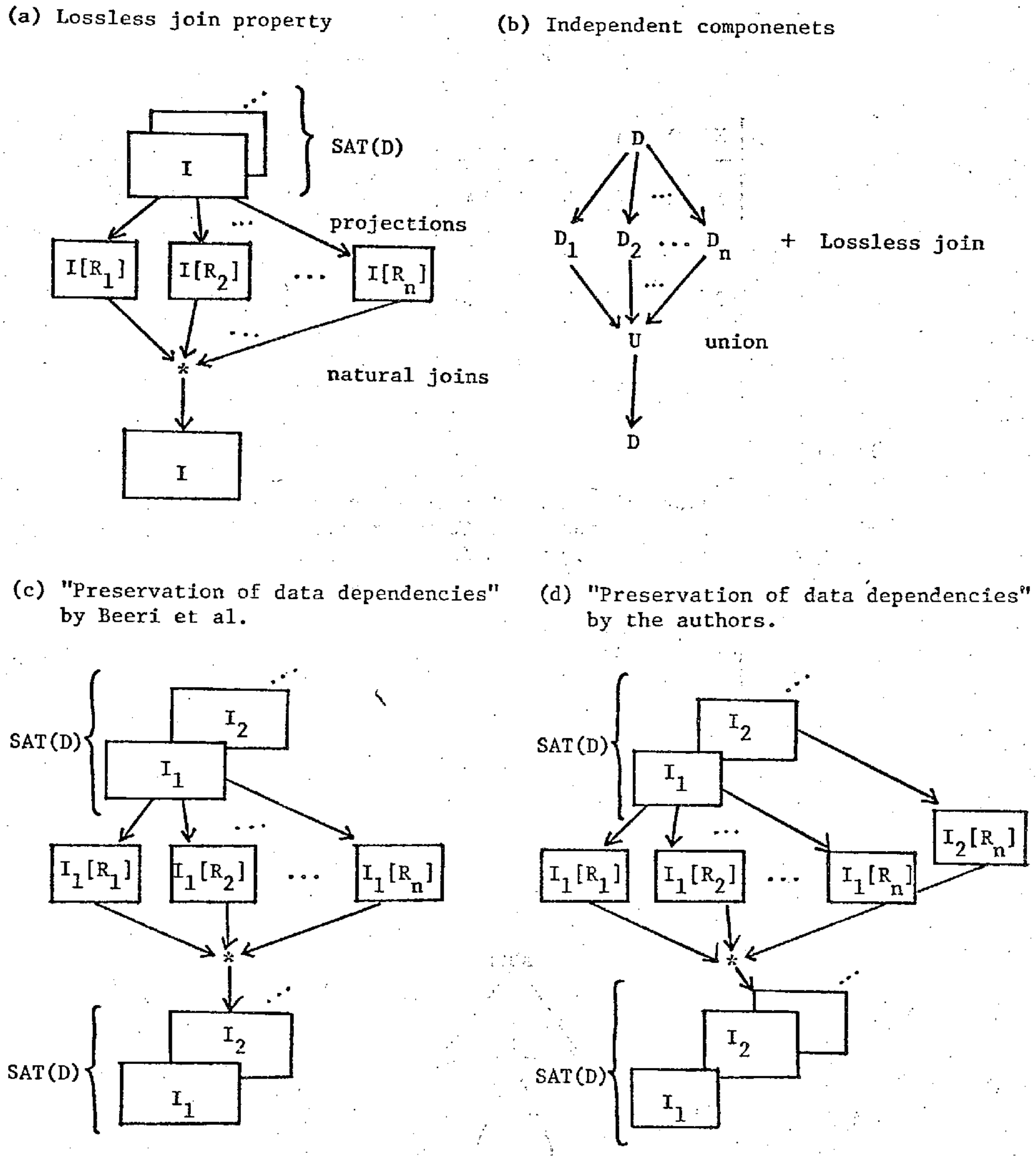
r_2

B	C	E
0	1	0
0	0	0
1	1	0

r_1*r_2

A	B	C	D	E
1	0	1	0	0
1	0	0	1	0
1	1	1	0	0

Fig.2 Various criteria for designing a relational database scheme.



(a) Lossless join property

(b) Independent componenets

(c) "Preservation of data dependencies" by Beeri et al.

(d) "Preservation of data dependencies" by the authors.

Fig.3 Example instances in which $m\{X,Y,Z\}$ holds, but $m\{X\cap S, Y\cap S, Z\cap S\}$ does not hold.

(a) Case 1 and Case 3.

X_1	X_2	Y_1	Y_2	Z_1	Z_2
1	1	0	0	1	1
1	1	1	1	0	0
0	0	1	1	1	0

(b) Case 2.

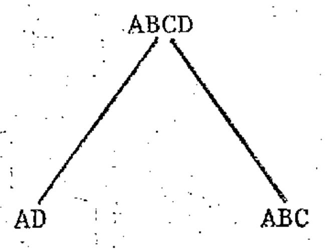
X_1	X_2	Y_1	Y_2	Z_1	Z_2
1	1	0	0	1	0
1	1	1	1	0	0
0	0	1	0	1	1

Fig.4

(a) An example instance r in which both $A \twoheadrightarrow D$ and $m\{A,B,CD\}$ hold, but $m\{A,B,C\}$ does not hold.

A	B	C	D
1	0	1	1
1	1	0	1
0	1	1	0

(b) Decomposition of R by $A \twoheadrightarrow D$.



(c) Decomposition of R by $m\{A,B,CD\}$ and $A \twoheadrightarrow D | C$.

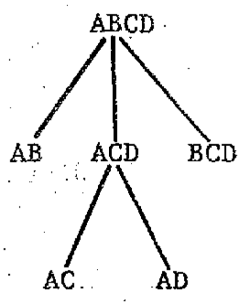


Fig. 5

(a) An example instance r_1 of R_1 with $A \to B$ and $A \twoheadrightarrow C$.

A	B	C	D
1	1	1	0
1	1	1	1
1	1	0	0
1	1	0	1

(b) An example instance r_2 of R_2 with $D_2=\phi$.

C	D	E
1	0	1
1	1	0
0	0	1

(c) $r_1 * r_2$

A	B	C	D	E
1	1	1	0	1
1	1	1	1	0
1	1	0	0	1

(d) An example instance r_3 of R_3 with $D_3=\phi$.

C	F
1	0
1	1
0	1

(e) $r_1 * r_3$

A	B	C	D	F
1	1	1	0	0
1	1	1	0	1
1	1	1	1	0
1	1	1	1	1
1	1	0	0	1
1	1	0	1	1

Fig.6

(a) Example instances r_1 and r_2 which satisfy the condition in Theorem 6.

r_1

X_1	X_2	Y_1	Z_1
1	1	0	1
1	1	0	0
1	0	1	0
1	0	0	0

r_2

X_1	Y_1	Z_1	W
1	1	0	0
1	0	1	1
1	0	0	0

(b) Example instance r_1, in which $X_1X_2 \twoheadrightarrow Y_1Y_2 | Z_1Z_2$ and $X_1X_2 \rightarrow Y_1 + Z_1$ hold.

r_1

X_1	X_2	Y_1	Y_2	Z_1	Z_2
1	0	1	0	1	0
1	0	1	0	0	1
1	0	1	1	1	0
1	0	1	1	0	1
1	1	1	0	1	1
1	1	0	1	1	1

DESIGN OF A DATABASE SYSTEM FOR SKULL LINE DRAWINGS PROCESSING IN ORTHODONTICS BASED ON THE RELATIONAL MODEL

Yoshinari KANAMORI and Yoshifumi MASUNAGA *

School of Dentistry, Tohoku University
* Research Institute of Electrical Communication,
Tohoku University, Sendai 980 JAPAN

Abstract

A database system was designed for processing of the skull line drawings used in orthodontics. The relational model was adopted as a data model. Twelve relations in the third normal form were deduced from the skull line drawings and the data on diagnoses and treatments of orthodontics.

In the database system, besides retrieving and updating data, the function called LINKER is attached to link the retrieved data with application programs used in orthodontic research.

The database system is now under the implementation which is primarily programmed in FORTRAN using a minicomputer system.

Contents

1. Introduction

2. Skull Line Drawings Processing

3. Data Model

4. Query Processing

5. Implementation

6. Conclusion

Acknowledgement

References

1. Introduction

Roentgen cephalograms are generally used to analyze the morphological characteristics of the craniofacial skeleton in orthodontics. To investigate such analysis, we have been developing a system which inputs the skull line drawings traced from the roentgen cephalograms by specialists and analyzes the morphological features of the skeleton applying some kinds of pattern processing techniques[1,2]. From the experiences of the system development, the following problems have been found:

(1) Almost all the requests desired to pick up some specific skull line drawings which satisfy a certain qualification.

(2) The system should provide such facility that the orthodontic specialists who are unskilled in using computer can easily produce application programs for their research.

(3) The features representing the skull line drawings were frequently replaced, appended and deleted with the research progression, because the analysis method of the skull line drawings is not yet perfectly clear at present. As a result, application programs were frequently rewritten. Therefore, it is desirable that the system provides such facility that the application programs are immune to these updates.

(4) The application programs frequently intended to combine the skull line drawings with the diagnosis and/or treatment informations on orthodontics organically.

To solve these problems, it seems to be natural to introduce a database system, where not only the skull line drawings can be retrieved, but also the pattern processings and the application programs can be executed using the retrieved data.

In this study, we describe a database system for processing of skull line drawings based on the relational model. In section 2, the outline of the skull line drawings processing is described and the processing results to be stored in the database are characterized. In section 3, 12 relations in the third normal form are proposed as the conceptual schema of the database. In section 4, the outline of query processing is shown utilizing a typical example. Finally, in section 5 the implementation of the system is discussed.

2. Skull Line Drawings Processing

The outline of the skull line drawings processing is shown in Figure 1.
The skull line drawings are input using a drum scanner with 0.2 X 0.2 mm accuracy. An example of input image is illustrated in Figure 2.
After the input, thinning, construction of graph data structure and preprocessing are performed. After the preprocessing, the information of one skull line drawing occupies about 32 KB.
In the structure analysis process, the input image is divided into 11 components(A to K) which consist of mandible, maxilla and so on as shown in Figure 3.
In the next process, the features of each component and those representing the relationships among components are extracted.
The feature extraction process has two stages:
feature point detection and feature production. An example of feature points and features is illustrated in Figure 4 in the case of mandible. Marked X represents the feature points and

α, β, γ and θ show features. After the feature extraction many kinds of application programs, for example, pattern classification, multivariate statistical analysis and etc., are carried out according to the research purposes.

From such processing, it is found that the imag㆔ data to be stored in the database consist of (a) X and Y coordinates of the skull line drawings after the structure analysis, and (b) X and Y coordinates of feature points and the features. In addition to these data, the clinical data which are useful with relation to the skull line drawings retrieval must also be stored in the database.

3. Data Model

The relational model[3] was adopted as a data model of our system from the following reasons:

(1) Orthodontic specialists who are unskilled in using computer could easily use the database system when the relational model is adopted.

(2) The features and the feature points may be appended, deleted and replaced according to the development of orthodontic researches, and therefore the application programs must be immune to these modifications. A high degree of data independence is easily obtained by the relational model.

In addition to data of the skull line drawings determined in section 2, necessary data on diagnoses and treatments were deduced from the discussions with orthodontic specialists.

After the normalization, the following 12 relations in the third normal form were deduced:

(1) CASE MANAGEMENT (CASE NO., NAME, SEX, BIRTHDAY, CASE DIVISION, INTERRUPTION, COMPLETION, STORED X RAY FILMS)

(2) DIAGNOSIS (CASE NO., DIAGNOSIS TIME, SEX, DIAGNOSIS AGE, HORIZONTAL FACIAL TYPE, VERTICAL FACIAL TYPE, FACIAL SYMMETRY, X RAY FILM NO.)

(3) GROWTH (CASE NO., DIAGNOSIS TIME, SEX, AGE, BONE AGE, DENTAL AGE, BODY HEIGHT, CAST DATA NO., ORTHOPANTOMO, X RAY FILM NO.)

(4) OPERATION (CASE NO., OPERATION DATE)

(5) TREATMENT INTERRUPTION (CASE NO., INTERRUPTION DATE)

(6) TREATMENT COMPLETION (CASE NO., COMPLETION DATE)

(7) ORTHODONTIC APPLIANCE (CASE NO., APPLIANCE NAME, START DATE, FINISH DATE)

(8) TOOTH EXTRACTION (CASE NO., TOOTH NAME, EXTRACTION DATE)

(9) FEATURE (X RAY FILM NO., GONIAL ANGLE, CHIN ANGLE,- - - , BODY LENGTH,- - - , F80)

(10) FEATURE POINT COORDINATE (X RAY FILM NO., FEATURE POINT NAME, X COORDINATE, Y COORDINATE)

(11) LINE DRAWINGS (X RAY FILM NO., COMPONENT NAME, LOCATION)

(12) TRACER & X RAY FILM ENLARGEMENT FACTOR (X RAY FILM NO., TRACER NAME, FACTOR)

Attribute names underlined show keys. The relation on the skull line drawings are (9) to (11). Relation (2) represents initial diagnosis. Relation (3) shows the change of growth at an interval of one year.

It is meaningless to retrieve individual X and Y coordinates constituting a component, because a retrieval unit of the skull

line drawings is component, for instance, maxilla or mandible. Consequently, relation LINE DRAWINGS in the relational database does not contain X and Y coordinates of skull line drawings, but has an attribute LOCATION. A value of LOCATION indicates the start place from which X and Y coordinates of a component are stored.
Examples of relations (1), (2) and (9) are illustrated in Figure 5, 6 and 7 respectively.

4. Query Processing

Query language used in the database system is similar to QUEL of INGRES[4]. In our system, one of the important problems in query processing is how to link an application program to the retrieved data. This linkage is the most significant property for the skull line drawings processing.
Figure 8 illustrates the outline of the query processing.
To make our discussion more clearly, let us observe the processing for the following query example:

" The case is mandibular prognathism, the initial diagnosis age is 14 to 25 years old, sex is female, the vertical facial type and the horizontal one are average and class 3 respectively, and Gonial angle(GOA) and Chin angle(CHA) of features are $120° \leq$ GOA $\leq 140°$ and $70° \leq$ CHA $\leq 85°$ respectively.
Find the mandible line drawings of the initial cases which satisfy these qualifications, and display the retrieved mandible on the space of which X coordinate and the origin are Mandibular Plane and Menton respectively. "

Note: Mandibular Plane means the line connecting between Gonion(GO) and Menton(ME) in orthodontics(see Figure 4). Consequently, the coordinates of the feature points GO and ME must be retrieved.

Notice here that the underlined part of the query corresponds to the application program which is named MADISP.
The former half of the query is written by QUEL as follows:

```
RANGE OF X1 IS D
RANGE OF X2 IS F
RANGE OF X3 IS FPC
RANGE OF X4 IS LD
RETRIEVE INTO Y (X1.XNO., X3.GOX, X3.GOY, X3.MEX,
X3.MEY, X4.LOCATION)
WHERE X1.DTIME="1" AND X1.HFT="CLASS 3"
AND X1.SEX="F" AND X1.VFT="AVERAGE" AND
"14" ≤ X1.DATE ≤ "25" AND
"120°" ≤ X2.GOA ≤ "140°" AND "70°" ≤ X2.CHA ≤ "85°"
AND X3.FPN="ME"  AND X3.FPN="GO" AND X4.CN="MANDIBLE"
AND X1.XNO.=X2.XNO. AND X1.XNO.=X3.XNO.
AND X1.XNO.=X4.XNO.
```

where relations of DIAGNOSIS, FEATURE, FEATURE POINT COORDINATE and LINE DRAWINGS are expressed by D, F, FPC and LD respectively.

Now, to process the latter half of the query, the following command is input after the above query was processed.

```
// LINK(A)
// MADISP
```

where LINK command performs to link the retrieved data to application program MADISP. LINKER of Figure 8 retrieves X and Y coordinates of the mandible line drawings from the skull line drawings database referring to the value of attribute LOCATION of the result relation Y, and stores these in the linkage file together with X and Y coordinates of GO and ME feature points. Here, the structure of the linkage file depends on the application program.

Therefore, variable A in LINK command indicates the file structure for application program MADISP. In our system, several types of linkage files are provided and accordingly users should choose which structure is suitable for an application program.

An example of display output is illustrated in Figure 9.

5. Implementation

The database system is under now the implementation on top of the DPS operating system for PANAFACOM Corporation U300 minicomputer system. The implementation is primarily programmed in FORTRAN.

As is evident from the above-mentioned example, it should be emphasized that the skull line drawings are frequently retrieved by specifying the features. Therefore, inverted files for some important features should be provided to improve a retrieval efficiency. To make inverted files, the value of these features will be quantized, for example, Gonial angle feature will be quantized with 10 degree step as shown in 100° ≤ GOA ≤ 110°.

6. Conclusion

The design of a database system for processing of the skull line drawings used in orthodontics was described.
The relational model was adopted as a data model because of a high degree of data independence and the easiness of developing application programs by orthodontic specialists.
As the conceptual schema of the database, 12 relations in the third normal form were deduced from the skull line drawings and the clinical data on diagnoses and treatments. The database system has the function called LINKER which links the retrieved data to application programs by LINK command.
The LINKER plays the most important role in processing of the query which intends to both retrieve the skull line drawings and process its results.

The database system is now under the implementation using FORTRAN, and could store about 600 skull line drawings in our minicomputer system which has a disk file of 30 MB capacity.

Acknowledgement

The authors are very grateful to Mr. J. Sugawara, School of Dentistry, Tohoku University, for his useful discussions in deducing twelve relations, and also thankful to Prof. T. Sakamoto School of Dentistry, Tohoku University, for his encouragement on this research.

References

[1] Kanamori, Y., Kido, K., Sugawara, J. and Sakamoto, T. : Pattern recognition system for line drawing image traced from roentgen cephalogram in orthodontics, MEDIS '78, Osaka Japan (Oct. 1978), B-5-8

[2] Sugawara, J., Kanamori, Y. and Sakamoto, T. : Analysis of mandibular form in orthodontics, MEDINFO 80, (1980)

[3] Codd, E.F. : A relational model of data for large shared data bank, Comm. ACM, 13, 6 (1970) 377-387

[4] Stonebraker, M., Wong, E., Kreps, P. and Held, G. : The design and implementation of INGRES, ACM TODS, 1, 3 (1976) 189-222

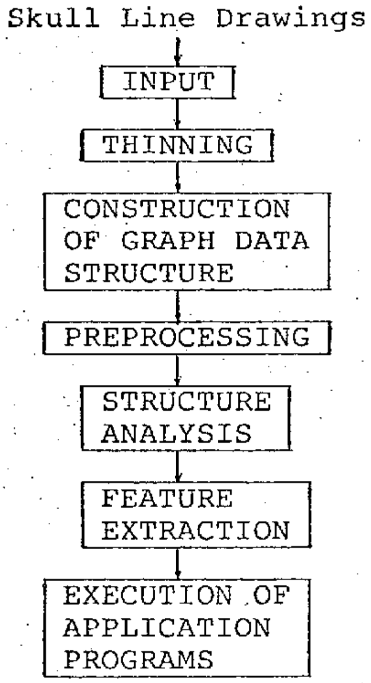


Figure 1 Outline of skull line drawings processing

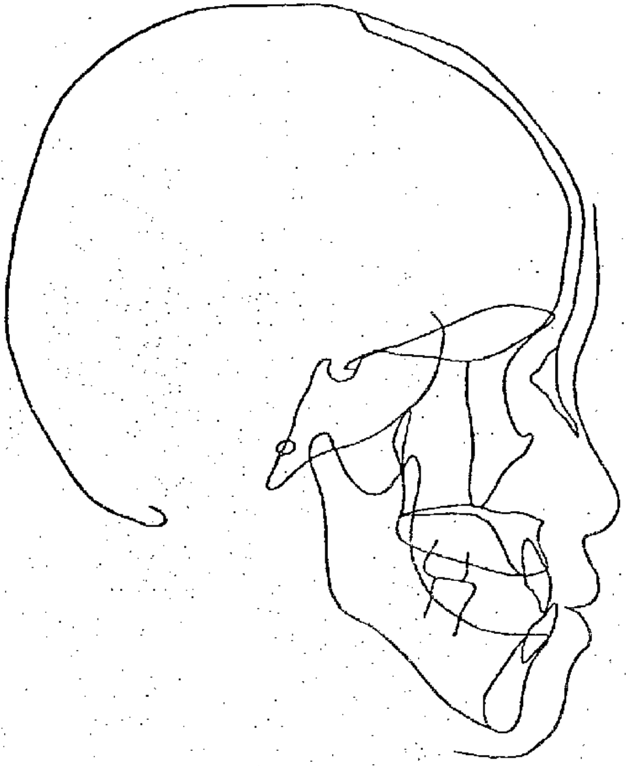


Figure 2 An example of skull line drawings

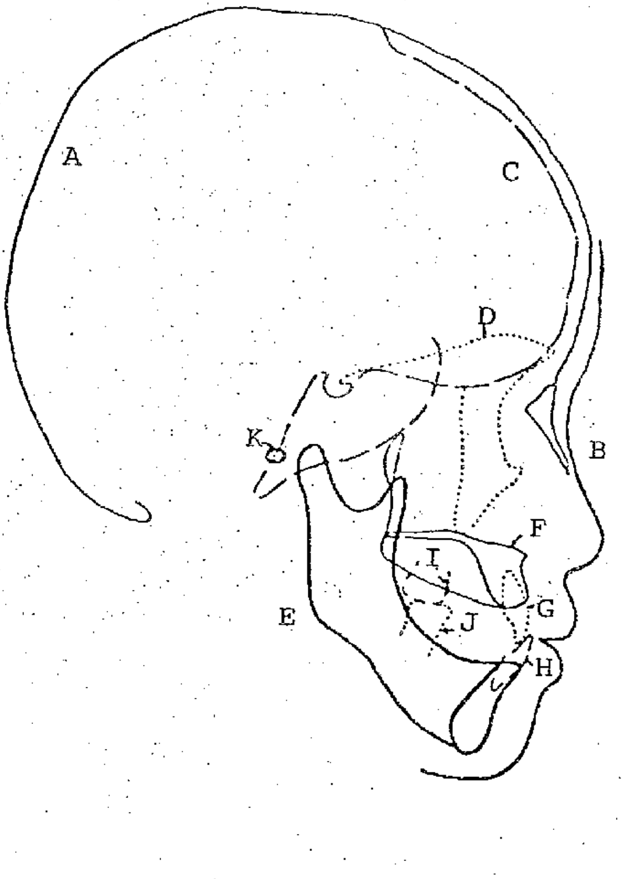


Figure 3 Eleven components of skull line drawing

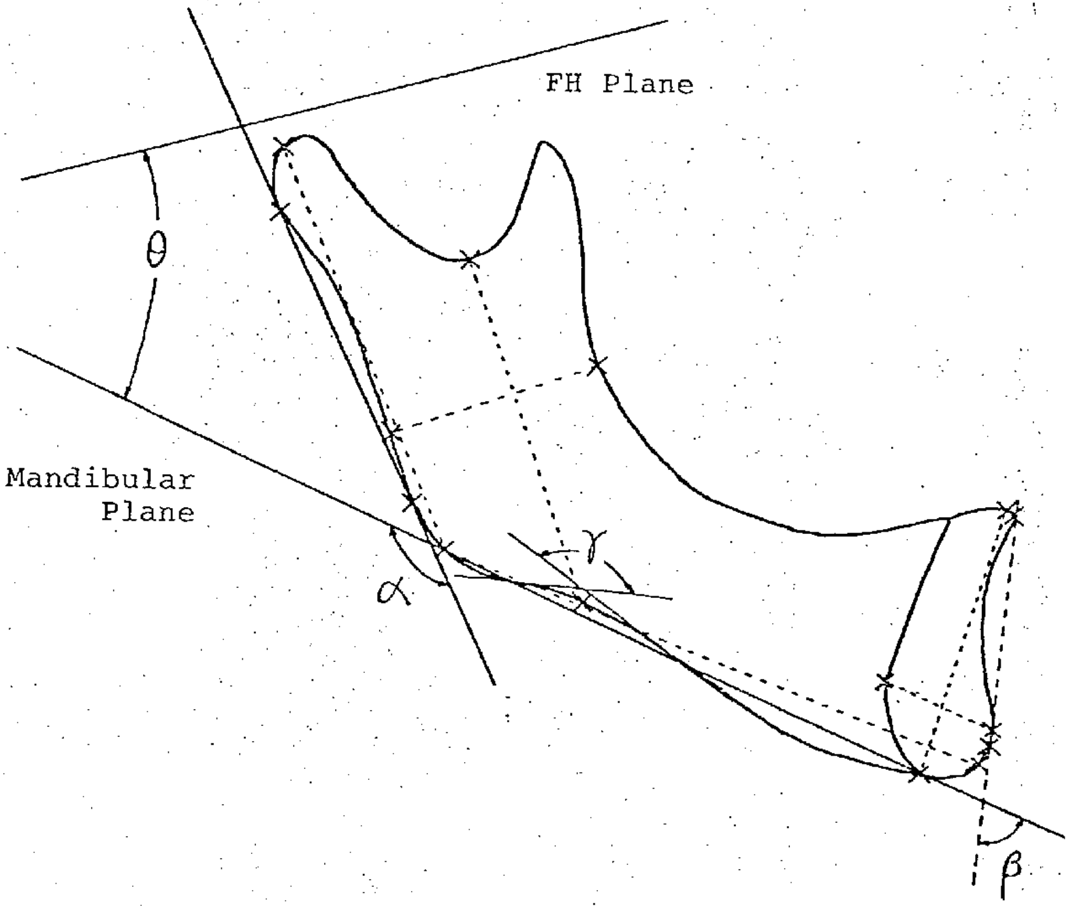


Figure 4 An example of features and feature points on mandible

CASE NO.	NAME	SEX	BIRTHDAY	CASE DIVISION	INTERRUPTION	COMPLETION	STORED X RAY FILMS
1000	Tanaka Taro	M	1970.3.5	General	No	No	3
1001	Yamada Hanako	F	1965.5.1	General	No	No	6
1002	Nakamura Jiro	M	1966.9.6	Educational	Yes	No	4
1003	Suzuki Mariko	F	1957.1.9	Surgical	No	Yes	2
1004	Sato Akiko	F	1960.7.2	Research	No	No	1

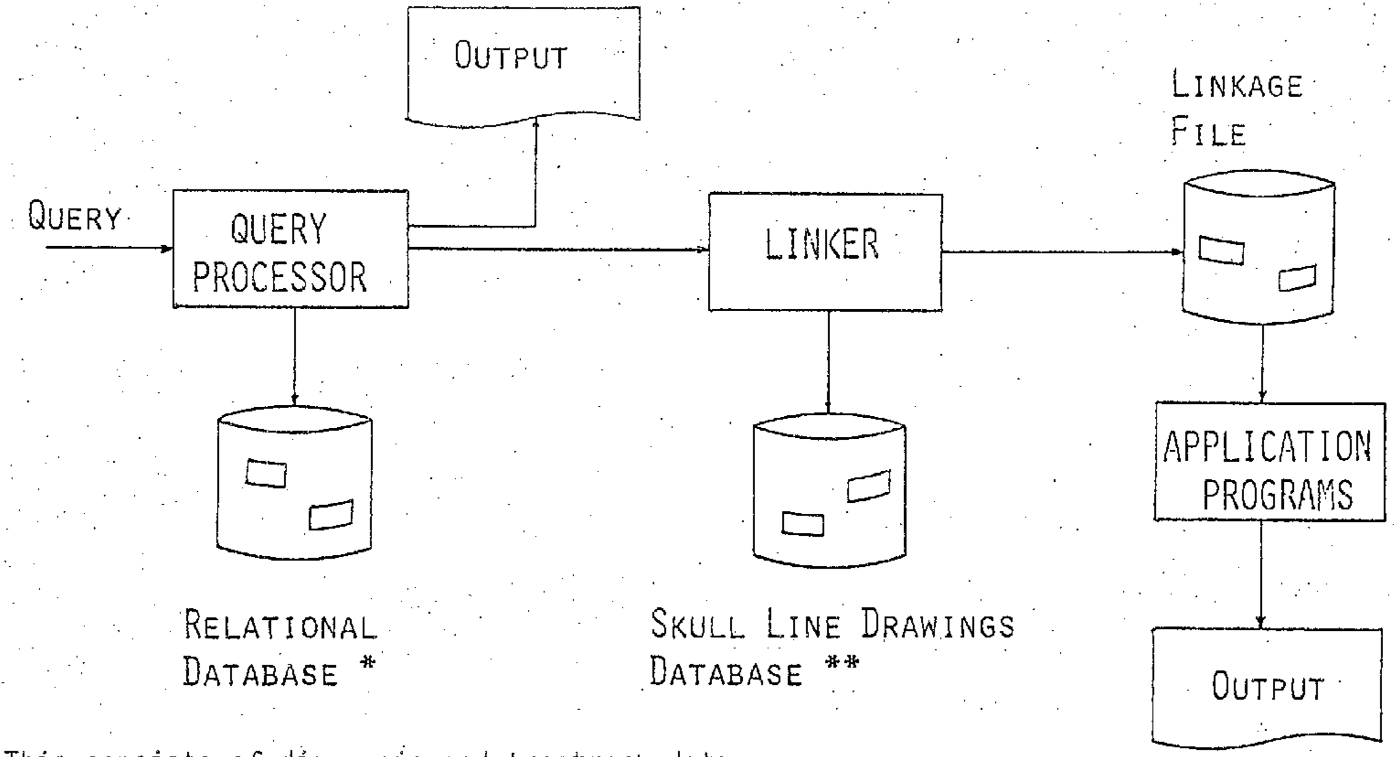
Figure 5 The relation CASE MANAGEMENT

CASE NO.	DIAGNOSIS TIME	SEX	DIAGNOSIS AGE	HORIZONTAL FACIAL TYPE	VERTICAL FACIAL TYPE	FACIAL SYMMETRY	X RAY FILM NO.
1000	1	M	7.2	Class 3	Long	S	5300
1001	1	F	9.4	Class 3	Average	S	5303
1002	1	M	6.9	Class 1	Average	S	5309
1003	1	F	21.5	Class 3	Short	AS	5313
1004	1	F	18.7	Class 2	Average	S	5315

Figure 6 The relation DIAGNOSIS

X RAY FILM NO.	COMPONENT NAME	LOCATION
5300	Maxilla	9800
5300	Mandible	9801
⋮	⋮	⋮
5300	Soft Tissue	9810
5301	Maxilla	9811
5301	Mandible	9812
⋮	⋮	⋮
5301	Soft Tissue	9821
⋮	⋮	⋮

Figure 7 The relation LINE DRAWINGS



* This consists of diagnosis and treetment data, features, feature point coordinates and location of skull line drawings.

** This consists of the X and Y coordinates of skull line drawings.

Figure 8 Outline of query processing

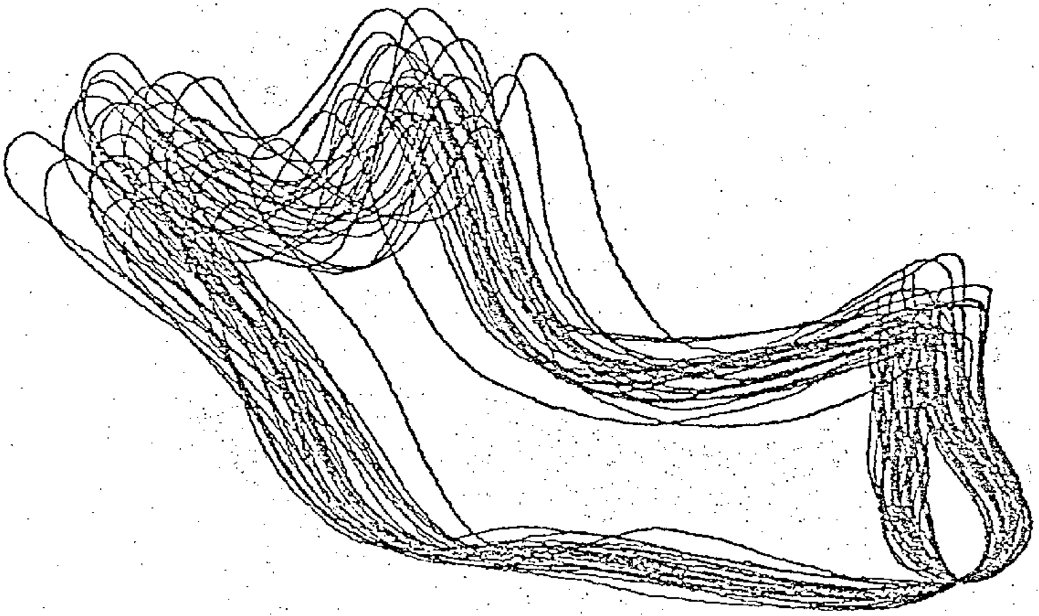


Figure 9 Output of query processing

JIPDEC Information Systems Seminar
on Semantic Aspects of Databases,
JIPDEC, Tokyo, Feb.21-22, 1980

IMPLEMENTATION AND EVALUATION OF RELATIONAL INVERTED STRUCTURE (RIS) FILES

Chung Le Viet[*], Yahiko Kambayashi[*],
Katsumi Tanaka[**] and Shuzo Yajima[*]

[*] Department of Information Science
Kyoto University
Sakyo, Kyoto 606, Japan

[**] College of Liberal Arts
Kobe University
Tsurukabuto 1-2-1, Nada, Kobe 657, Japan

Contents

1. Introduction

2. Basic definitions

   2.1 Relational Operations

   2.2 Abstracted Characteristics

3. Organization and Characteristics of Relational Inverted Structure (RIS) Files

   3.1 RIS File Organization

   3.2 Partition and Arrangement of domain values

4. Pseudo-Operations

5. Evaluation of RIS files

   5.1 Model of the Relational Inverted Structure (RIS) Files

   5.2 Storage Space Evaluation

   5.3 Retrieval Costs

   5.4 Update Costs

   5.5 Comparisons

6. Implementation of a Data Storage and Retrieval Subsystem

   6.1 System Description

   6.2 Query Processing Method

Acknowledgements

References

1. INTRODUCTION

The design and implementation of very large database systems are currently of growing interest. Particularly, in a relational database system the implementation of a file organization or an access path structure suitable for relational operation is important, since a well designed and implemented file organization can increase the efficiency of the system.

By "access paths" we mean physical connections, e.g., pointers or adjacency, between data [TSICL 77]. Two kinds of access paths, secondary indexes and pointer chains (links) are proposed for relational databases [ASTRB7606], [TSICL77], [STON7509]. Many implementation techniques for the two access paths are proposed. However, the following problems are not solved completely:

1. The full inversion problem: In order to support all operations equally (e.g. selections), all of attributes sometimes need be "inverted". This lead to a very large storage space required for all indexes, and in this case, the maintenance of these indexes can become very involved.

2. Access Path management overhead: In some system such as System R [ASTRB7606], indexes and links are realized by different implementation techniques. Therefore, creation and management of these access paths are difficult and costly.

3. Efficiency problem of relational operations: Basically, indexes and links are used to support selections, and join, respectively. However, the processing efficiency of joins when using links sometimes is not preferrable. Moreover, divisions

are not supported efficiently using the two access paths. Among access paths/file organizations proposed so far, no one is considered to be suitable for divisions.

In order to deal with these problems, we have proposed a file organization called Relational Inverted Structure (RIS) [TANAL7808]. The main features of RIS can be summarized as follows:

1. Domain oriented organization: A RIS file is organized as a collection of indexes of attributes belonging to conceptually joinable domains. In this meaning, a RIS file can be said to be a merged type index. By using a RIS file, both selections and m-way joins as well as binary joins can be supported efficiently.

2. Pseudo-operations [TANAL7808]: In a RIS file, hashed values of attribute values are incorporated to support pseudo-operations. In pseudo-operations, these hased values are compared instead of the attribute values themselves in proper operations. By using hashed values, pseudo-operations especially pseudo-division can be carried out. Furthermore, partial match can also be performed by choosing proper hash functions.

3. Realization of abstracted characteristics of data [LEVIK7911]: Some characteristics abstracted from the data which are needed for high level operations, such as improvement of null responses, integrity check, view update check, etc. can be stored in the directory part of a RIS file.

The main idea in designing RIS files is similar to that of Haerder's Generalized Access Path Structure [HAER7610], [HAER7809] and other implementation techniques for links [TSIC7502], [CODA71] However, the use of hased values for pseudo-operations and the abstracted characteristics described above make RIS files more powerful than others.

In this paper, the implementation and evaluation aspects of RIS files are discussed. Evaluation of storage space, retrieval and update costs are carried out for 2 versions of RIS files and conventional indexes. It will be shown that using RIS files, storage space can be saved in compared with full inversion index systems. For retrieval costs, consider only basic operations (without combination with pseudo-operations), when the number of attributes in a RIS file is small, RIS files are preferable than Indexes. In particular, in order to process joins efficiently conventional indexes approach must deal with the arrangement problem of indexes on disc packs. For update operations, especially updates of hashed values in RIS files the costs are not preferable. However, it will be shown that the difference between update costs of domain values and pointers (tuple id's) for RIS files and indexes can be neglected.

As a results of our evaluation, the problem of clustering of attributes belonging to a RIS file, and the problem of arranging RIS files on disc packs are briefly discussed. Clustering and arranging RIS files properly when designing them, processing efficiency is considered to be increased much more.

The implementaiton of a Data Storage and Retrieval Subsystem (DSRS) for a relational database system called ARIS (A Relational Information System) will also be discussed. Using RIS files as the main access paths, DSRS realizes the following retrieval facilities:

(1) Selections and m-way joins,

(2) Pseudo-selections and pseudo-joins,

(3) Logical operationsl, e.g. AND, OR, and

(4) Access to relation files and checking for hash collision.

Finally, query processing method in DSRS is presented using a sample query on our database.

2. BASIC DEFINITIONS

2.1 Relational Operations

Given sets D1, D2, ..., Dn(not necessarily distinct), R is a relation on these n sets, if it is a subset of the Cartesian product D1xD2x...xDn. Set D1, D2, ..., Dn are called the domains of R. When implemented, each tuple can be identified by an identification code(TID). A relation is usually represented by a table, which has n columns corresponding to the n domains, and many rows, each of which represents one n-tuple. A column is called an attribute and a domain is the set of possible values of its associated attribute(s). The range of an attribute is defined for each combination of an attribute and a relation to be the set of all values appearing in the attribute.

Let θ be any element in $\{=,\neq,<,<=,>,>=\}$ and A, B be attributes of relation R. The θ-restriction of R on attributes A, B is defined by $R[A\theta B] = \{r: r \in R, r[A]\ \theta\ r[B]\}$, where every element of R[A] is θ-comparable with every element of R[B], that is for every r[A], s[B], r[A] θ s[B] is true or false(not undefined). Here, r[A] denotes the value of the attribute A of tuple r. R[A] denotes the set $\{r[A]: r \in R\}$, that is the projection of R on A. When θ is =, we simply call θ-comparable domains "comparable domains". A special but usually used case of restriction is defined by $R[A\theta c]$, where c is a constant θ-comparable with R[A]. Such a restriction is called a selection.

The θ-join of relation R1 on attribute A with relation R2 on attribute B is defined by

$$R1[A\theta B]R2 = \{(r,s): r \in R1,\ s \in R2,\ \text{and}\ r[A]\ \theta\ s[B]\}$$

where (r,s) denotes the concatenation of tuples r and s.

Providing R1[A] and R2[B] are comparable, the division of $R1(A,\overline{A})$ on A by $R2(B,\overline{B})$ on B is defined by

$$R1[A \div B]R2 = \{r[\overline{A}]: r \in R1,\ R2[B] \subseteq R1[A,r(\overline{A})]\},$$

where $R[A,r(\overline{A})] = \{r'[A]: r' \in R1,\ r'[\overline{A}] = r[\overline{A}]\}$.

2.2 Abstracted Characteristics

In this section, the concept of abstracted characteristics (AC's) of data[LEVIK7911] is presented. The introduction of AC's aims at (1) Increasing the efficiency of query processing, (2) Improving the meaning of null responses, and (3) Providing a powerful view update checking facility.

AC's are defined to be the characteristics extracted from sets of tuples in relation(s). AC's have the following characteristics:

(1) There is not always one-to-one corresponding between AC's and their underlying data. In other word, underlying data cannot be recreated using some AC, the underlying data are more informative than their AC's.

(2) AC's must always reflect their underlying data: some AC's are time-independent; some of them are time-dependent. AC's of the latter class must be updated when the underlying data are changed.

(3) AC's are useful for query processing, update operations and other high level operations

There exist several methods of abstraction, therefore we can have several different AC's. But in order to keep small the maintenance cost of AC's, we use only AC's which are easy to compute and maintain.

A characteristic is global if it concerns all tuples in a snap-shot of a relation scheme and is satisfied by all the tuples, and is local if it concerns only a subset of tuples in a snap-shot of a relation scheme and is satisfied by all the tuples in the subset.

A characteristic is time-independent if it is satisfied at all snap-shots of the corresponding relation scheme, and is time-dependent if it is satisfied at some snap-shot(s) of the relation scheme but not all.

Here, we discuss only AC's which will be used in this paper. For other AC's [LEVIK7911] can be referred. One important AC is the time-independent global functional dependency (conventional FD) [CODD7105F]. The concept of FD can be generalized to define time-independent local FD, which is always satisfied by a specific subset of tuples in a relation, time-dependent global/ local FD, and bound numbers. Fig. 2.1illustrates an example of these FD's. The maximum (minimum) bound number of attribute B associated to attribute A in relation R(A, B) is the maximum (minimum) number of B values associated to every A value. In the case of FD, bound number becomes 1. The range and cardinality (the number of different values) of an attribute A are also useful AC's.

3. ORGANIZATION AND CHARACTERISTICS OF RELATIONAL INVERTED STRUCTURE (RIS) FILES

In this section, we introduce a new file organization called Relational Inverted Structure (RIS), which is the access path structure used in ARIS. The main design objectives of RIS files are as follows: (1) To increase the processing efficiency of conventional relational operations (restriction, join and division), (2) To decrease the overhead caused by introducing higher level relational operations such as query processing using abstracted characteristics, partial matching and relationship retrievals, (3) To obtain an easy access path generation/maintenance mechanism, and (4) To cope with some problems of partial inversion systems.

The major characteristics of RIS files are (1) To introduce the concept of "pseudo-operations" using hashed values of original attribute values, and (2) To embed some of abstracted characteristics in order to achieve the objectives described above.

In order to improve the efficiency of sequential searches in RIS files, partition and arrangement of domain values are proposed and discussed.

3.1 RIS File Organization

Usually, two kinds of accesses are required in a relational database: (1) Direct access to a certain tuple having a specified attribute value in a relation, and (2) Navigational access from tuples to tuples of different relations. The first kind of accesses can be implemented by indexes on attribute values.

Each value is associated with a list of TID's (Tuple IDentification Code) whose corresponding tuples have the value. The second kind of accesses can be represented by the connection of TID's whose corresponding tuples match on some attribute values. These two methods are implemented in System R as Images and Links [ASTRB7606], respectively.

The characteristics of RIS files can be summarized as follows:

(1) Domain-oriented organization: A RIS file is organized as a collection of indexes of attributes belonging to conceptually joinable domains. RIS files can be said to be a merged type index. Conventional facilities of indexes and links are realized by this organization, and n-ary joins can be supported efficiently as well as binary joins. Furthermore, domain-oriented queries such as relationship retrieval, for example retrieval of names of relations and attributes by specifying some domain values, can also be efficiently supported.

(2) Pseudo-operations [TANAL7808]: In RIS files, hashed values of attribute values are also incorporated to support pseudo-operations. In pseudo-operations, these hashed values are compared instead of the attribute values themselves in proper operations. The results obtained by pseudo-operations always contain those in proper operations. By using several pseudo-operations, such as pseudo-restriction, pseudo-join, and pseudo-division, we can reduce the number of accessions to base relation files. Furthermore, partial match can also be performed efficiently by the method.

(3) Use of abstracted characteristics of data[LEVIK7911]: Some abstracted characteristics needed for high level operations described in Section 2 can be stored in RIS files, and can be rapidly retrieved from RIS files. Some of them are time-dependent functional dependency, bound numbers, attribute range, cardinality of attribute, etc.

A RIS file is a modified inverted structure of Haerder's "Generalized Access Path Structure"[HAER7610],[HAER7809], which is a unified access path structure to support both indexes and links. The use of hashed values for pseudo-operations and the abstracted characteristics described above make RIS files more powerful than Haerder's Generalized Access Path Structure.

Let the domain of a given attribute A_i be denoted by $D(A_i)$. The term domain-related is defined as follows: (1) If $D(A_i) \cap D(A_j) \neq \emptyset$ then A_i and A_j are domain-related, denoted by $A_i \sim A_j$, (2) For distinct i,j and k, if $A_i \sim A_j$, and $A_j \sim A_k$, then $A_i \sim A_k$.

A RIS file, $RIS(A_i)$ is created for a set of domain-related attributes $\{A_{i1}, \ldots, A_{ik}\}$. Hereafter, the term "domain value x" in $RIS(A_i)$ means a value which belongs to at least one $D(A_{ij})$ $(j=1, \ldots, k)$.

A RIS file consists of two parts: the accession data part and the directory part.

The accession data part is a set of records each of which correponds to a domain value in the RIS file. These records are sorted on the domain values. A record consists of the domain value and a number of accession lists. Each accession list corresponds to a relation and an attribute, and has a list of TID's whose corresponding tuples have the domain value on the attribute. Incorporated into each TID is a list of hashed values of attributes other than the attribute A_{ij} in the relation. Fig.3.1 illustrates the structure of a record of RIS accession data part. The directory part of a RIS file is a set of tuples from the following relation:

RIS_DIR(REL_NAME,ATT_NAME,RIS NAME,RIS_ATT,MAX_VALUE, MIN_VALUE,ATT_CARDINALITY,MAX_BOUND,MIN_BOUND)

where ATT_NAME is the name of attribute in the relation given in REL_NAME; MAX_VALUE and MIN_VALUE are the maximum and minimum values of the attribute, respectively. Similarly, ATT_CARDINALITY is the number of different values. In the case of binary relations, MAX_BOUND and MIN_BOUND are the maximum and minimum bound numbers associated to the inverted attribute in RIS_ATT.

The following characteristics are realized in RIS files: (1) Time-dependent functional dependency and bound numbers: in the case of binary relations, if the bound numbers given in MAX_BOUND and MIN_BOUND are 1 then time dependent functional dependency betwwen the two attributes can be obtained. This can be expanded to the case of compound attributes, (2) Time-dependent attribute range: MAX_VALUE and MIN_VALUE give the range of corresponding attribute, (3) Number of attribute values(CARDINALITY).

From the directory part of a RIS file, time dependent global characteristics of attributes in the RIS file can be obtained. These characteristics can be accessed by relation and attribute names. When the underlying data are changed, these characteristics can be updated using accession data in the RIS file. Other time dependent local characteristics can be obtained directly from the accession data part.

Using the sample data in Fig.3.2, an example illustrating a RIS file is presented in Fig.3.3.

The basic question of what set of domain related attributes to be inverted in a RIS file is very important at the design time of RIS files for a given database.

In order to keep small the update maintenance cost of RIS files but remain their effectiveness for a given data base, the following considerations can be used as design criteria of RIS files

(1) To provide a partial inversion for each relation. That is not all attributes have a RIS file,

(2) To provide one RIS file for a set of domain-related attributes.

The size of a RIS file is directly proportional to the number of attributes having hashed values in the file. To keep the size of RIS files under an allowable limit, only the following attributes are selected to have hashed values:

(3) Attributes with high access frequency, and (4) Attributes whose values can be partially matched.

In fact, the selection of hashing function according to attribute types, and the objectives to have hashed values is also important because the length of hashed values is a main factor which affects the size of a RIS file.

A number of hashing functions can be constructed according to the following objectives:

(1) Order preserving,

(2) Partial matching, and

(3) Data compression.

Based on the type of data, and using query statistics, the selection of hashing functions must be carried out carefully.

As will be discussed in following sections, processing efficiency of relational operations, especially the join operations can be increased if the set of attributes to invert in a RIS file is chosen properly. Moreover, by partitioning attribute values in a RIS file according to "shared degree", and arranging these partitions, the efficiency of sequential searches of RIS files can be improved considerably. This will be discussed in detail in the next section.

3.2 Partition and arrangement of domain values

When using a RIS file to support a join operation, all the RIS file must be searched sequentially. Even the search ranges can be reduced using abstracted characteristics, when the ranges of attributes are almost the same, AC's can not be utilized.

Given a set of n attributes belonging to the same domain. For an attribute value, its "shared degree" is defined to be the number of different attributes sharing the value. There exists n shared degrees for a set of n attributes. Using this concept, we can partition domain values into partitions of different degrees. In addition, there exist n!/k!(n-k)! different combinations of k attributes. Therefore, a domain containing n attributes can be partitioned into 2^n-1 different partitions.

For example, domain containing two attributes, say A and B, can be partitioned into three partitions, two of shared degree 1 and one of shared degree 2.

When partitioning domain values, values belonging to the same attribute are devided into different partitions. However, in order to support sequential search concerning that attribute efficiently, these partitions are required to be arranged consecutively.

Ehrich and Lipski[EHRIL7601] proposed an algorithm which arranges partitions optimally with redundancy. Fig.3.4 shows an example of arrangement of three attributes by their algorithm.

Applying the partition and arrangement of domain values to out RIS design, the search range of each m-way join can be reduced considerably. In order to distinguish RIS files without and with domain value partition and arrangement in the evaluation in Section 5, we call the former RIS1 and the latter RIS2.

4. PSEUDO-OPERATIONS

In this section, we describe the query processing method using the following pseudo-operations: (1)pseudo-selection, (2)pseudo-join. Division is also efficiently processed by translating it into pseudo-division using RIS files[TANAL7808].

For simplicity, we only consider a two-variable query Q(X) AND Q(Y) AND P(X,Y). Here, Q(X) and Q(Y) are conjunctions of selection queries. That is, $Q(X) = \bigwedge_i Q_i(X)$ and each $Q_i(X)$ is a selection query such as $X.A_i$='constant'. P(X,Y) is a conjunction of join queries. That is, $P(X,Y)= \bigwedge_k P_k(X,Y)$ and each $P_k(X,Y)$ is a join such as $X.A_{k1}=Y.B_{k2}$. According to the allowable access modes of RIS files, we have two major query processing methods: Direct Access Method and Sequential Access Method.

The main idea of both methods is that we remove the TID's whose tuples can be proved not to be contained in the answer by by the comparision of hashed values, when searching RIS files. Based on this idea, we can reduce the number of accesses to the relation files.

[Direct Access Method]

Step 1: Select some $Q_i(X)$ and $Q_j(Y)$ in the query such that RIS files exist for attributes in $Q_i(X)$ and $Q_j(Y)$.

Step 2: (Processing of selections and pseudo-restrictions)
From the RIS file for $Q_i(X)$ $(Q_j(Y))$, select TID's, whose corresponding hashed values satisfy $\overline{Q}'(X)(\overline{Q}'(Y))$, together with the hashed values of attributes in $P'(X,Y)$.

Step 3: (Processing of pseudo-joins, relation file access and check)
From the obtained two sets of TID's, select only TID's

which satisfy $P'(X,Y)$. By using the remaining TID's, access corresponding relation files to obtain tuples, and examine whether each pair of tuples satisfies $Q(X)$ AND $Q(Y)$ AND $P(X,Y)$.

Here, $\overline{Q}'(X)=\bigwedge_{\ell\neq i}Q'_{\ell}(X)$ and $\overline{Q}'(Y)=\bigwedge_{m\neq j}Q'_{m}(Y)$. $Q'_{\ell}(X)$ is a pseudo-selection $H(X.A_{\ell})=H('\text{constant}')$ which means that the hashed value of $X.A_{\ell}$ is equal to the hashed value of the given constant. $Q'_{m}(Y)$ is also defined in the same manner. $P'(X,Y)=\bigwedge_{n}P'_{n}(X,Y)$; $P_{n}'(X,Y)$ is a pseudo-join $H(X.A_{n1})=H(Y.B_{n2})$. At the final step, the examination of $Q(X)$ AND $Q(Y)$ AND $P(X,Y)$ is required because of the possibilities of hash collisions in evaluating $\overline{Q}'(X)$ AND $\overline{Q}'(Y)$ AND $P'(X,Y)$.

[Sequential Access Method]

Step 1: Select some $P_{k}(X,Y)$ in the query such that a RIS file exists for the attributes in $P_{k}(X,Y)$. Access sequentially the RIS file.

Step 2: (Processing of join $P_{k}(X,Y)$, pseudo-selections and pseudo-joins) At each domain value, if there exist TID's of both relations and if the corresponding hashed values of these pairs of TID's satisfy $Q'(X)$ AND $Q'(Y)$ AND $\overline{P}'(X,Y)$ then select these pairs of TID's. Repeat this step for all domain values in the RIS file.

Step 3: (Relation file access and check) Using the selected pairs of TID's, access the corresponding relation files to obtain tuples and examine whether each pair of tuples satisfies $Q(X)$ AND $Q(Y)$ AND $P(X,Y)$.

Here, $Q'(X)=\bigwedge_{i}Q'_{i}(X)$, $Q'(Y)=\bigwedge_{j}Q'_{j}(Y)$ and $\overline{P}'(X,Y)=\bigwedge_{\ell\neq k}P'_{\ell}(X,Y)$. Other notations are similar to those in the Direct Access Method.

Partial matching queries, for example X.TITLE='*database*' are usually translated to pseudo-selection, where hashed values can be applied.

5. EVALUATION OF RIS FILES

5.1 Model of the Relational Inverted Structure files

In this section, we describe a simple model of the physical structure of the RIS file on which the evaluations followed will be based.

In our model, a RIS file is assumed to be built on a B+tree like data structure[COME7909] of order d. It consists of two parts, the directory part and the leaf part(see Fig.5.1). The depth of the index part is known as $\log_d m$, where m is the number of keys in the leaf part.

There can be two realizations of a logical record of RIS file which differ from each other in choosing the key: (a) Key= domain value, and (b) key = domain value, relation number, attribute number. In the method (a), a phical record realizes a logical record and less storage space is required than in the method (b). In the method (b), a number of consecutive physical records realize a logical record. In order to compare RIS files with conventional secondary index files, we choose the method (b), because this organization can also be applied for the latter.

In the evaluation followed, a RIS file for a set A_i of attributes is considered. The file is denoted by RIS(A_i), and attributes in A_i are denoted by A_{ij}, j=1, ..., k. Table I summarizes the parameters used in the following Section.

5.2 Storage Space Evaluation

In this section, the evaluation of storage space for RIS(A_i) is discussed and compared with that of conventional secondary indexes. For simplicity, only storage space for leaf pages associated with logical records of RIS(A_i) is taken into consideration.

A physical record of a RIS file consists of a domain value an an accession list(concerning only this part, this organization is similar to that of secondary indexes). In an accession list, there are a relation number, an attribute number, the number of tuples, a number of TID's and associated hashed values.

In RIS(A_i), the number of physical records is given by

$$n_i = \sum_{j=1}^{k} D(A_{ij})$$

where $D(A_{ij})$ is the number of different values of attribute A_{ij}. Then average storage needed for n_i records is:

$$S_{DV} = n_i * l_d \qquad (5-1)$$

Note that if we choose the organization (a) in Section 5.1, n_i must be less than or equal to that in this case of organization (b).

The storage need for n_{ij} accession list of relation R_j is given by:

$$S_{Rj} = N_j*(l_T+l_h*r_{hj}) + n_{ij}*l_c \qquad (5-2)$$

Then the total storage S_i needed for n_i records becomes:

$$S_i = S_{DV} + \sum_{j=1}^{k} S_{Rj}$$

$$= n_i*l_d + \sum_{j=1}^{k} (N_j*(l_T+l_h*r_{hj}) + n_{ij}*l_c) \qquad (5-3)$$

As mentioned in Section 3, a RIS file has the access ability of both index and link. Therefore, the storage S_i should be compared with the total storage of corresponding indexes, provided that link is realized by a number of indexes.

Here, assume that the index for attribute A_{ij} has a set of records, each consists of one domain value, a tuple number indicator and a number of TID's. In this case, relation and attribute number are not necessary to be considered.

The storage I_{ij} needed for the index of A_{ij} is given by

$$I_{ij} = N_j * l_T + n_{ij}(l_d + l_{nt}) \qquad (5-4)$$

The total storage I_i for all indexes, each for one A_{ij} becomes

$$I_i = \sum_{j=1}^{k} I_{ij} = \sum_{j=1}^{k} (N_j * l_T + n_{ij}(l_d + l_{nt})) \qquad (5-5)$$

In case of complex queries where hashed values can be used for pseudo operations, RIS files can be compared with a full inversion system, whose storage is as follows:

$$I_f = m\sum_{j=1}^{k} I_{ij} = m * I_i$$
$$= m\sum_{j=1}^{k} (N_j * l_T + n_{ij}(l_d + l_{nt})) \qquad (5-6)$$

where m is the average number of attributes in a relation.

The difference D_i between (5-3) and (5-5) can easily derived:

$$D_i = S_i - I_i$$
$$= \sum_{j=1}^{k} (N_j * l_k * r_{hj}) + n_{ij} * l'_c \qquad (5-7)$$

where $l'_c = l_c - l_{nt}$.

The difference D_f between (5-3) and (5-6) is:

$$D_f = S_i - I_f$$
$$= D_i - (m-1)I_i \qquad (5-8)$$

From (5-7) and (5-8), it can be concluded that an amount of D_i space is paid for hashed values in a RIS file, however by the sake of this, we can save D_f space in comparison with a full inversion system.

5.3 Retrieval Costs

Retrieval costs for selections and join when using RIS1, RIS2(RIS1 with domain values partitioned), and conventional secondary index, or index for short, are derived and compared below. First, processing procedures are explained for each case. Then the costs are derived and compared. Basically, the number of pages fetched is assumed to be the relevant measure in our evaluation. Table II summarizes parameters will be used in the evaluation.

5.3.1 Retrieval Costs for Selection

Consider the selection $R_k.A_j$='a'. The following procedure is applicable to all the access paths RISi, RIS2, and Index.

Step 1: The access path (RIS1, or RIS2, or Index) is directly accessed to fetch the desired sets of TID's.

Step 2: The corresponding relation file is accessed using the set of TID's obtained in Step 1.

Let the costs(in accessed pages) required by step 1 for RISi, RIS2, and Index be S_i, S_2 and S_I, respectively. In the case of RIS1 and Index, the directory is accessed only once, thus S_1 and S_I become:

$$S_1 = \log_d(m) + 1$$

$$S_I = \log_d(m_j) + 1$$

where m and m_j is the number of different values in RIS1(RIS2) and in the Index of A_j, respectively.

In the case of RIS2, the directory is accessed once for each partition in the worst case. Let h be the number of partitions containing $R_k.A_j$ values, the directory is accessed (h+1)/2 times in average. Therefore,

$S_2 = h*\log_d(m) + 1$ in the worst case,

$S_2' = (1/2)*(h+1)*\log_d(m) + 1$ on the average

The cost S_r required by step 2 is the same for all the three access paths:

$S_r = \log_d(P_k) + F_k*P_k$ if R_k is clustered on A_j

$S_r' = (\log_d(P_k) + 1)*F_k*N_k$ if R_k is not clusted on A_j

We can compared the three costs concerning only step 1. For RIS1 and Index, in case the RIS1 file contains values of other attributes, $m > m_j$. However, if we choose d to be as large as possible, e.g. d=200, the difference between S_1 and S_I (which is of log order) becomes small. In the case of S_1 and S_2, when $h \geqq 2$, $S_2 > S_1$. However, S_2 can be improved if a directory index on domain values, which gives the identifier of the partition containing the corresponding domain value exists. By using this directory, redundant accesses when using RIS2 can be reduced.

5.3.2 Retrieval Costs for Join

We consider the join operation $R1.A_k = R2.A_h$, the case of m-way joins can be derived similarly. The RIS files of A_k and A_h are denoted by RIS1(a)(i.e. without domain value partition), and RIS2(a)(i.e. with domain value partition). The Index files of A_k and A_h are denoted by INDEX(A_k) and INDEX(A_h), respectively. We consider the procedure and retrieval costs in pages for each case. For simplicity, we assume block factor I in RIS1, RIS2 and Index are the same(For parameters used in the evaluation, see Table II).

(A) Procedure using RIS1(A)

Step 1: RIS1(A) is accessed sequentially to obtain a set of (R1.TID,R2.TID) pairs in which, R1.TID and R2.TID are corresponding to the tuples having the same value on A_k and A_h in R1 and R2, respectively,

Step 2: Relation files of R1 and R2 are directly accessed using the TID's obtained in Step 1.

In order to sequentially search all the file RIS1(A), step 1 requires to fetch m/I pages.

Suppose we use TID's obtained in step 1 to access relation files directly, step 2 costs

$$J1*P1*(\log_d(P1)+1) + J2*P2*(\log_d(P2)+1)$$

if the two files are clustered on the join attributes, and

$$J1*N1*(\log_d(P1)+1) + J2*N2*(\log_d(P2)+1)$$

if the two files are not clustered on the join attributes.

In the following cases of RIS2 and INDEX, the costs of step 2 are the same as given above, therefore we will not derived them again.

(B) Procedure using RIS2(A)

The procedure is similar to that of RIS1, except for that in step 1, not all the RIS2 file is required to be searched. only those partitions containing values of both R1.A_k and R2.A_h are accessed. Let C_{kh} be the number of domain values must be read to perform step 1, we have the cost of step 1 as follows: C_{kh}/I.

(C) Procedure using Indexes

Step 1: INDEX(A_k) and INDEX(A_h) are accessed parallely in sequential mode. As the two files are accessed, the TID's of common domain values in both indexes are obtained.

Step 2: see step 2 in (A).

In this case, step 1 costs:

$m_k/I + m_h/I$

In order to compare access costs for join using RIS1, RIS2 and Indexes, we must take into consideration the fact that using indexes, at least two indexes are sequentially accessed at the same time. But in the case of RIS1 and RIS2, only one file is accessed at the same time.

Let T_r be the average time to read the next record in a sequential access mode. Let T_s be the average time to move the disc head from a record of one file to a record in another file.

The time needed to fetch the next page when sequentially search RIS1(A) or RIS2(A) can be considered as T_r. To read the first page of RIS1(A), it costs $T_s + T_r * \text{Log}_d(m)$ to traverse the index part, therefore cost in time T_1 for join operation using RIS1(A) becomes as follows:

$$T_1 = T_s + (\log_d(m) + m/I) * T_r$$

Similarly, cost T_2 for the case of RIS2(A) is:

$$T_2 = T_s + (\log_d(m) + C_{kh}/I) * T_r$$

In the case of indexes, suppose that the values of A_k and A_h in both indexes are uniformly distributed. That is, in order to process the join, they can be read interchangeably one block after another. Then the cost in time T_I when using indexes becomes:

$$T_I = 2 * T_s + (\log_d(m_k) + \log_d(m_h) + m_k/I + m_h/I) * T_r$$
$$+ \sigma_{kh} * \min(m_k/I,\ m_h/I) * 2 * T_s$$

where

$$\delta_{kh} = \begin{cases} 0 & \text{if the two indexes are on different disc packs} \\ 1 & \text{if the two indexes are on the same disc pack} \end{cases}$$

We use the following example to compare the three cases. For simplicity, we suppose $m=2m_k=2m_h=1000$, $C_{kh}=2*m/3$, $I=2d=20$. T_s can be taken as average seek time, approcimately 25 msec. T_r is the sum of latency time and transfer time, which is approcimately 9 msec, suppose that the record size is of 300 byte length for both RIS and index. Then we have the ratio D_{1I} and D_{21} between T_1 and T_I, and T_2 and T_1, respectively as follows:

$$\begin{aligned} D_{1I} = T_1/T_I &= (T_s + (\log_{10}(1000) +50)*T_r) \\ &\quad /(2*T_s + (2*\log_{10}(500) + 50)*T_r + \delta_{kh}*OV \\ &= 0.90 \quad \text{if } \delta_{kh} = 0 \\ &= 0.27 \quad \text{if } \delta_{kh} = 1 \end{aligned}$$

where OV stands for $\min(m_k/I, m_h/I)*2*T_s$.

5.4 Update Costs

In a RIS file (RIS1 or RIS2), there exist three basic update operations:

1. Updates caused by the deletion/insertion of a tuple in the underlying relation file,

2. Modification of domain values caused by the update of some tuples in the underlying relation files, and

3. modification of hashed values.

In the following we consider only updates of the class 1, which will become the basic for those in classes 2 and 3.

(A) Update Costs for RIS1: Consider the deletion/insertion of a tuple t into relation R with $R.A_h$ = 'a'. The updates of RIS1 file caused by this update will be considered.

Step 1: Locate the record whose domain is 'a', assume that there exists such a record in RIS1(A_h)).

Step 2: Delete/Insert the TID from/into the located record.

In order to delete a TID or a domain value, a mrking bit is used. For simplicity, it is assumed that there is enough space in the located record to insert a new TID, and no compaction of record is considered when a deletion is carried out.

Let I_1 and D_1 be the costs in pages of insertion and deletion, respectively. They can be given as follows:

$$I_1 = D_1 = 2*(\log_d(m) +1)$$

(B) Update Costs for RIS2.

One property of the structure of RIS2 is that the intersection of domain value sets of every two partitions is empty. Therefore, update of a tuple in RIS2 may cause the corresponding record be deleted and another be inserted. These "chaining updates" occur when the update of a tuple changes the shared degree of the corresponding record.

In the case of deletion, if there is no "chaining updates" the procedure given in (A) can be applied. Then the average cost D_2 can be readily derived as follows:

$$D_2 =((k+3)*\log_d(m) + 2)/2$$

where k is the number of partitions containing values of R.A_h.

In the case that there exists "chaining update", the procedure requires one more write, thus the average cost D_2' in this case is:

$$D_2' = ((k+5)*\log_d(m))/2 + 3$$

For insertion, there may be three possibilities:

(a) Insertion of a new record with a domain value already existed,

(b) Insertion of a new record with respect to an attribute in the RIS2 file, and

(c) Insertion of a new record with a new domain value.

The costs I_{2a}, I_{2b}, and I_{2c} of the cases (a), (b), and (c) respectively, can be summarized as follows:

I_{2a} = $((k+3)*\log_d(m))/2 + 2$ on the average

I_{2b} = $((k+5)*\log_d(m))/2 + 3$ on the average

I_{2c} = $(k+1)*\log_d(m) + 2$

From the expressions derived above, it is clear that if k is large, the update efficiency of RIS2 is not desirable. In this case, a directory index on domain values, which gives the address of the partition containing the corresponding domain value can be created to improve update efficiency.

(C) Update Costs for Indexes

The procedure for index is the same as in (A). The insertion and deletion costs I_I and D_I in this case are:

$I_I = D_I = 2*(\log_d(m_h) + 1)$

where m_h if the cardinality of A_h.

The difference between update costs of RIS1 and Indexes is of log order, therefore if we choose d to be large, it can be neglected.

5.5 Comparisons

We have derived and compared storage spaces, retrieval and update costs of RIS1, RIS2 and indexes. The followings can be concluded from our evaluation:

(1) In the case of indexes, in order to perform m-way joins efficiently, the m corresponding indexes must be arranged properly on different disc packs to decrease the seek time from one file to another. This is an,interesting problem that we call the file arrangement problem.

(2) A RIS1 file of n attribute is much more suitable for m-way joins in compared with binary joins.

(3) RIS2 files are useful when the number of partitions is small and when there are not many update requests. In particular, efficiency is increased when the cardinalities of partitions with shared degrees greater than 1 are small in compared with m, the total cardinality of the set of attributes.

(4) One trade-off is paid for the power of RIS files due to pseudo-operations is the update cost concerning hashed values.

(5) Finally, we note that at the design time of RIS files, the problem of selecting proper domain related set of attributes is very important. Our approach to this problem is to use a clustering algorithm to divide the original domain related set to smaller subsets, and then arrange the RIS files corresponding to these subsets on different disc packs so that the navigation accesses between these RIS files can be carried out efficiently like the case of indexes above.

In conclusion, we can say that RIS files with pseudo-operations, and useful design approach, e.g. domain clustering, file arrangement, are believed to be powerful access path structures and are suitable for relational operations.

6. IMPLEMENTATION OF A DATA STORAGE AND RETRIEVAL SUBSYSTEM

We have been developing a relational database system called ARIS(A Relational Information System). ARIS is a database system based on the relational data model[CODD7006]. This system is designed and implemented to handle bibliographic data as well as ordinary business oriented data, and it provides flexible and efficient data management facilities.

ARIS consists of the following major parts: User Interface, Information Retrieval Support Subsystem, Data Language Subsystem, Language/Access Path Interface, and Data Storage and Retrieval Subsystem. Fig.6.1 shows the configuration of ARIS. A detail description of ARIS can be found in [KAMBK7911], [YAJIK8001].

In this section, the implementation aspect of Data Storage and Retrieval Subsystem which uses RIS files as the access path structure is discussed. Query processing method in Data Storage and Retrieval Subsystem(DSRS) will also be presented.

6.1 System Description

The system works on a database constructed by a set of relations. In order to support accesses to the database such as data retrieving, updating, etc., the relational Inverted Structure (RIS) files are used. Using RIS files as the main access paths, DSRS realizes the following major facilities:

(1) Pseudo-selections and pseudo-joins which are based on selections,

(2) Pseudo-selections and pseudo-joins which are based on joins,

(3) Logical operations, e.g. AND, OR,

(4) Access to relation files and checking for hash collision, and, (5) Creating and updating relation files and RIS files.

The major characteristics of DSRS are as follows:

(1) Simultaneous processing of m-way joins, and

(2) Realization of pseudo-relational operations using hashed values.

DSRS consists of the following modules: Access Path Manager (APM), Data Retrieval Module(DRM), Data Storage Module(DSM), File Generator(FG), and Output Module(OM)(see Fig.6.1). Explanation of these and other related modules is followed.

In the Language/Access Path Interface, the Access Path Selector(APS) receives the parsed query from the Data Language Subsystem, and selects an optimal access path using information about RIS files with the aid of Physical Optimizer(PO).

(1) Access Path Manager (APM): APM receives the access path consisting of a number of access sequences from APS and call DRM and DSM to perform the specified operations. APM also performs logical operations combining the access sequences. Finally, APM calls the Output Module to output/display the results.

(2) Data Retrieval Module (DRM): DRM consists of Direct Access Module, Sequential Access Module, Scan Module and TID-Access Module(see Fig.6.2). Direct Access Module(DAM) access directly to RIS files and process selections, pseudo-selections, pseudo-joins and some logical operations. Sequential Access Module(SAM) perform sequential access to RIS files and has the ability to process joins, pseudo-join, pseudo-selections and some logical

operations. Scan Module(SM) makes sequential access to relation files and has the ability to process relational operations and logical operations. TID-Access Module access relation files directly/sequentially through TID's and checks for hash collision due to pseudo-operations.

(3) Data Storage Module (DSM): DSM performs insertion, deletion and update of relation files and RIS files. As relation files are updated, integrity check is carried out by Integrity Check Module.

(4) File Generator (FG): FG consists of Relation File Generator (RELGEN), and RIS file Generator(RISGEN). RELGEN receives a set of tuples corted on some attributes and creates one file for each relation. Each page in a relation file has the organization shown in Fig.6.3. RISGEN receives a number (may be one) of relations and creates a RIS file for specified attributes in these relations. For organization of a RIS page, see Fig.6.4.

6.2 Query Processing Method

In ARIS, a user query is evaluated in five phases: (1) Query interpretation, (2) Access Path Selection, (3) Access to RIS files, (4) Access to relation files, and (5) Post-processing and outputting. These phases may be repeated in evaluating a user query.

The Data Language Subsystem performs operations in phase (1). Phase (2) is due to Access Path Selector(APS). DSRS performs operations in remaining phases. The Data Language Subsystem carried out both semantic and syntactic checks on the user query and procduces to the Language/Access Path Interface a parsed query expressed in an internal data structure. The parsed query is constructed by relational operations and logical operations such as AND, OR, etc.

Each access path selected by APS corresponding to a user query has some or all following access sequences:

(a) Direct Access Sequence,

(b) Sequential Access Sequence, and

(c) Scanning Access Sequence,

where (a) and (b) include access to RIS files in direct access mode and sequential access mode, respectively, and (c) includes sequential access to relation files.

(a), (b) and (c) are processed by Direct Access Module(DAM), Sequential Access Module(SAM), and Scan Module(SM), respectively.

In fact, APS decomposes the parsed query into subqueries which can be processed efficiently by DAM, SAM and SM. In order to reduce the sizes of intermediate results, these subqueries may overlap with each other.

For example, consider the sample database and RIS files given in Fig.6.5. We will use the query given in Fig.6.6 and Fig.6.7 to illustrate the query processing method in DSRS.

In SM, relation files are sequentially searched and tuples are checked for whether or not they satisfy the corresponding subquery. Intuitively, SM is not efficient as DAM, and SAM because SM searches relation files exhautively, where DAM and SAM with pseudo-operations need only to access relation files directly. In the first version of ARIS, SM is only caleed to process high(low) selections, operations without RIS file, etc. In the following, DAM and SAM will be presented in detail.

Direct Access Module (DAM): This module realizes the facility of Direct Access Method discussed in Section 4. However, DAM does not access relation files, it only produces the TID set of tuples may satisfy the direct access sequence(subquery).

Subquery of n variables to be processed by DAM must include a set S of at least n selections, one or more for each variable. A subquery can include a set PJ of joins and a set PS of selections both to be processed by pseudo-operations. These sets of operations must be connected by logical AND's. Let $R_t \in R$, where R is the set of the n relations. Let $S_{ti} \in S$, by S_{ti} we mean a selection of relation R_t; similarly, $PS_{tk} \in PS$ stands for a selection (to be processed by pseudo-operation) of relation R_t. The subquery must be in the following conjunctive normal form:

$$\bigwedge_t D(S_{ti}) \bigwedge_j PJ_j \bigwedge_t \left(\bigwedge_\ell D_\ell(PS_{tk})\right) \qquad (6\text{-}1)$$

where $D(S_{ti})$ is a disjunction of S_{ti} of the same attribute in R_t; $D_\ell(PS_{tk})$ is a disjunction of $PS_{tk} \in PS$; and $PJ_j \in PJ$.

In our example, the subquery in Fig.6.8 is processed by DAM.

In order to process a subquery, DAM uses the RIS files for attributes contained in selections belonging to S, and performs the following steps:

Step 1: For each $S_i \in S$, access directly the corresponding RIS file to obtain a set of TID's and hashed values which will be used in pseudo operations later.

Step 2: Using hashed values to perform pseudo-selections which belong to PS. TID's of hashed values which do not match the hashed value of the given constant in a pseudo-selection are omiited at this step. As hashed values are checked, logical operations are also carried out.

Step 3: Using hashed values to perform pseudo-joins in PJ. The only permitted logical connection in this case is AND, therefore, pseudo joins can be processed serially and at each step, each join is processed using the results of the joins already performed until that step.

Sequential Access Module(SAM): This module realizes the facility of Sequential Access Method discussed in Section 4. SAM performs m-way joins vasically. An m-way join joins m relations at the same domain. Using RIS file of the domain, this operation can be performed readily and efficiently.

Subqueries of m variables processed by Sam include an m-way join, and a set PS of pseudo-selections to be performed by pseudo-operations. The subquery must be in the following conjunctive normal form:

$$\bigwedge_{i} J_i \bigwedge_{t} \left(\bigwedge_{\ell} D_{\ell}(PS_{tk}) \right) \qquad (6\text{-}2)$$

where $\bigwedge_{i} J_i$ stands for the m-way join.

In our example, the subquery in Fig.6.9 is processed by SAM. The SAM procedure is similar to that of Sequential Access Method in Section 4, thus it is not considered again here.

Both DAM and SAM produce the set of TID's of tuples satisfying the corresponding subqueries. As described in Section 6.1, APM receives these sets of TID's and performs logical operations. In our example, APM must process the AND operation which connects the two subqueries in Fig.6.8 and 6.9. In fact, APM produces the intersection of the two TID sets provided by SAM and DAM.

Finally, the TID-Access Module is called to access relation files using the resulting TID sets produced by APM, and to check for hash collision and other operations which are not processed by DAM and SAM.

ACKNOWLEDGEMENTS

The authors are grateful to the colleagues in Yajima Laboratory in Kyoto University. Special thanks are due to Mr. Yasuo Imai, Mr.Mikio Yamada, and Mr. Toshikazu Kato for their co-operation during the development of our relational database system ARIS.

This work is partly supported by the Science Foundation Grant of the Ministry of Education, Science and Culture of Japan.

REFERENCES

[ASTRB7606] ASTRAHAN,N.M. et al., "System R: Relational Approach to DataBase Management", ACM Trans. on Database Systems, Vol.1, No.2, pp.97-137, June 1976.

[BLASE77] BLASGEN,M.W. and ESWARAN,K.P., "Storage and Access in Relational Databases", IBM System Journal, No.4, pp.363-377, 1977.

[CARD7505] CARDENAS,A.F., "Analysis and Performance of Inverted Database Structures", CACM, Vol.18, No.5, pp.253-263, May 1975.

[CODA71] Committee on Data Systems Languages, "CODASYL Data Base Task Group Report", ACM, New York, 1971.

[CODD7006] CODD,E.F., "A Relational Model of Data for Large Shared Data Banks", CACM, Vol.13, No.6, pp.377-387, June, 1970.

[CODD7105F] CODD,E.F., "Further Normalization of the Data Base Relational Model", Courant Compt. Sci. Symposia, pp.34-64, May 1971.

[COME7906] COMER,D., "The Ubiquitous B-tree", ACM Computing Surveys, Vol.11, No.2, pp.121-137, June 1979.

[EHRIL7601] EHRICH,H.D. and LIPSKI,W.JR., "On the Storage Space Requirement of Consecutive Retrieval with Redundancy", Information Processing Letters, Vol.4, No.4, pp.101-104, Jan. 1976.

[HAERD7610] HAERDER,T., "An Implementation Technique for a Generalized Access Path Structure", IBM Res. Rep., RJ-1837, Oct. 1976.

[HAER7809] HAERDER,T., "Implementing a Generalized Access Path Structure for a Relational Database System", ACM Trans. on Database Systems, Vol.3, No.3, pp.285-298, Sept. 1978.

[KAMBK7911] KAMBAYASHI,Y., KONISHI,O., TANAKA.K., LE VIET,C. nd YAJIMA,S., "Relational model Based Research Information System ARIS", Yajima Lab. Res. Rep. ER79-06, nov. 1979.

[LEVIK7911] LE VIET,C., KAMBAYASHI,Y., TANAKA,K. and YAJIMA,S., "Use of Abstracted Characteristics in Relational Databases", COMPSAC'79, pp.404-414, Nov.1979.

[STON7509] STONEBRAKER,M., "A Comparison of the Use of Links and Secondary Indices in a Relational Database Systems", Memorandum No.ERL-M591, Univ. of California, Berkeley, Sept. 1975.

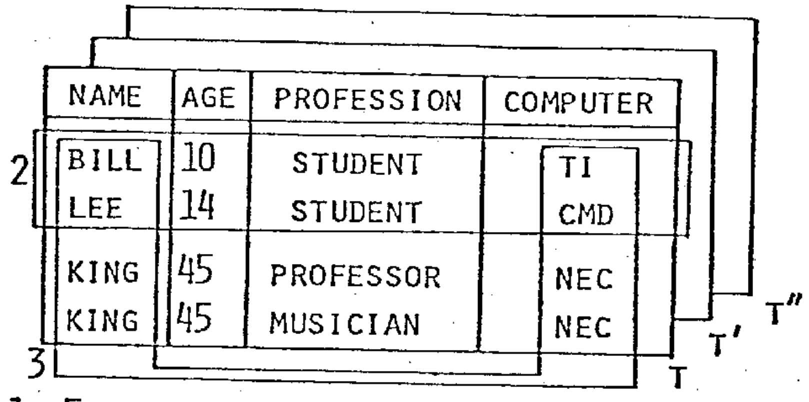
[TANAL7808] TANAKA,K.,LE VIET,C., KAMBAYASHI,Y. and YAJIMA,S., "A File Organization Suitable for Relational Database Operations", Prod. of the Int. Conf. on Math. Studies of Information Processing, pp.183-214, Aug. 1978, Lecture Notes in Computer Science, Vol. 75.

[TSIC7502] TSICHRITZIS,D., "A Network Framwork for Relational Implementation", Univ. of Toronto, Computer systems Research Group Rep. CSRG-51, Feb. 1975.

[TSICL77] TSICHRITZIS,D.C. and LOCHOVSKY,F.H., "Data Base Management Systems", Academic Press, 1977.

[YAJIK8001] YAJIMA,S., KAMBAYASHI,Y., KONISHI,O., TANAKA,K., LE VIET,C. and KATO,T., "Bibliographic Information processing Facilities for Relational Database system ARIS", Proc. of the 13th Hawaii International conference on System Science, Vol.II, pp.198-207, Jan. 1980.

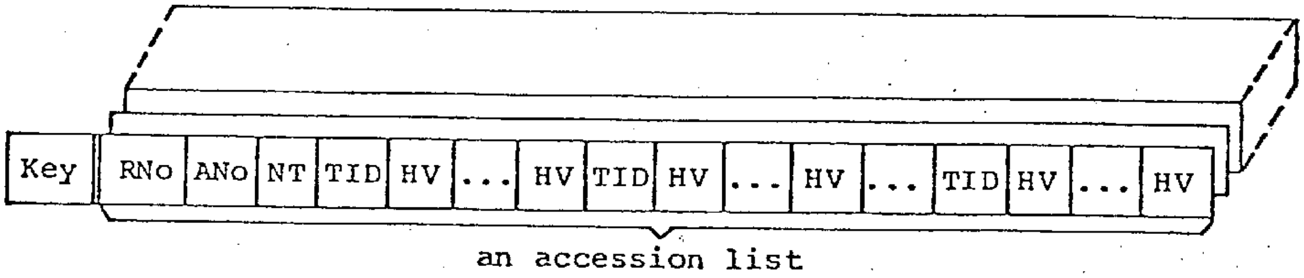
[YAO7906] YAO,S.B., "Optimization of Query Evaluation Algorithms", ACM TODS, Vol.4, No.4, pp.133-155, June 1979.



NAME	AGE	PROFESSION	COMPUTER
BILL	10	STUDENT	TI
LEE	14	STUDENT	CMD
KING	45	PROFESSOR	NEC
KING	45	MUSICIAN	NEC

1. FUNCTIONAL DEPENDENCY(FD): NAME $\longrightarrow$ AGE
2. LOCAL FD: NAME $\longrightarrow$ (AGE, PROFESSION)
   WHERE AGE < 15
3. GLOBAL TIME-DEPENDENT FD:
   NAME $\xrightarrow{T}$ COMPUTER, NAME $\not\xrightarrow{T', T''}$ COMPUTER

Fig.2.1 An example of functional dependencies



RNo	:	Relation number
ANo	:	Attribute number
HV	:	Hashed value
NT	:	Number of tuples

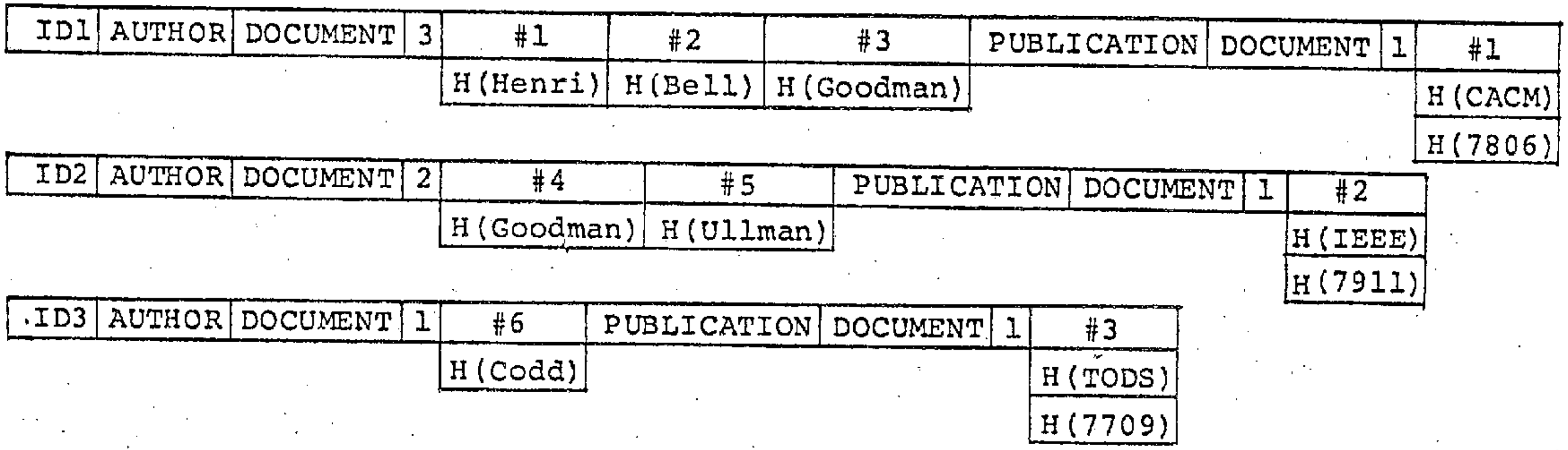
Fig.3.1 Structure of a RIS file record.

AUTHOR	DOCUMENT	NAME
#1	ID1	Henri
#2	ID1	Bell
#3	ID1	Goodman
#4	ID2	Goodman
#5	ID2	Ullman
#6	ID3	Codd

PUBLICATION	DOCUMENT	PUB_NAME	DATE
#1	ID1	CACM	7806
#2	ID2	IEEE	7911
#3	ID3	TODS	7709

Fig.3.2 Sample relations.

RIS1:



ID1 | AUTHOR | DOCUMENT | 3 | #1 | #2 | #3 | PUBLICATION | DOCUMENT | 1 | #1
H(Henri) | H(Bell) | H(Goodman) | H(CACM) | H(7806)

ID2 | AUTHOR | DOCUMENT | 2 | #4 | #5 | PUBLICATION | DOCUMENT | 1 | #2
H(Goodman) | H(Ullman) | H(IEEE) | H(7911)

ID3 | AUTHOR | DOCUMENT | 1 | #6 | PUBLICATION | DOCUMENT | 1 | #3
H(Codd) | H(TODS) | H(7709)

RIS_DIR

REL_NAME	ATT_NAME	RIS_NAME	RIS_ATT	MAX_VALUE	MIN_VALUE	ATT_CARDINALITY	MAX_BOUND	MIN_BOUND
- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -
AUTHOR	NAME	RIS1	DOCUMENT	Ullman	Bell	5	3	1
PUBLICATION	PUB_NAME	RIS1	DOCUMENT	TODS	CACM	3	1	1
PUBLICATION	DATE	RIS1	DOCUMENT	7911	7709	3	1	1
- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -	- - -

Fig. 3.3 A RIS file for attributes DOCUMENT's.

A	1	A	10	A	100
				AC	101
		AB	11	AB	110
				ABC	111
		B	01	B	010
				BC	011
				C	001
				AC	101
				ABC	111

Fig.3.4 Consecutive arrangement algorithm.

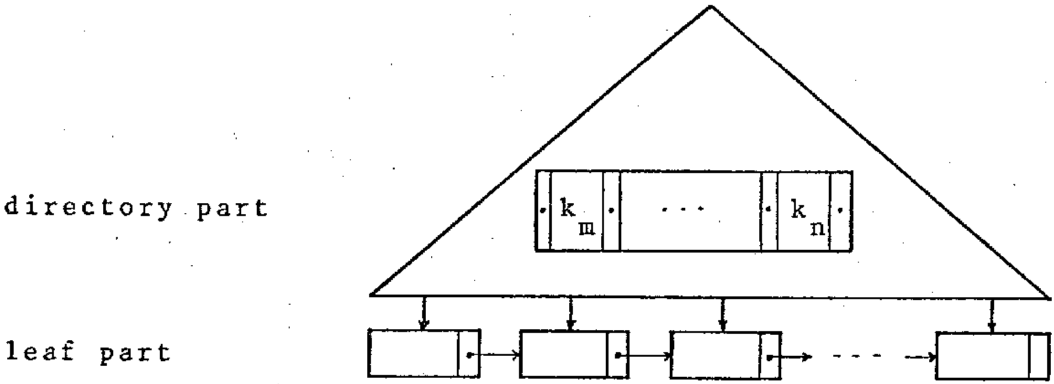


Fig.5.1 B^+tree data structure.

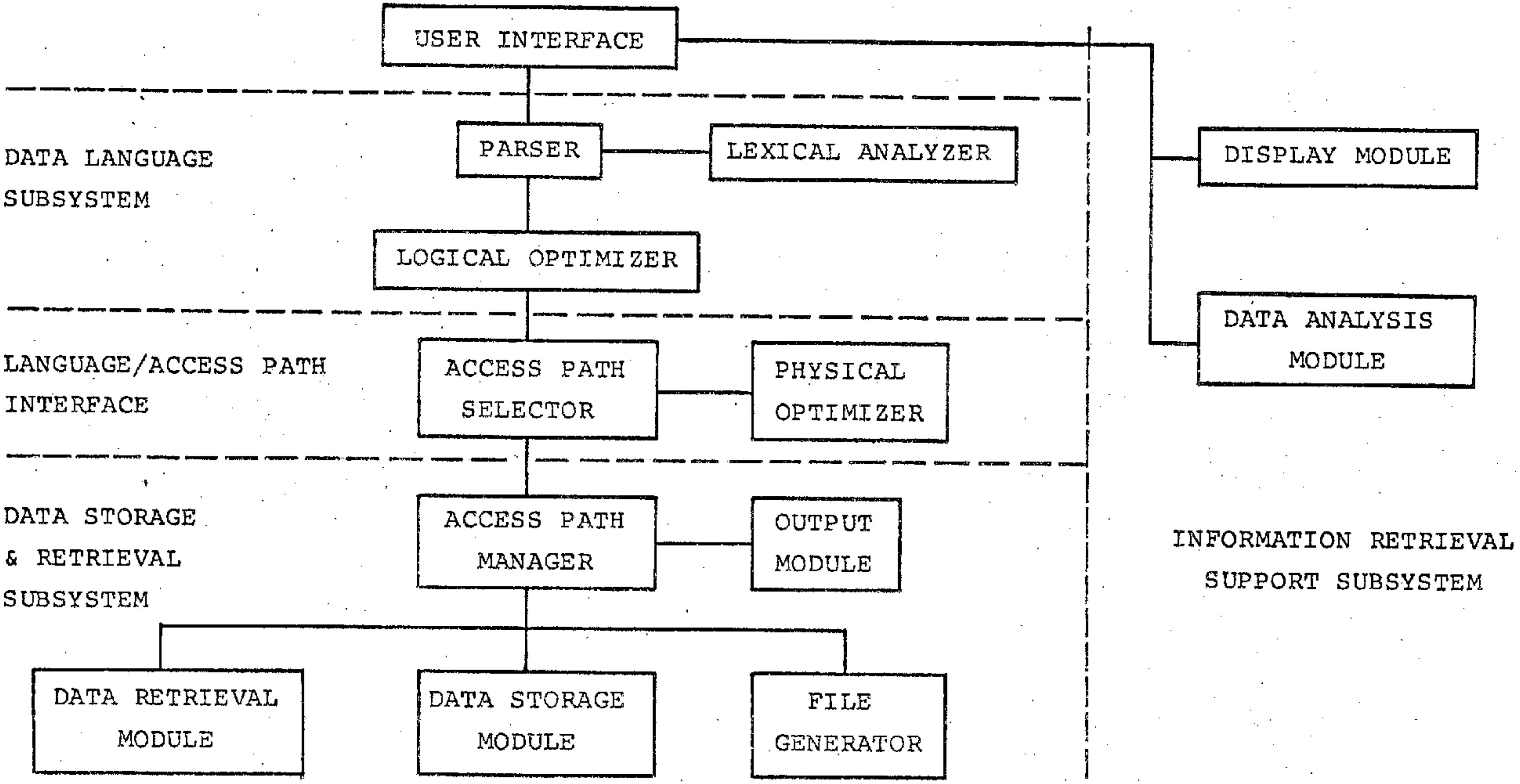


Fig.6.1 ARIS system configuration.

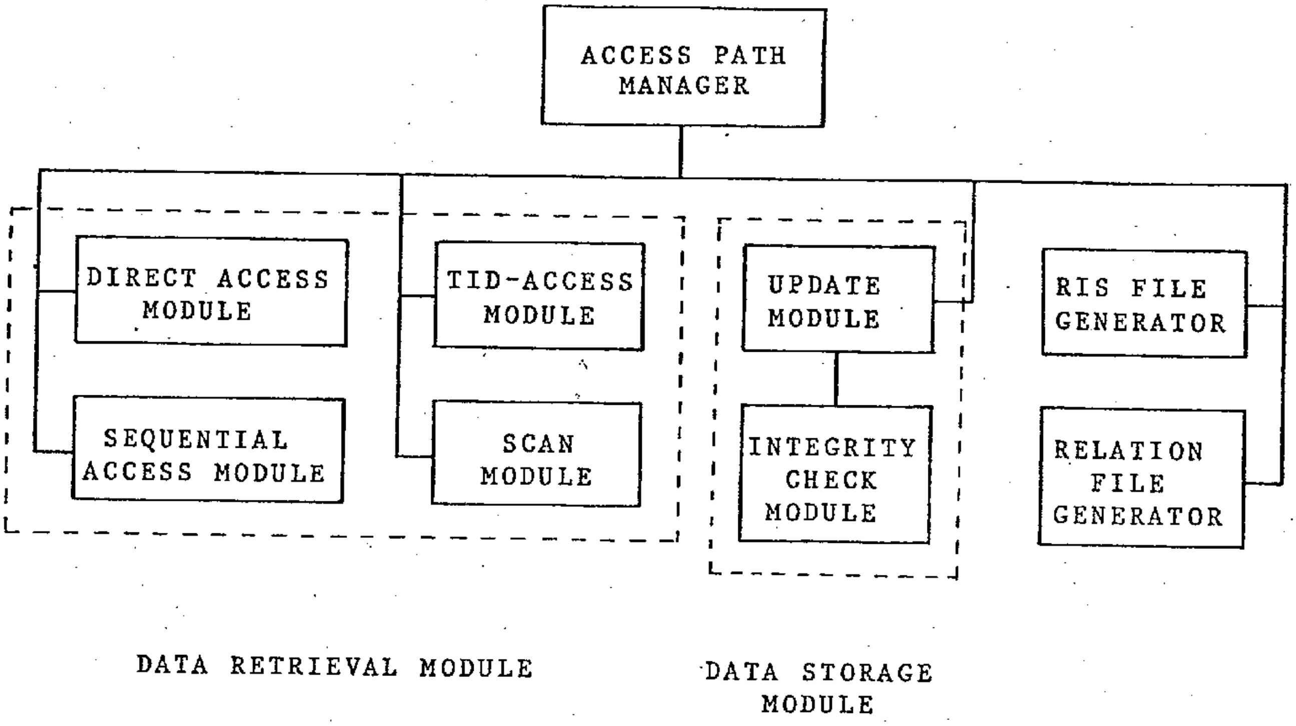


Fig.6.2 Configuration of data storage and retrieval subsystem.

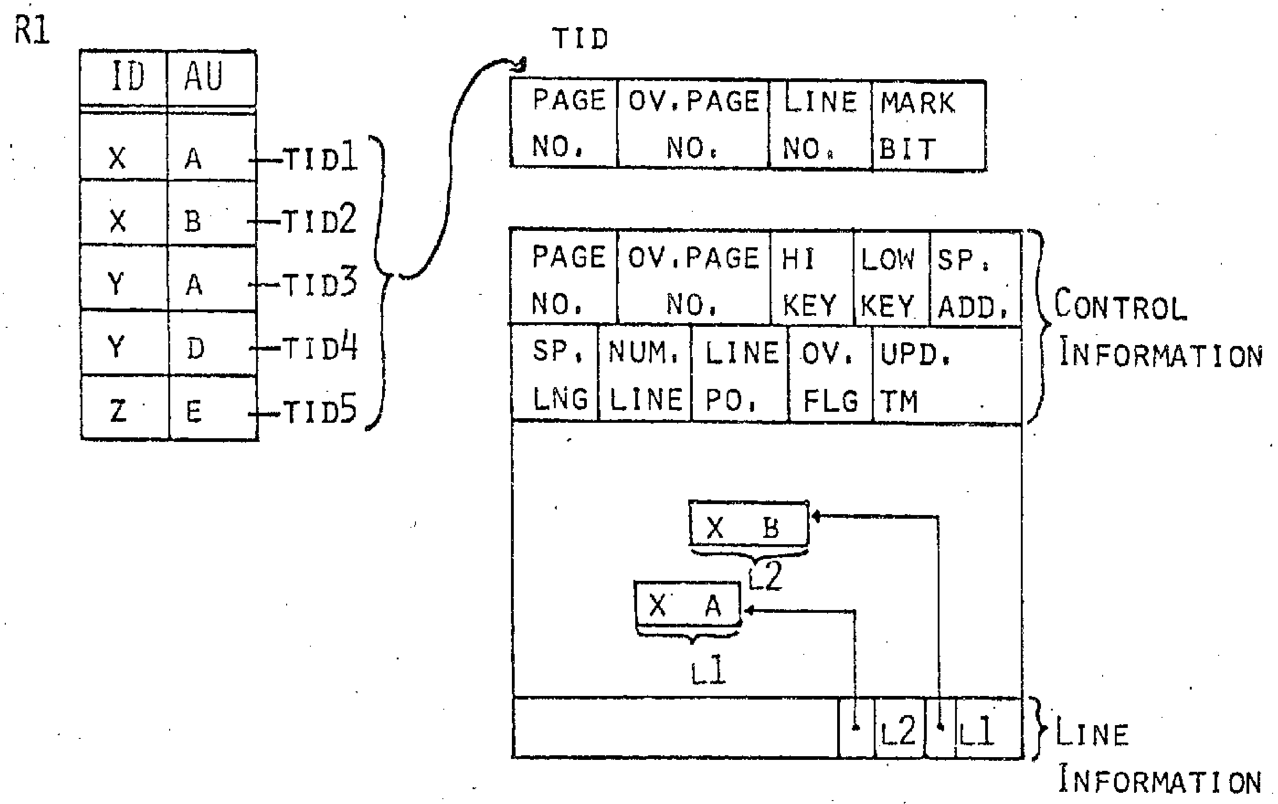


Fig.6.3 Page organization of relation files

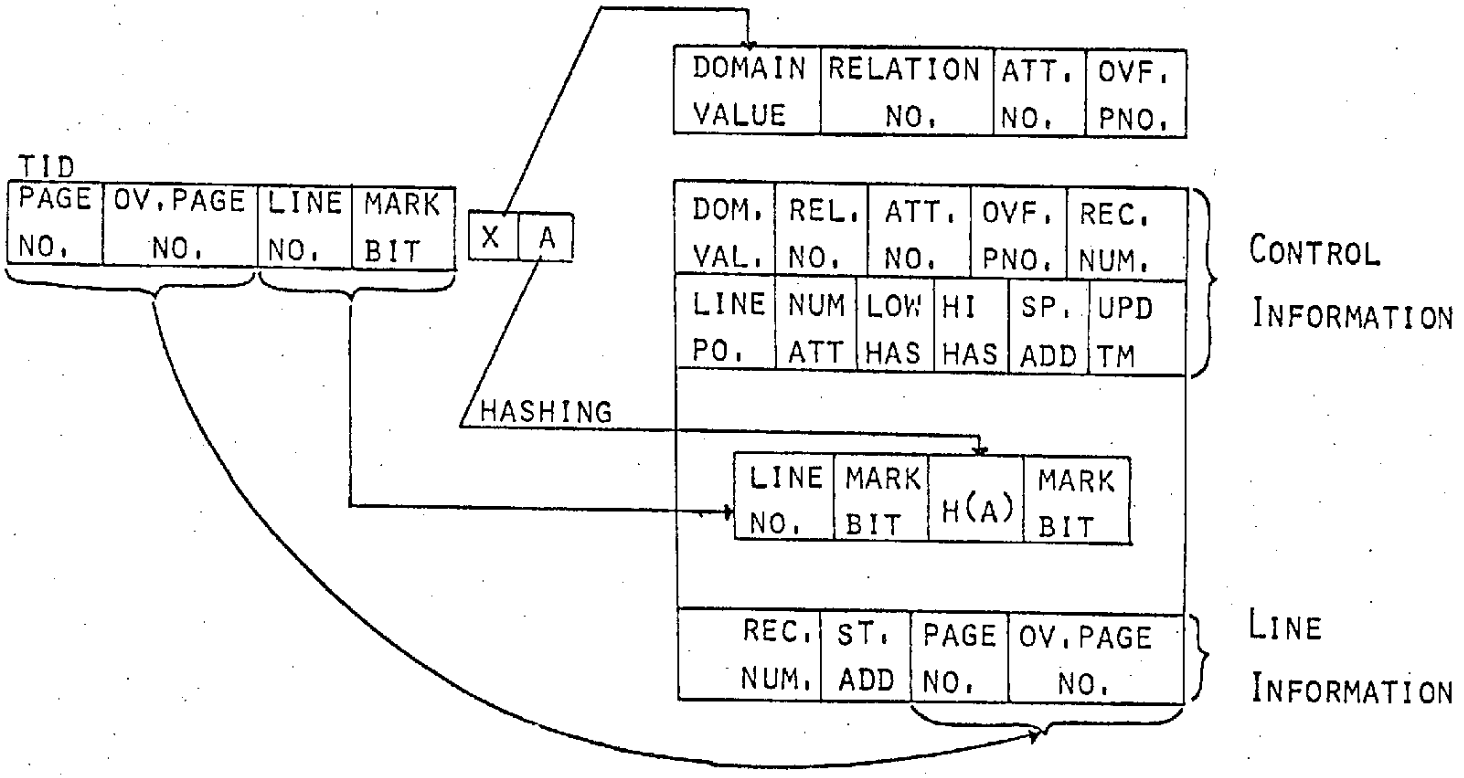


Fig.6.4 Page organization of RIS files.

```
DOCMNT (ID, DA, TI, PA)
AUTHOR (ID, AU)
CATEG  (ID,CA)
REFER  (ID, IDF)

RIS1  (DOCMNT.ID,  AUTHOR.ID,  CATEG.ID,
       REFER.ID,  REFER.IDF)
RIS2  (DOCMNT.PA)
```

Fig.6.5 A sample database and RIS files.

```
RANGE OF X IS DOCMNT
RANGE OF Y IS AUTHOR
RANGE OF V IS CATEG
RANGE OF W IS REFER
GET  (X.TI, Y.AU, V.CA)
WHERE  W.IDF = 'FAGI7906'
AND  X.ID = W.ID
AND  X.DA =  1979
AND  (X.PA = 'COMPSAC' OR X.PA = 'NCC')
AND  X.ID = Y.ID
AND  X.ID = V.ID
```

Fig.6.6 An example query.

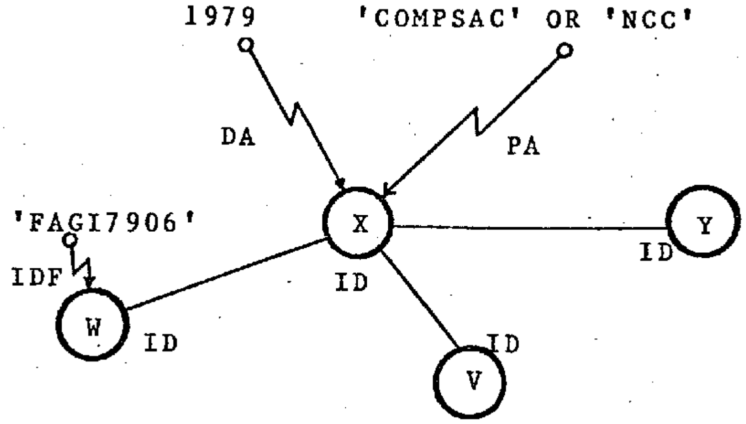


Fig.6.7 A graphic representation of the query in Fig.6.8.

```
     W.IDF = 'FAGI7906'
AND  X.ID = W.ID
AND  X.DA = 1979
AND (X.PA = 'COMPSAC' OR X.PA = 'NCC')
```

Fig.6.8 Subquery to be processed by DAM.

```
     X.DA = 1979
AND (X.PA = 'COMPSAC' OR X.PA = 'NCC')
AND  X.ID = Y.ID
AND  X.ID = V.ID
```

Fig.6.9 Subquery to be processed by SAM.

l_d : average length of a stored domain value.

l_T : length of a tuple identification code(TID).

l_h : length of a hashed value.

r_{hj} : the number of hashed attributes in relation R_j corresponding to a RIS file.

l_c : total length of a relation number, attribute number and tuple number indicator.

l_{nt} : length of tuple number indicator.

N_j : the number of tuples in relation R_j.

Table I: the list of parameters used in Section 5.2.

N_j : the number of tuples in relation R_j.

E_j : average number of tuples of relation R_j in a page

P_j : the number of pages of relation R_j, $P_j=N_j/E_j$.

F_j : the ratio of tuples which satisfy the selection operation to N_j.

J_j : the ratio of tuples which satisfy the join operation to N_j.

Table II: the list of parameters used in Section 5.3.

A REQUIREMENTS ENGINEERING APPROACH FOR DATABASE DESIGN

Kiyoshi AGUSA, Koichi TABATA,
Atsushi OHNISHI, and Yutaka OHNO
Kyoto University

Abstract

An approach to a support system for the DB design based on requirements engineering is presented. It includes three steps; to define the requirements of the target system, to derive the requirements conceptual schema and the requirements external schemata and to calculate an actual weight of an access path to optimize the DB system.

The primary design example derived automatically will help the logical DB designers' activities such as view modeling and view integrations. The designer can utilize the access frequencies for view restrcturing.

1. INTRODUCTION

Today, it is recognized that all stages of the life cycle of the information system including the requirements describing stage should be properly maintained to get the reliable system. If the system now under designing is used by only designers and implementers, the developed system will work as intended. In usual cases, however, its user is neither the designer nor a specialist about the information system. A close communication between a system designer and users is required to realize such a system. Almost all DB systems belong to the class of such systems.

It is easier for non-specialists to express their intension about a target system with their mother languages, i.e., natural languages than with a special language such as a programming language. Daily conversations often include ambiguous meanings which we can suppose with its context. Since this ambiguousness should be deleted for the system specifications, we use an artificial language whose meanings are well defined.

Several languages for requirements descriptions are proposed. For instance, PSL(Problem Statement Language) of Michigan University[ISDOS(1975),Teichroew(1977)], RSL(Requirements Statement Language) of TRW [Bell(1976)], BDL (Business Definition Language) of IBM[Goldberg(1975)] etc. are known well. Moreover, there are several papers about the formal specification for

Database systems[Smith(1977), Lockemann(1979)]. However, a few papers are presented about the Database System Description using a Requirements Engineering approach[Kahn(1976)].

In this paper, we propose how to describe the requirements of the DB systems and how to utilize them to design the optimal systems. DB design process can be devided into the following phases.

1. information requirements specification and analysis
2. logical DB design
3. evaluation of logical DB design
4. physical DB design
5. evaluation of physical DB design
6. DB construction and initialization
7. perfomance evaluation

Between any pair of adjacent phases, there is a gap of abstraction level. It is difficult to bridge over the gap in an automatic way with a computer. It is a designer's job to read and understand the descriptions given as an output of a previous steps, and to map its meaning onto the description of its abstraction level. However, when the requirements definer describes the requirements of the target DB system with an intention to be recognized by a computer, some primary design will be derived automatically. Now, let us investigate the logical DB design phase more in detail. It may be devided into the following steps.

2-1. view modeling

2-2. view investigation

2-3. view restructuring

2-4. schema analysis and mapping

These steps except the last one are the objects of an automatic design. The design results may not be final ones and be utilized as design aids for a logical DB designer. That is, he reads the requirements description, refers the primary design provided as a result of its analysis by a computer, and designs the logical DB of a target system by himself.

2. CORRESPONDENCE OF REQUIREMENTS TO DBMS LAYER

We follow the approach of the ANSI/SPARC framework[Jardine (1977)]. The DBMS architecture is partly based on the concept of "nested machine". Its model consists of three layers, i.e. external, internal and conceptual schema.

To define DB systems, their two aspects should be described, that is, data structures and data utilizations. We call the former aspect static requirements and the latter dynamic requirements. The requirements descriptions which represent a direct interface between the system and users are treated as an external schema. There may be many external schemata corresponding with DB applications. Each application has its own view of the DB corresponding with static requirements and procedures defined as dynamic requirements. The requirements description of DBMS should clarify what kinds of data are stored, retrieved and updated, how often such processes are activated, and how the data in the DB relate to each other. Some relations among the data are so inherent that they may be regarded as a fundamental common sense in the DB system. Then such relations derived from the requirements descriptions will be regarded as a conceptual schema, more precisely said, a requirements conceptual schema of data structures.

For designing the DB systems, the system parameters such that process operations, their frequencies and the cardinalities of relations should be also defined to determine how often a certain access path is used. The requirements about typical processes can indicate a dynamic mapping ratio between two relations. The dynamic mapping ratio means the ratio of the cardinality of inputs of the process to the cardinality of outputs. It depends on the characteristics of the preceeding processes, the function of the process, and the database. Since such a ratio varies from process to process, we can define several data structures and data accessing methods which are suitable for each process. These data structures can be corresponded to an external schema. Then we call it a requirement external schema.

With regard to the principle that we should leave as much room as possible for designers' choice, the user needs not to describe about an internal schema. Since an internal schema has close relation not only to the conceptual schema but also to the system configuration, it seems reasonable that the internal schema is left undefined until the latter phase of the system development.

Thus, in our approach, the first phase of DB design process mentioned above can be devided into the followings.

1-1. requirements analysis
1-2. constraction of requirements external schemata
1-3. extraction of a requirements conceptual schema
1-4. analysis of dynamic requirements conceptual schemata

1-2,1-3 correspond to 2-1,2-2 respectively. That is, constraction of requirements external schemata corresponds to view modeling in the logical DB design phase and extraction of a requirements conceptual schema to view integration.

3. REQUIREMENTS DESCRIPTION OF DB SYSTEM

In this chapter, we present four steps in the requirements specification phase of DB design process.

3.1 REQUIREMENTS ANALYSIS

In general, prior to requirements description, requirements definer should comprehend the behavior and function of a target system well. Then he should refine the system requirements stepwisely through interviews and review it.

In DB system requirements, the definer should pay attention to two aspects, that is, data structure and data utilization. These aspects are indispensable with DB design.

3.2 REQUIREMENTS EXTERNAL SCHEMA AND REQUIREMENTS DESCRIPTION LANGUAGE

3.2.1 REQUIREMENTS DESCRIPTION FOR EXTERNAL SCHEMA

The requirements description can be regarded as an external schema since it defines the interface between a real world and the DB system. It expresses a static data structure and a dynamic one. The requirements external schema corresponds to view model in the logical DB design.

The static data structure are defined with the descriptions of the relation between two entities and the inclusive relation between the entity and the elements. We call it a static requirements.

We may classify data structure defined by static requirements into three types; i.e. (1)network (2)hierachy and (3)relational structure. These types are well known as typical data structures in DB system. However, it does not mean that the requirements definer should declare what type of them is used in his description. He can define his data structure model as a network one in another part. The system designer or implementer is held responsible for what type of a data structure is adopted in the target system. In the requirements description, we also define the processes which shows how the data in DB are used through the user's view, how frequently it will occur and the the effective cardinality of the entity determined dynamically and so on. We call it a dynamic requirements.

It is impossible that the requirements definer describes all procedures accessing to the DB which will appear in the real operations. Then, he will write only the typical or most frequently used procedures. It is true that there may be sometimes the procedure which is frequently used and he passes over. But it has no problem if it accesses to the DB legally.

Generally speaking, there is no existency without the perception in the requirements description. The designer will make use of this static and dynamic requirements to get the suitable conceptual schema and its internal schema. He can add some function and schema which, he thinks, will be needed in future or in the extended use. It can help to recover some portion of the requirements definers' overlooks.

3.2.2 REQUIREMENTS DESCRIPTION LANGUAGE FOR DB SYSTEM

We use PSL as a DB requirements description language, since it has an enough capability from the syntactic view. We add some semantics to it to extract a primary design of the target DB system. The addition of semantics means the restrictive use of some reserved terms such as HAPPENS, CARDINALITY, RELATION etc.

PSL is one of the languages which support to maintain the documentation describing the target system, the environments where it operates, and the project organization. One of the most important features of PSL is a computer-aided management of the documentations. PSL will give full play when combined with its analyzer PSA (Problem Statement Analyzer).

PSL is invented as a definition language of the requirements of arbitrary kinds of information systems. The concept appeared in the target system may vary from each other. To accept all varieties of the system concepts, PSL has a simple and extensible language structure. Sometimes it is too simple to analyze its semantics in detail. The meaning of the descriptions is left undefined until the common concepts or mutual agreements are founded between the definer and the reader. For example, the inputs of a process are defined with USES statement in PSL. We can define the data and/or the control as its inputs. It is very difficult to distinguish them without other information, that is the meaning of the system and the terms, the environments, the definer's intensions, etc. However, with a careful investigation of PSL, it comes to light that almost all concepts appearing in DB system requirements can be defined using PSL syntactically.

We add some functions which derive the useful parameters for the DB system design to PSA. To utilize these additional functions, the requirements definer should be careful to use some special reserved words of PSL. With PSL, we can express the requirements of any kind of information system. The flexibility of requirements lies in the fact that definer can use any label and can assign meanings to reserved terms freely etc. For example, he can label an object 'man', 'teacher', 'professor'..etc. and can assign the section INPUT to control information in one case, to processing data in another case.

The lack of common concepts among definers will cause misunderstanding. In a particular application, even if the interpretation of some terms is fixed, the flexibility of requirements description language will not decrease. The definers will not feel inconvinient with its restriction.

We show the correspondence of PSL sections and statements to the terms in DB requirements.

(a) SET: A SET means one DB. That is a collection of ENTITYs. Where ENTITYs may be thought as logical records, a SET may be thought as a logical file.

(b) ENTITY: An ENTITY is a record type of CODASYL or a table of a relational DB.

(c) ELEMENT: An attribute of a record are defined as an ELEMENT of PSL. Notice that an ATTRIBUTE statement of an ELEMENT section shows a domain of the attribute of DB. That is, there is no description of each occurence in DB.

(d) RELATION: A RELATION defines a relationship between two ENTITYs. This corresponds to a set, i.e., the owner-member relationships of CODASYL. This section defines the type of DB. That is, if there is no RELATION description, it means a relational DB. If there is, a hierarchical DB or a network type depending on the existence bi-directional relation. A CONNECTIVITY statements is used to specify an average ratio of occurences of two ENTITYs.

(e) USES/DERIVES: In PSL, a PROCESS USES some data and DERIVES another data. If such data is an ELEMENT or an ENTITY defined as an attribute or a table of a DB, we assume that this PROCESS will access this DB. Moreover the access is assumed to be executed through the path defined in RELATION sections. It may happen that a designer need to describe more than one RELATION to a particular pair of ENTITYs depending on element relations. If there is more than one path, we can define explicitly one of them using MAINTAIN statements.

(f) others: The frequency of the activation of a process is defined with HAPPENS statement. The cardinality of ENTITYs or ELEMENTs is defined CARDINALITY statements.

3.3 REQUIREMENTS DESCRIPTION FOR CONCEPTUAL SCHEMA

The conceptual schema is a fundamental relation of entities and doesn't vary with each application view. The conceptual schema corresponds to integrated view in the logical DB design. The example of these requirements conceptual schema is the relation between an entity and its inherent attributes. A man has his name, address, years, height,...etc. If we deal with individual people, we can not say about a man without his name and address. It doesn't depend on its application.

The requirements definer does not need to state the conceptual schema. It may be specified by the designer or implementer. However, it is better for a supporting system to provide the function to derive the candidate of the conceptual schema with a computer, if possible. We show the correspondence of the ANSI/SPARC DB layer with the requirements layer in Fig. 1.

It is a very difficult job for a computer to decide what is a fundamental relations in the requirements descriptions. The trivial method to derive a conceptual schema is to make a union of all external schemata defined as the static requirements.

3.3.1 REQUIREMENTS SCHEMA CONSTRUCTION

The requirements descriptions written in PSL may be used to support the whole development of the target system. However, we will treat only the part of it to get the DB schema of the requirements phase.

The requirements descriptions of DB systems include the data descriptions and the process descriptions. If the designer or implementer of the DB systems defines the requirements, they shall be regarded as an conceptual schema of DB. On the other hand, if the application system designer or the user, their descriptions will be external schemata. Here, we consider the latter case, that is, in the case that the requirements is defined by users.

Each description defines the data structure derived from the static requirements. The weights of its structure are attained by the calculation of the utilizing frequency of the data.

3.3.2 INTEGRATION OF DATA

To design a DB system, we need a certain conceptual schema which represents a common concept in user's view. The requirements in PSL is easy to read and understand because of its formatted outputs. However, some terms in one description may be used to express different meanings in other one. In other case, the different terms are used for the same meaning. It is very difficult to solve these problems with a computer. Some conversation between the DB design support system and the designer is required to combine the descriptions written by more than one requirements designer.

There is no section to combine the descriptions of different DBs of requirements description into one in PSL. However, the COMBINE command is provided in PSA to correct errors in the descriptions of a DB. For example, a requirements definer remembers a spell of some word incorrectly and has written his descriptions with the erroneous spell. To correct all errors is tedious job. If he can find the use of PROF and PROFESSOR as as a same meaning and wants to merge them, then he will send a COMBINE command.

A result of the execution of this command is shown in Fig. 2-a. It is true that it is not a recommanded way to use a COMBINE command, but a DB designer can use it to derive a conceptual schema. If some data or its structure is seemed to be natural, common or inherent, he can describe his own ENTITY or RELATIONs between them and combine such definers' data into his data. If needed, he should combine also the ELEMENTs which appear in CONSISTS statement in the ENTITY section.

On the other hand, the same name may be used in difficult meaning by some requirements definers. The logical DB designer should make a clear distinction among them. For instance, an element of the ENTITY "INDEX" represents department name in one case and human name in another case. Then he should modify them to be able to distinguish as shown in Fig. 2-b.

The requirements conceptual schema is defined as a union of the external schemata combined by the DB designer.

3.4 ANALYSIS OF DINAMIC REQUIREMENTS CONCEPTUAL SCHEMA

Here, we will consider the dynamic behaviour of the DB system written as the dynamic requirements. We should utilize the information defined in the requirements description as much as possible to get a suitable system. Then we caluclate the usage frequencies of the relations, and attach them to the union as their weights. A static cardinality is defined with CONNECTIVITY statement in RELATION section. Now, we consider the dynamic cardinality and relation connectivity specified in PROCESS section. It is utilized to calculate access frequency.

When a process P_i recieves an input I_i whose cardinality is C_i, uses it as an access key of a DB and derive an output O_i, what is the cardinality of O_i? The cardinality of O_i may be defined in the description of O_i. In this case, however, the cardinality of O_i depends on the number of the occurrences which are retrieved from the DB with the corresponding key. Of course, it depends on also the task of the process. It is very difficult to calculate its dynamic cardinality using the task of the process without understanding the meaning of the description. In the RELATION section, the definer can define its connectivity N in the CONNECTIVITY statements. A CONNECTIVITY statement specifies the number of ENTITY occurrence of the first argument ENTITY in the BETWEEN statement that are related to a number of ENTITY occurrence of the second argument ENTITY. If a particular ENTITY occurrence may be related to only one other ENTITY occurrence, the cardinality is one to one. If a particular ENTITY occurrence may be related to more than one ENTITY occurrence, the cardinality is one to many. We use the ratio of the cardinality of two ENTITIES for the CONNECTIVITY if there is no description of the RELATION like in the case of a relational DB. If there is more than one RELATION, we can chose one of them using MAINTAIN statements. This facility can help to get more precise value when the definer assigns the individual value to the different process.

We use this value to calculate the dynamic cardinality, e.g., $C_i \times N$ in the above case. This is illustlated in Fig.3. If there is a feedback loop, we can assume that the input cardinality is $C/(1-\rho)$ where C means the input cardinality defined as the static requirements, and ρ means the probability of the feedback from the output to the input. If there is more than one DB access in a PROCESS description, the dynamic cardinality of the output can not be estimated so easily. For the simplification, however, we define it with the maximum number among them. These values are regarded as the weights of the relations between the data. It is easy to get these values in a computer-aided way. The designer can decide which relation should be maintained by the inverted files with them. Moreover, he can add the shortcut access path from an ENTITY to another if all of calculated weights between them are significantly large.

The dynamic aspect of the conceptual schema is obtained in the same way. That is, the designer combines several PROCESSes into one, if needed. Next, the activation frequencies of the application programs are multiplied to the weights. Then we sum up all of them. It is regarded as the weight of the relation in the requirements conceptual schema.

Consequently, we can show the equation of the weight W of the relation as follows;

$$W=\sum_{i} P_{ix}\max(C_{ij}\prod_{k} n_{ik})$$

$i \in$ all requirements external schemata

P_{ix}: the frequency of the activation of the process i corresponding one of the requirements external schema.

C_{ij}: the cardinality of j which is an input of the process i.

n_{kj}: the connectivity of the relation. k is defined with the path along which the global inputs reach to its subprocess.

4. EXAMPLE

Let us consider a university DB system which has informations about professors, students and medical checks.

The requirements definer analizes the perfomance, functions size, etc required by users. There may be more than one application system utilizing the university DB. To define the requirements of the application system, the sequence of processes which retrieve and update the DB is described. In this example, we define the external requirement schema with data structure aspects in PSL and dynamic behaviors of the system appearing in typical routines. One of typical routines is the calculation of correlation between health condition (e.g. sight, intelligence quotient) and score of students. Another routine is to find the student's mark of the subject which a particular professor teaches under the condition that the student belongs to the laboratry of the professor.

Several numbers of typical processes using their data structure are described in PSL. The system definer will combine some data into one. For instance, the ELEMENT "name" of the ENTITY " professor " is seemed identical to the ELEMENT " prof-name " of the ENTITY " subject " in the student DB. Then, the command

```
COMBINE        CNAME=PROF-NAME      DNAME=NAME
```

will be issued. One of the resulting external requirements schemata is shown in Fig.4. This can be regarded as a primary view model. The requirements definer of each application system presents his own view of the DB system with any combination of a relational, hierarchy and network model.

In general, each requirements definer has his own requirements DB. If so, we need a facility to merge several requirements DBs into one. When DBs are merged, the system designer shall specify which name in one requirements DB corresponds with a particular name. This procedure is not so easy that he should read through all requirements DBs and take into consideration what the requirements definer intends. The integrated view shown in Fig.4 corresponds to a conceptual requirements schema.

In Fig.4, we show the integrated processes also. To merge processes, we need careful check of what to be intended. It is a difficult job for a computer. Then we extract the conceptual schema with union operation of all of external requirements schemata. This process is regarded as view integration in logical DB design.

Next, we calculate the access frequencies to help the DB designer to recognize the most important the access path. Fig.5

shows the calculation result. The access path along bold lines is used so frequently that the direct access path from P-RET-SUBJ to P-RET-STU may be recommended to be added as shown with a dotted line. Note that it is not a computer but a designer that decides finaly whether the view model is modified or not.

We regard this process as view restructuring. Of course, there may be many other factors to be considered such as security, future extension etc., to restructure the view models.

The designer combines some processes into one and sends a command to calculate all weights of the relations. He will decide the conceptual schema using this requirements conceptual schema and the careful investigation of the formatted output of PSA. If needed, he will add several relations which are not appeared in the requirements descriptions.

5. CONCLUSION

The development of structured or automated DB design methodology is one of the main subjects in the DB research fields. The formal description of the requirements of the target DB system is a first step of it. It is said that the requirements description should have the property called design free, that the requirements description does not impose the particular design selection. However, the approach described in this paper is introduced to propose the particular design selection for an automated DB design. Of course, some parts of design selection are left to be decided by the designer, e.g., what should be seemed as a conceptual schema, what is most important in the DB application, security, privacy or others, etc.

The DB designer should check all external schemata described in the requirements with PSA commands. Since the command to get the information along the particular pathes not provided, he must calculated the access frequency with PSA reports. We have proposed ASA (Assertion Statement Analyzer) to verify the requirements descriptions described in PSL.[AGUSA(1979)] It has the base on the path analysis of the system structure or the data derivation of the system aspects. We will extend our ASA to enable to output some reports about the dynamic data utilization. Moreover, a facility to help deriving a conceptual schema from requirements descriptions is now under the implementation.

REFERENCE

ISDOS Project.(1975). " Problem Statement Language Version 3.0 Language Reference Manual ". ISDOS Working Paper, No.68. May 1975.

Teichreow D. and Hershey E.A. (1977). PSL/PSA:A Computer-aided Technique for Structured Documentation and Analysis of Information Processing System. IEEE Trans. on Software Engineering, Vol.SE-3, No.1. January 1977, pp 41-48.

Bell T.E. and Bixler D.C. (1976). A Flow-oriented Requirements Language. Proc. of MRI Symposium on Computer Software Engineering, April 1976, pp 109-121.

Goldberg P.C. (1975). Structured Programming for Non-Programmer. IBM Report RC-5318. March 1975.

Smith J.M. and Smith D.C. (1977). Database Abstraction:Aggregation and Generalization. ACM Trans. on Database Systems, Vol.2, No.2, June 1977, pp 105-133.

Lockemann P.C., et al. (1979). Data Abstraction for Database Systems. ACM Trans. on Database Systems, Vol.4, No.1, March 1979, pp 60-75.

Kahn B.K. (1976). A Method for Describing Information Required by the Database Design Process. ACM SIGMOD 1976. pp 53-64.

Jardine D.A. (1977) " The ANSI/SPARC DBMS Model ". North-Holland Publishing Company. 1977.

Agusa K., et al. (1979). Verification of Requirements Descriptions. Proc. of 12th HICSS, vol.1, January 1979, pp131-139.

Yao S.B., et al. (1979). The Functional Dependency Model for Database Design. Fifth International Conference on Very Large Data Bases, October 1979.

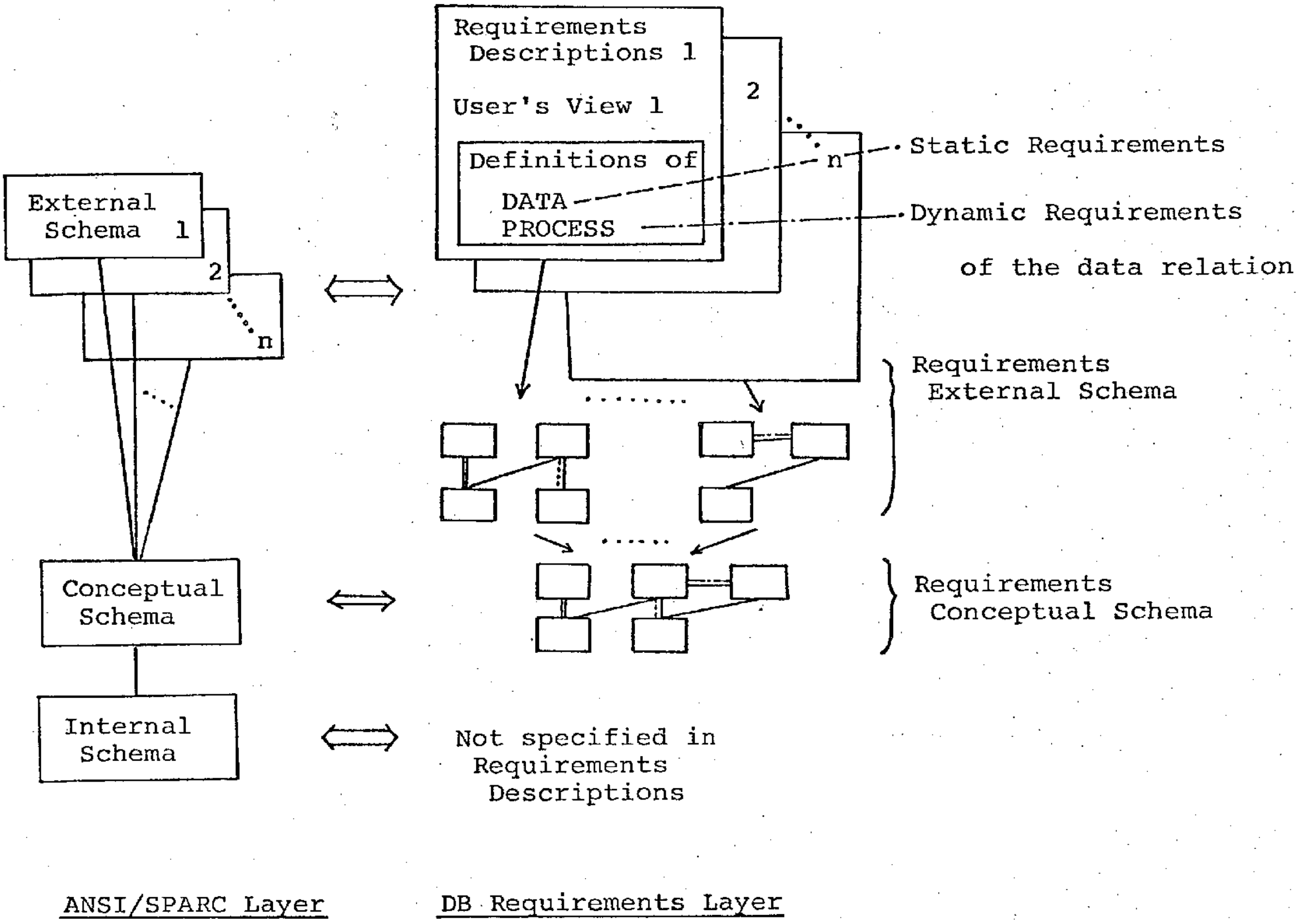


Fig.1 The Correspondce of the ANSI/SPARC Layer with the Requirements DB Layer.

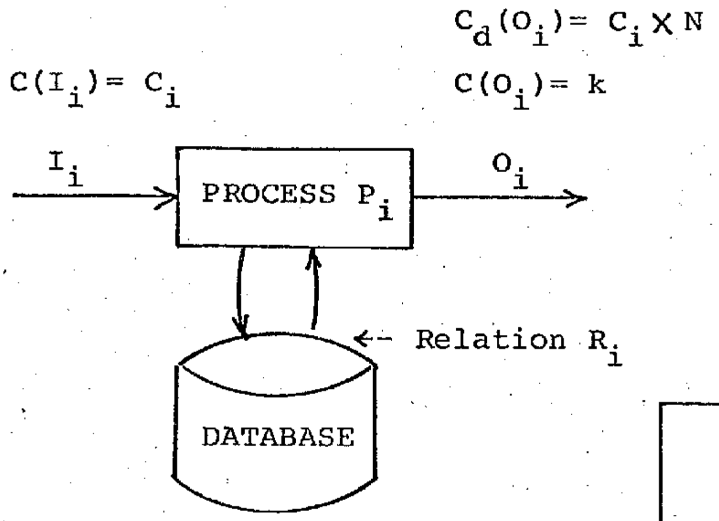
```
 CSTS  PROF-NO,
       NAME,
       CLUB;
      .
      .
      .                  ENT    PROF;
ENT    PROF;      ⟹      CSTS  PROF-NO,
 CSTS  PROF-NO,                 NAME,
       PROF-NAME,               CLUB,
       SUBJECT,                 SUBJECT,
       LAB-NAME;                LAB-NAME;
```

Fig.2-a. Synonym Integration

```
ENT    INDEX;                          ENT    INDEX-DEPT;
 CSTS  INFORMATION-SCIENCE,             CSTS  INFORMATION-SCIENCE,
       MATHEMATICS,                           MATHEMATICS,
       ELECTRONICS;                           ELECTRONICS;
         .                                      .
         .                  ⟹                   .
         .                                      .
ENT    INDEX;                          ENT    INDEX-NAME;
 CSTS  AGUSA,                           CSTS  AGUSA,
       TABATA,                                TABATA,
       OHNISHI,                               OHNISHI,
       OHNO;                                  OHNO;
```

Fig.2-b. Homonym Resolving

Fig.2 Example of Synonym/Homonym Processing



C(a): the cardinality of a.

C_d(a):the dynamic cardinality of a.

```
PROC P_i;
 USES  I_i;
 DRVS  O_i;
REL  R_i;
 BTWN I_i AND O_i;
 CONN IS 1 TO N;
ENT  I_i;
 CARD C_i;
```

Requirements Description

Fig. 3 The Calculation of the Dynamic Cardinality.

PROFESSOR DATABASE

professor(prof-no.,name,club,subject,
lab-name)

laboratory(lab-name,establishment-year)

STUDENT DATABASE

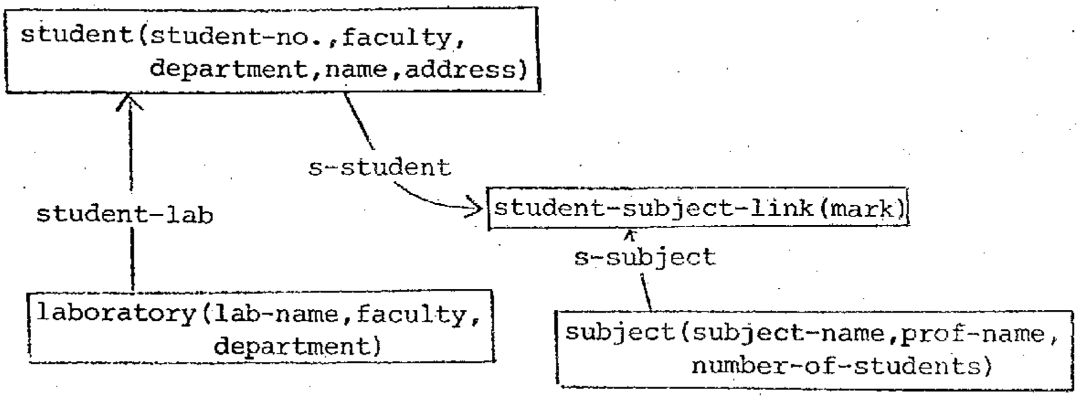

student(student-no.,faculty,
department,name,address)
s-student
student-lab
student-subject-link(mark)
s-subject
laboratory(lab-name,faculty,
department)
subject(subject-name,prof-name,
number-of-students)

Fig. 4-a Static Requirements

```
SET      PROF-DB;
 CSTS    PROF,LAB;

ENT      PROF;
 CSTS    PROF-NO,
         NAME,
         CLUB,
         SUBJ,
         LAB-NAME;

ELE      PROF-NO;

SET      STU-DB;

ENT      STUDENT;
 CSTS    STU-NO,
         FACULTY,
         DEP,
         ST-NAME,
         ADDR;

REL      STUDENT-LAB;
 BTWN    LAB
 AND     STUDENT;
 CONN    N2;

PROC     P-RET-LAB;
 USES    LAB-NAME;
 DRVS    STUDENT-NAME
 USING   LAB;
 HAP     P1 TIMP ONE-
         TIME-UNIT;
 MTNS    STUDENT-LAB;

INP      INPUT-DATA;
 CSTS    PROF-NAME;
 HAP     C1 TIMP ONE-
         TIME-UNIT;
```

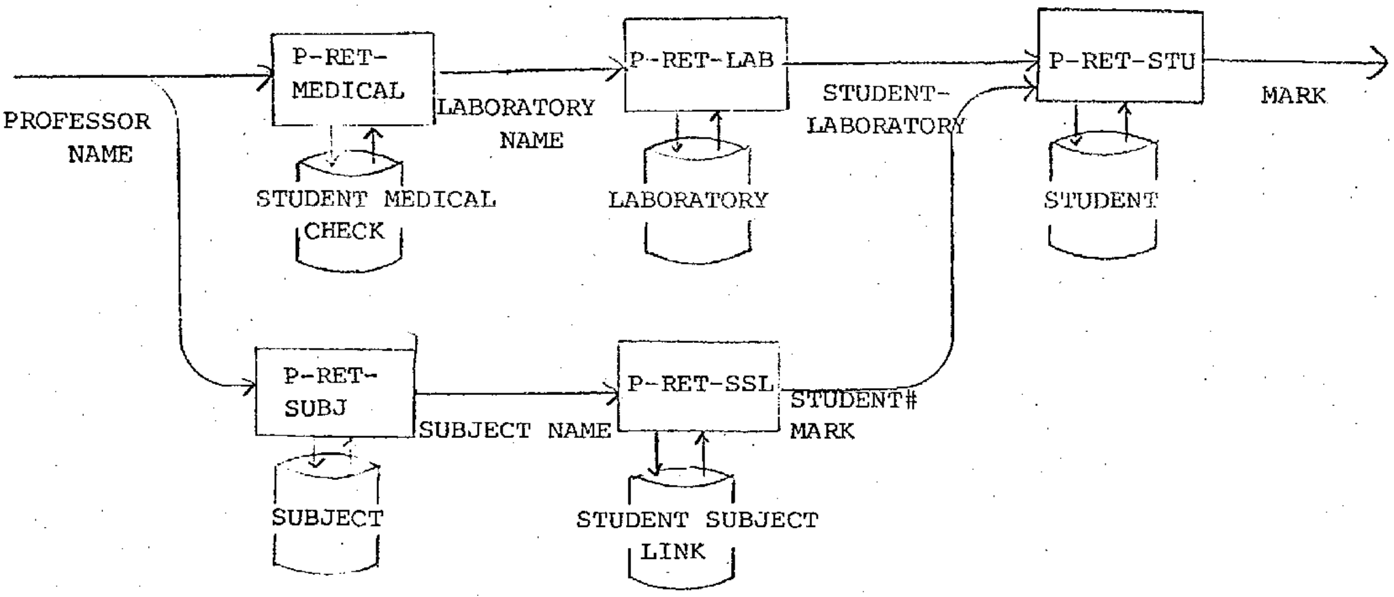
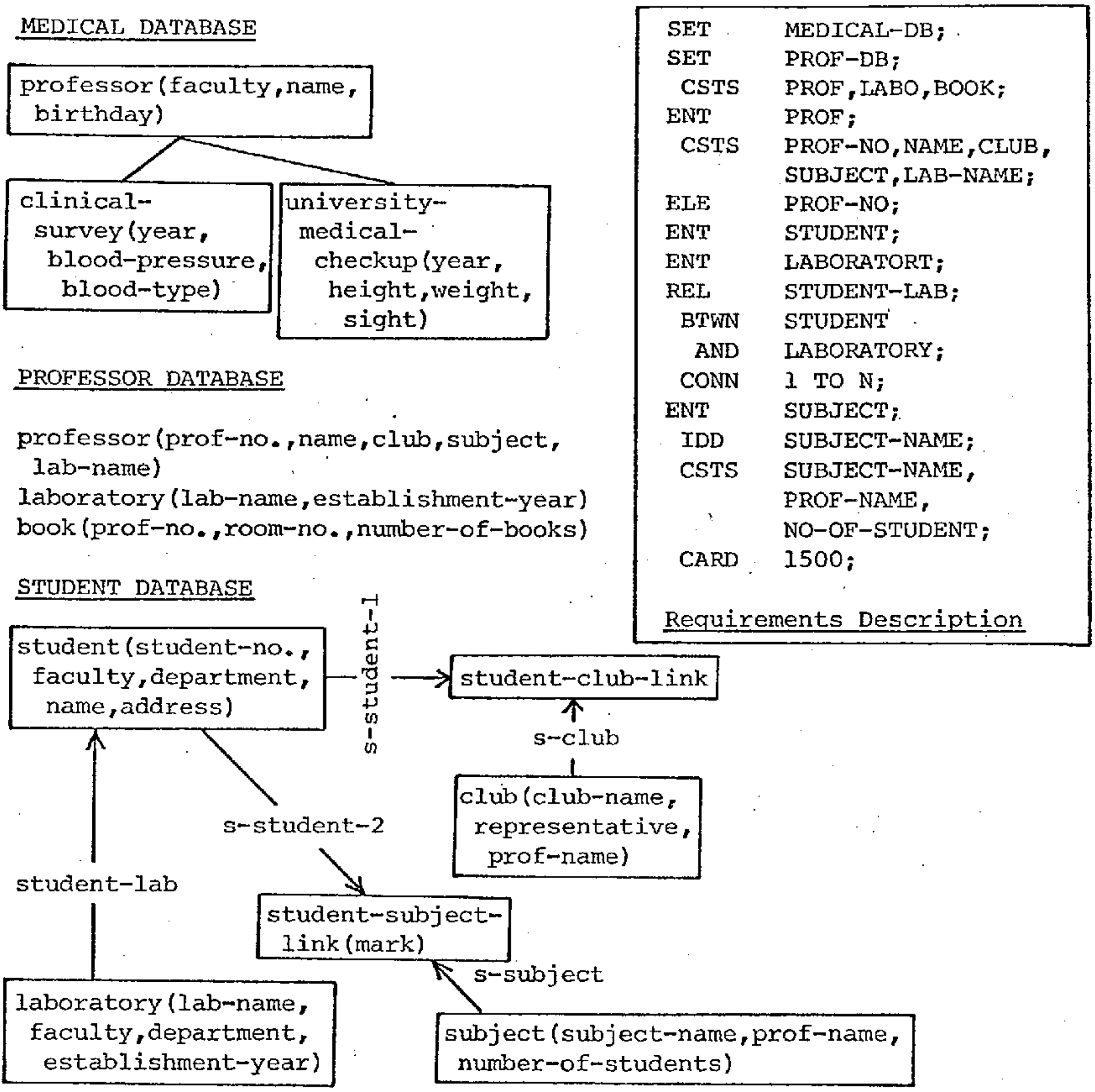


Fig. 4-b Dynamic Requirements

Fig. 4 An Requirements External Schema and its Requirements Description in PSL.


MEDICAL DATABASE
professor(faculty,name, birthday)
clinical-survey(year, blood-pressure, blood-type)
university-medical-checkup(year, height,weight, sight)
PROFESSOR DATABASE
professor(prof-no.,name,club,subject, lab-name)
laboratory(lab-name,establishment-year)
book(prof-no.,room-no.,number-of-books)
STUDENT DATABASE
student(student-no., faculty,department, name,address)
s-student-1
student-club-link
s-club
club(club-name, representative, prof-name)
s-student-2
student-lab
student-subject-link(mark)
s-subject
laboratory(lab-name, faculty,department, establishment-year)
subject(subject-name,prof-name, number-of-students)
Requirements Description

```
SET       MEDICAL-DB;
SET       PROF-DB;
 CSTS     PROF,LABO,BOOK;
ENT       PROF;
 CSTS     PROF-NO,NAME,CLUB,
          SUBJECT,LAB-NAME;
ELE       PROF-NO;
ENT       STUDENT;
ENT       LABORATORT;
REL       STUDENT-LAB;
 BTWN     STUDENT
  AND     LABORATORY;
 CONN     1 TO N;
ENT       SUBJECT;
 IDD      SUBJECT-NAME;
 CSTS     SUBJECT-NAME,
          PROF-NAME,
          NO-OF-STUDENT;
 CARD     1500;
```

Fig. 5 The Requirements Conceptual Schema of University DB.

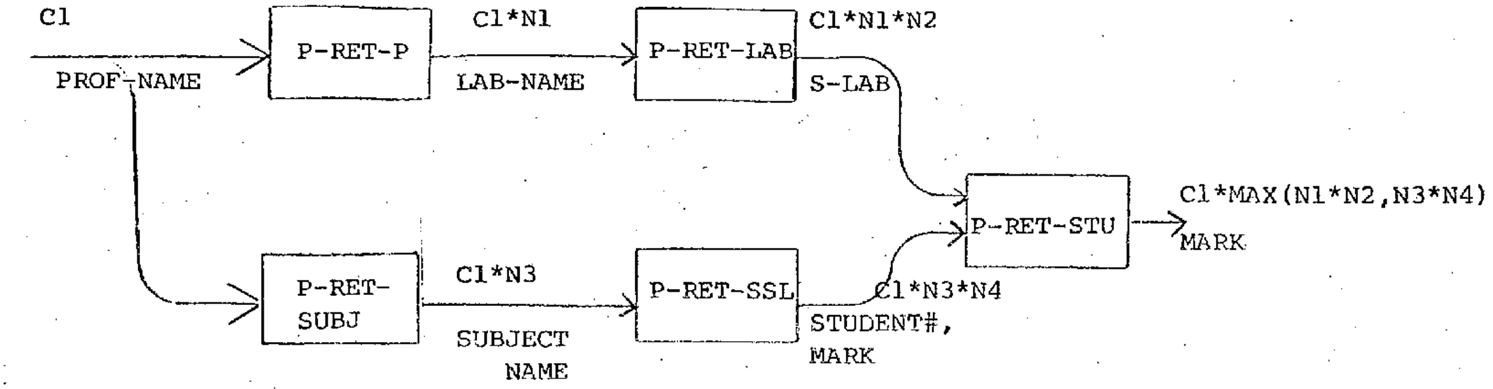


Fig. 6-a Calculation of Dynamic Requirements 1.

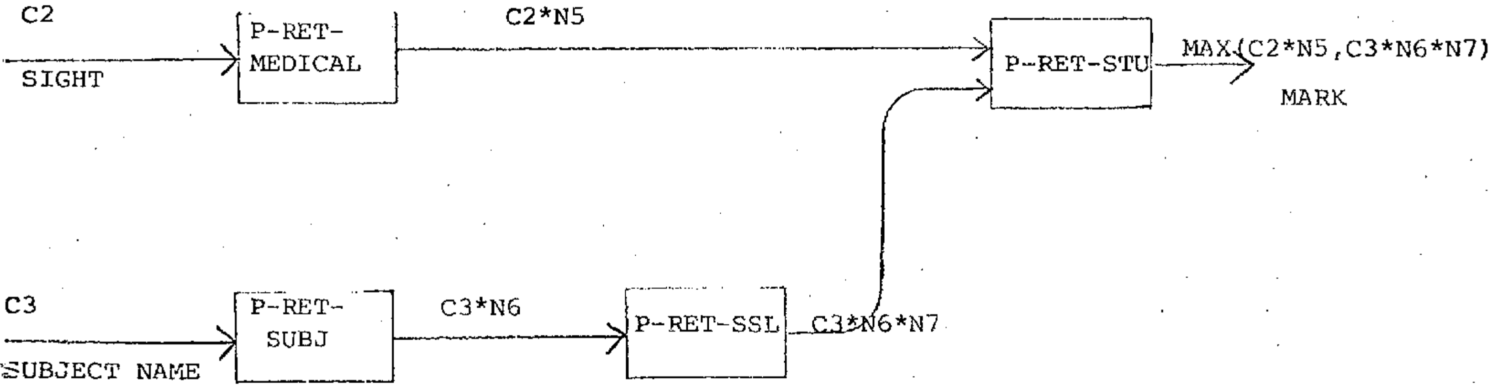


Fig. 6-b Calculation of Dynamic Requirements 2.

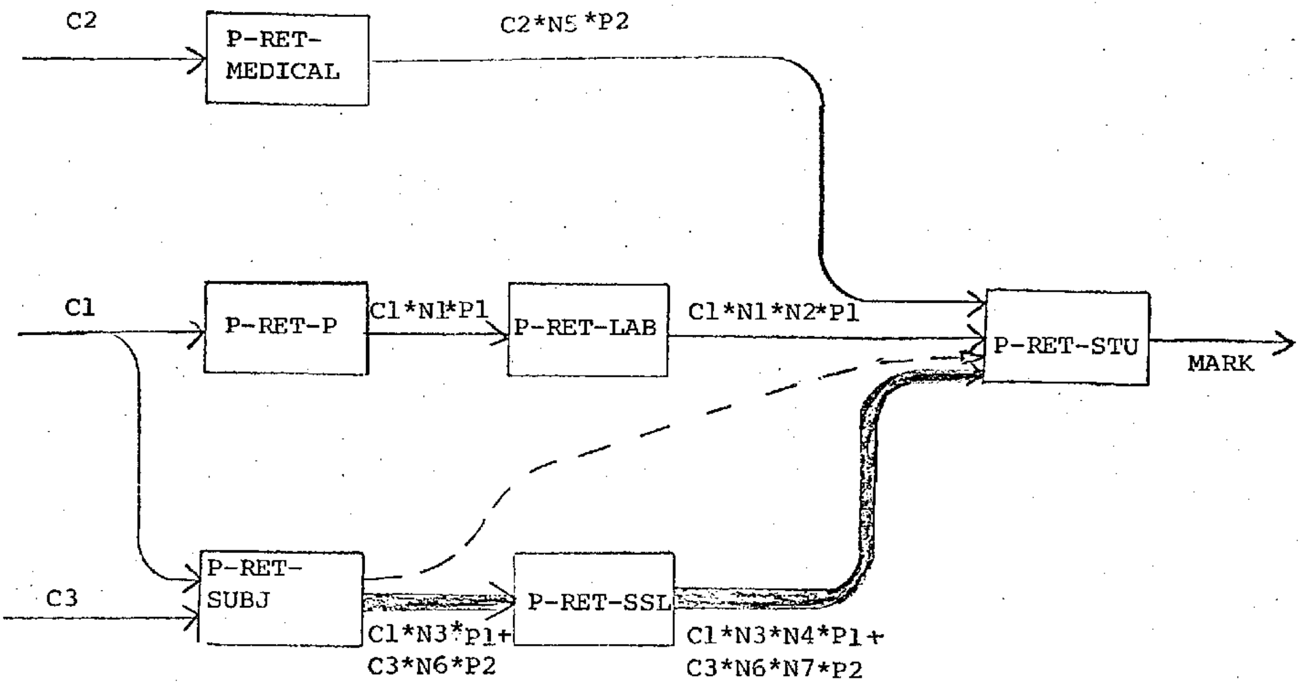


Fig. 6-c Integration of Dynamic Requirements.

Fig. 6 Calculation of the Dynamic Cardinality for View Restucturing.

JIPDEC Information Systems Seminar
on Semantic Aspects of Databases,
JIPDEC, Tokyo, Japan, Feb. 1980

Distribution Problems in Distributed Database Systems:
Integration and Query Decomposition

Makoto Takizawa

Development Department

Japan Information Processing

Development Center

3-5-8, Shiba-koen, Minato-ku,

Tokyo 105, Japan

tel. 03 (434) 8211 ext. 305

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

2. Four-Schema Structure (FSS) Approach to Designing DDBS

2.1 Four Schemas

2.2 Network Data Directory (NDD)

2.3 Design and Implementation

2.4 Unified Architecture

3. Integration and Distribution Information

3.1 Semantic Links (SL)

3.2 Distribution Description (DD)

3.3 Distribution Information (DI)

4. Query Decomposition (QD)

4.1 Basic Assumptions

4.2 Objectives

4.3 Strategies

4.4 Query Normalization (QN)

4.5 Query Modification (QM)

4.6 Initial Local Query Processing (ILQP)

4.7 Transmission Scheduling (TS)

4.8 An Example of the TS

4.9 The Architectures of the GDP and LDPs

5. Concluding Remarks

Acknowledgement

References

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 allocation 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 O A_2 represents that both relations

A_1 and A_2 have a set of tuples in common. A_1 U 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.

$\underline{\text{drange}}$ ($x_1,\ldots,x_m$) ($X_1{:}s_1,\ldots,X_m{:}s_m$) ;

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 GA.

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: <u>QPI/REL</u> (<u>site-no</u>, <u>rel-no</u>, cardinality, width) and <u>QPI/ATT</u> (<u>site-no</u>, <u>rel-no</u>, <u>att-no</u>, 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 corresponds 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:

$j2$: $c(R1{:}1 \to 2) = c(R2{:}2 \to 1) = 500*2 = 1000$

$j4$: $c(R4{:}4 \to 1) = 100*5 = 500 \qquad |R4| < |R1|$

$j5$: $c(R3{:}3 \to 1) = 200*2 = 400 \qquad |R3| < |R1|$

$*\underline{j6:}$ $c(\underline{R4{:}4 \to 3}) = 100*1 = \underline{100} < 500 \qquad |R4| < |R3|$

$j8$: $c(R3{:}3 \to 4) = 200*1 = 200 \qquad |R3| < |R5|$

Hence, $R4{:}4 \to 3{:}R3$ 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, $R5{:}4 \to 3{:}R3$ is selected[see Fig.4.2b], because R3 and R5 are marked DEST and FREE, respectively, and $c(R5{:}4 \to 3) = 300 < 500$ 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:

$j2$: $c(R2{:}2 \to 1) = c(R1{:}1 \to 2) = 500*2 = 1000 > 500$

$j9$: $c(R1{:}1 \to 3) = 500*3 = 1500 > 500$.

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 $c(R3{:}3 \to 1) = 600 > 500$, the THV value is reset, i.e. $THV \leftarrow (1500 + 1000 + 500)/3 = 1000$.

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.

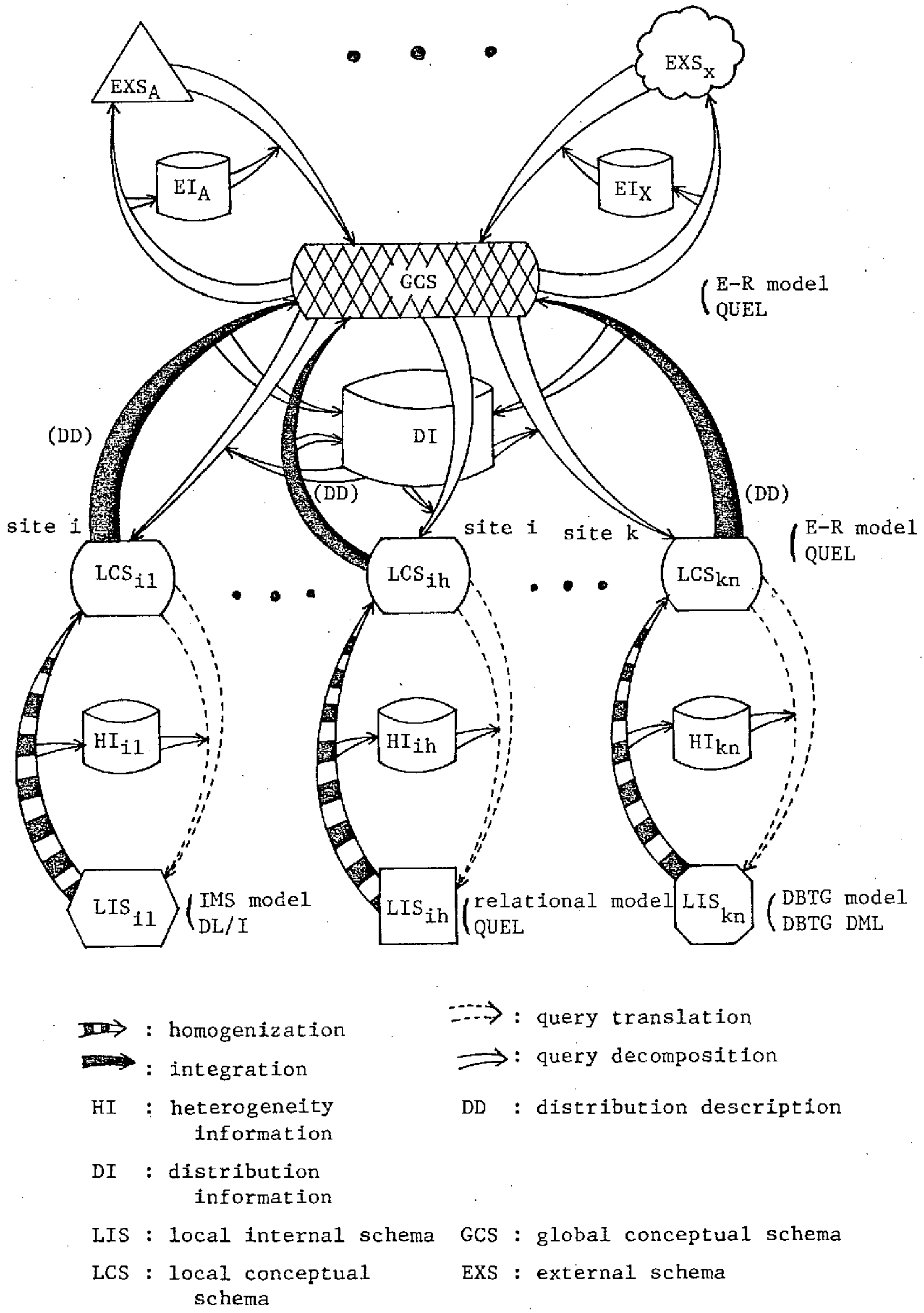
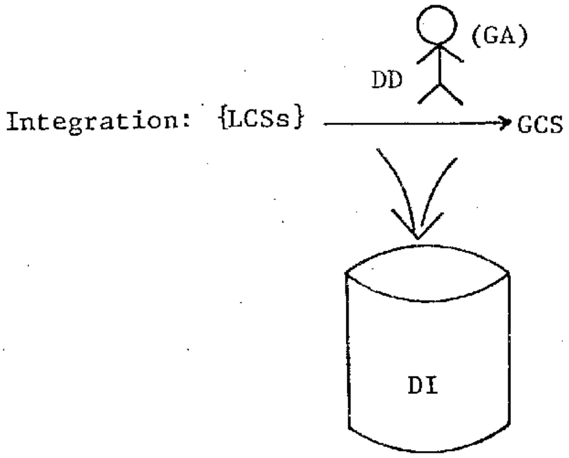


Fig. 2.1 Four-Schema Structure (FSS)



DD: distribution description
DI: distribution information
GA: global administrator
GCS: global conceptual schema
LCS: local conceptual schema

Fig. 3.1 Overview of Integration

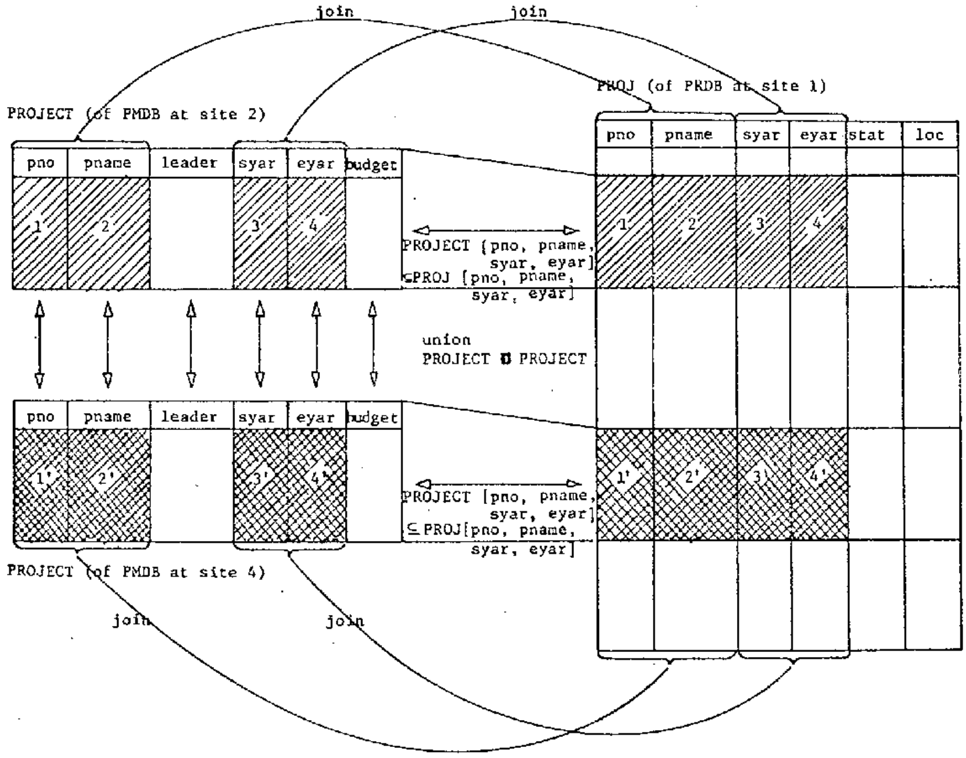

join
join
PROJECT (of PMDB at site 2)
PROJ (of PRDB at site 1)
pno
pname
leader
syar
eyar
budget
1
2
3
4
pno
pname
syar
eyar
stat
loc
PROJECT [pno, pname, syar, eyar] ⊆ PROJ [pno, pname, syar, eyar]
union
PROJECT ∪ PROJECT
pno
pname
leader
syar
eyar
budget
1'
2'
3'
4'
PROJECT [pno, pname, syar, eyar] ⊆ PROJ[pno, pname, syar, eyar]
PROJECT (of PMDB at site 4)
join
join

Fig. 3.2 Relationships Between Three Relations

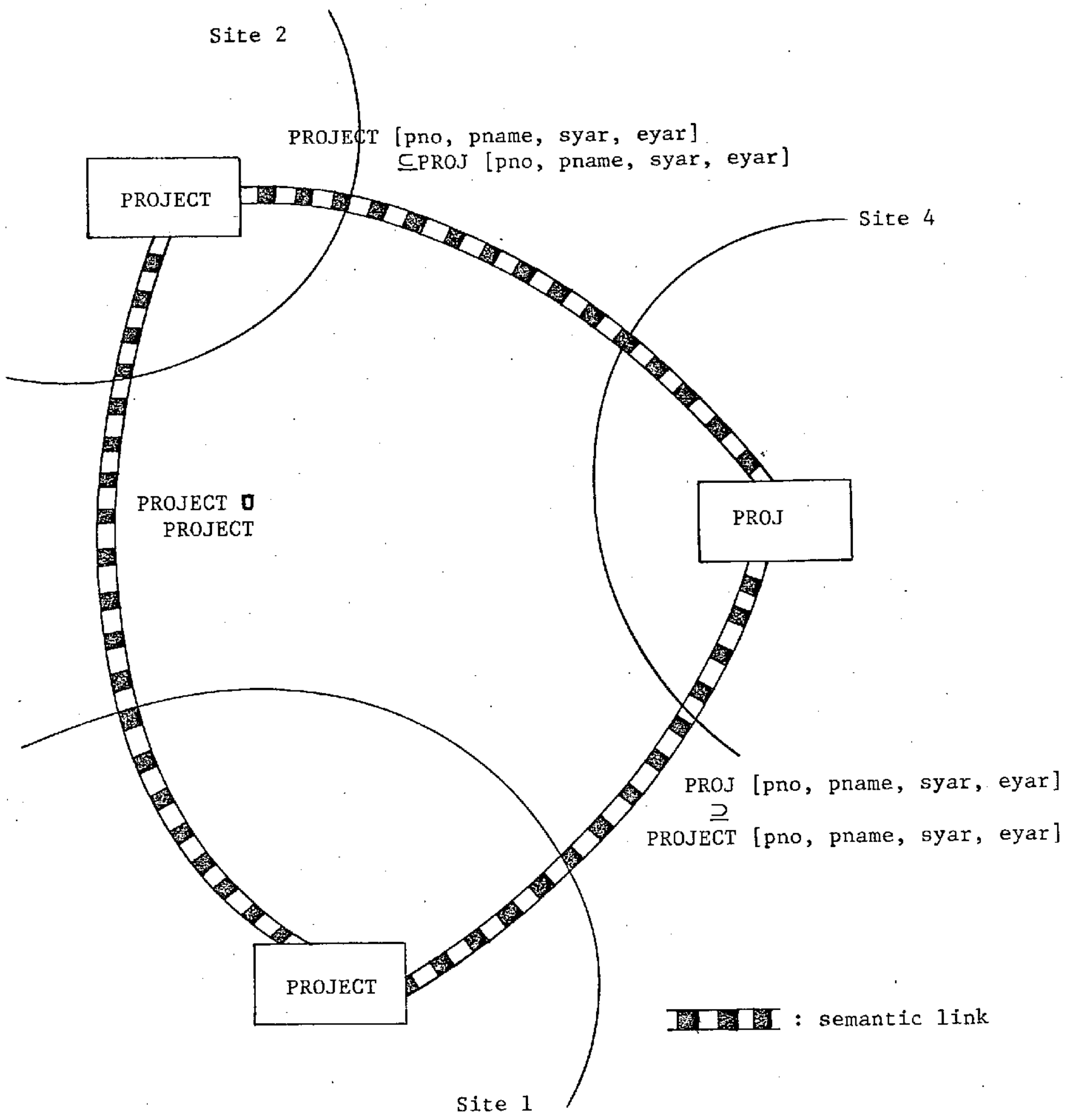

Site 2
PROJECT [pno, pname, syar, eyar]
⊆PROJ [pno, pname, syar, eyar]
PROJECT
Site 4
PROJECT ∪
PROJECT
PROJ
PROJ [pno, pname, syar, eyar]
⊇
PROJECT [pno, pname, syar, eyar]
PROJECT
: semantic link
Site 1

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          )
      ┌  ( p.pno, p.pname, manager=p1.leader, budget=p1.budget, p.loc )
 (1) ┤  where  p.pno = p1.pno and p.pname = p1.pname and p.syar = p1.syar and
      └         p.eyar= p1.eyar :
      ┌  ( p.pno, p.pname, manager=p2.leader, p2.budget, p.loc        )
 (2) ┤  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

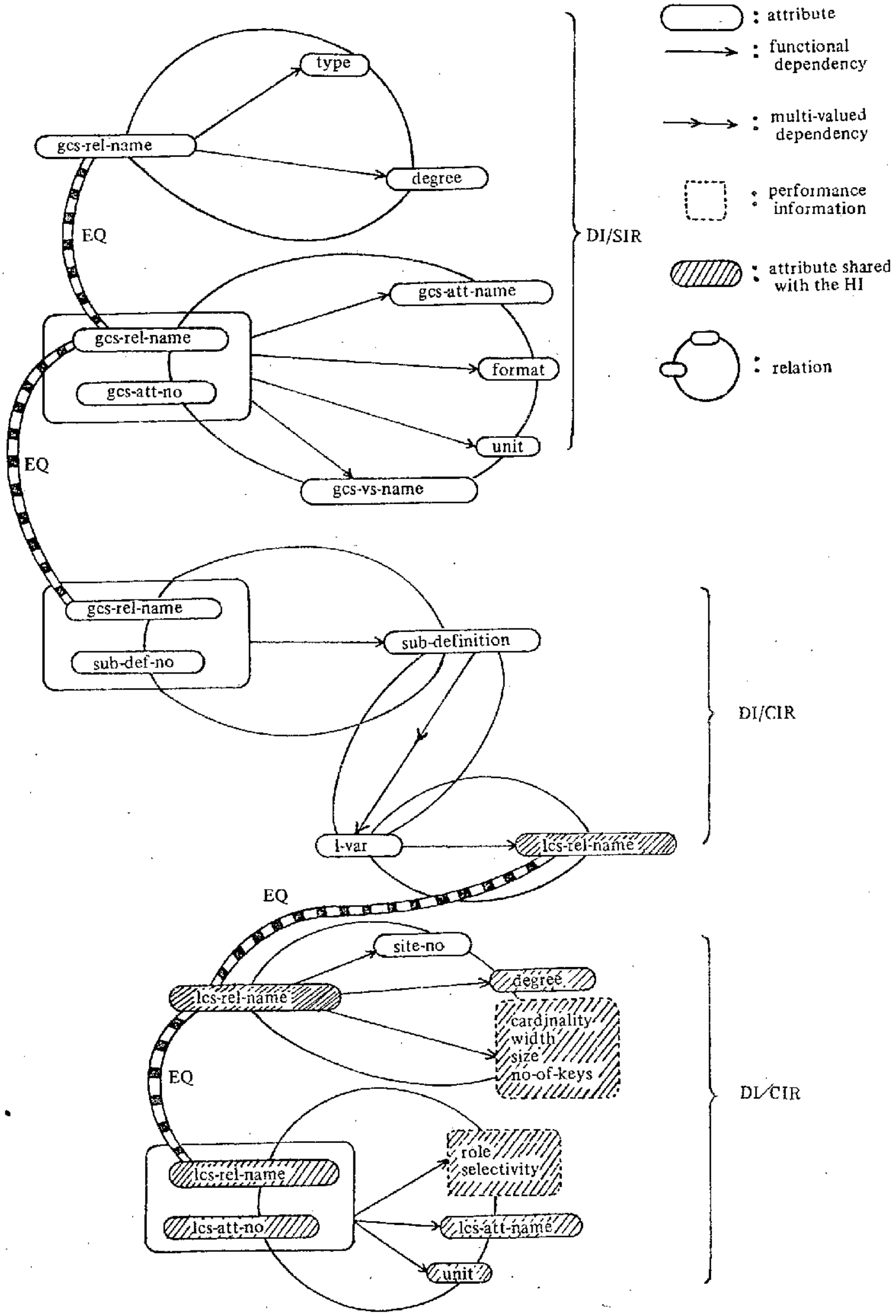
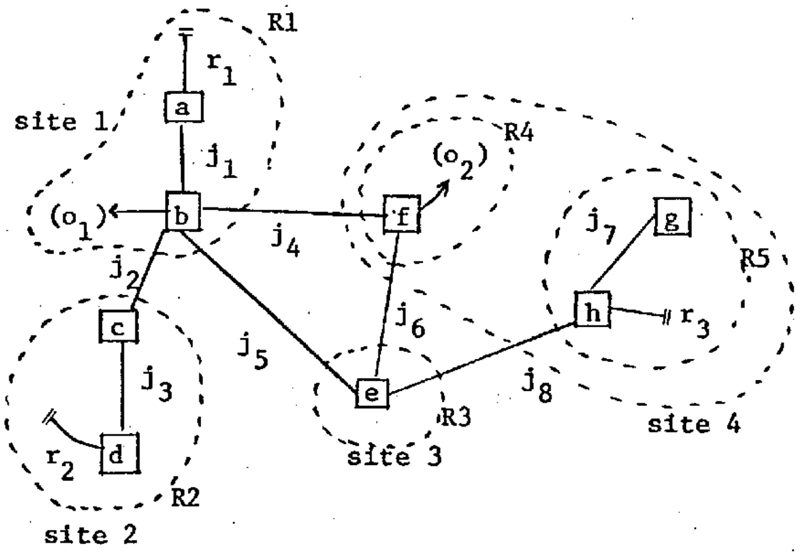


Fig. 3.5 Distribution Information (DI)



□ : node for a tuple variable

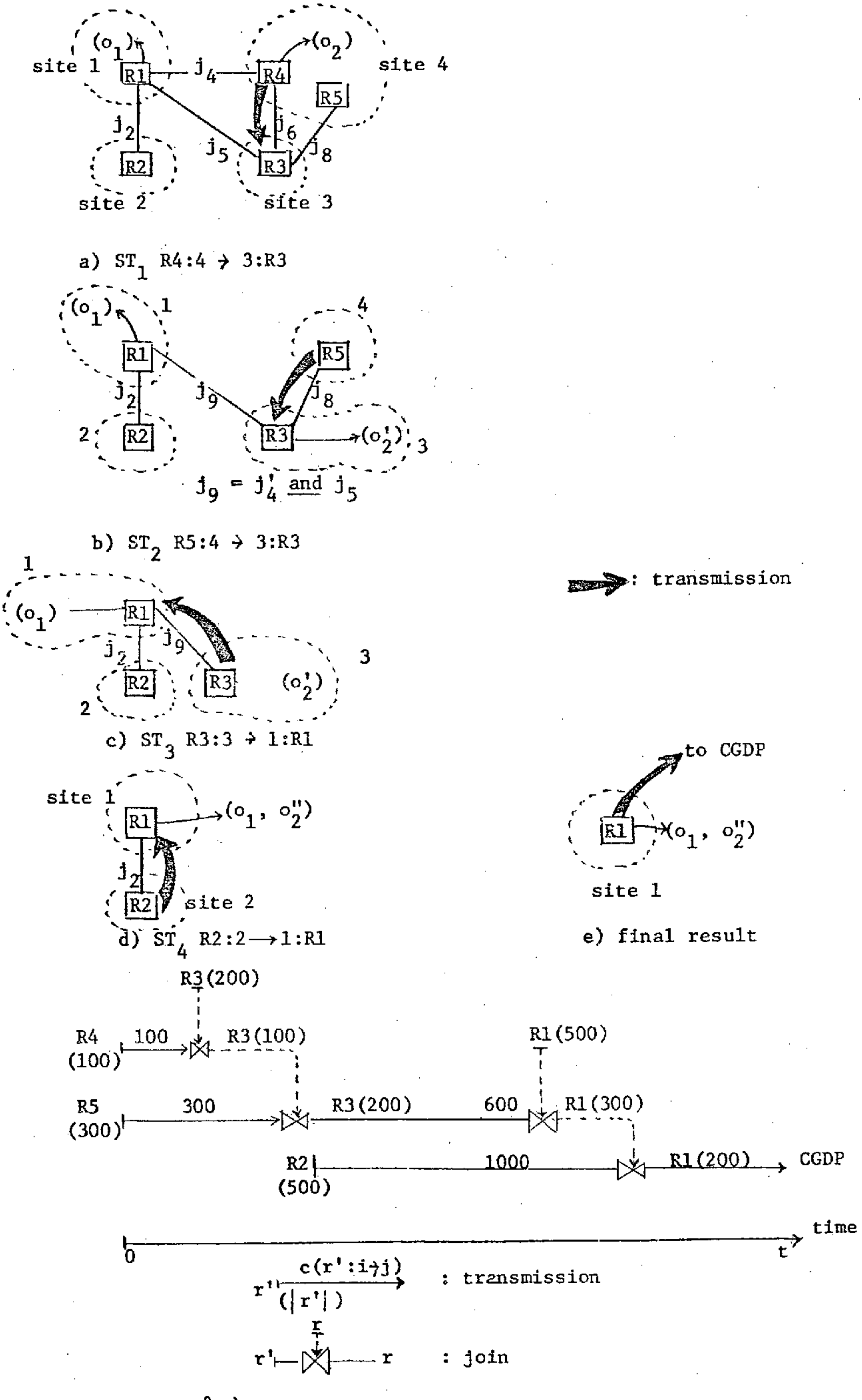
□—j—□ : join link

‖—r—□ : restriction link

□——→(o) : result link

Ri: result relation generated by the ILQP.

Fig. 4.1 An Example of the GLQG



a) ST_1 R4:4 → 3:R3

b) ST_2 R5:4 → 3:R3

c) ST_3 R3:3 → 1:R1

d) ST_4 R2:2 ⟶ 1:R1

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)  $c(r':i \rightarrow j)$ 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, $r':i \rightarrow j:r$.
     - 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

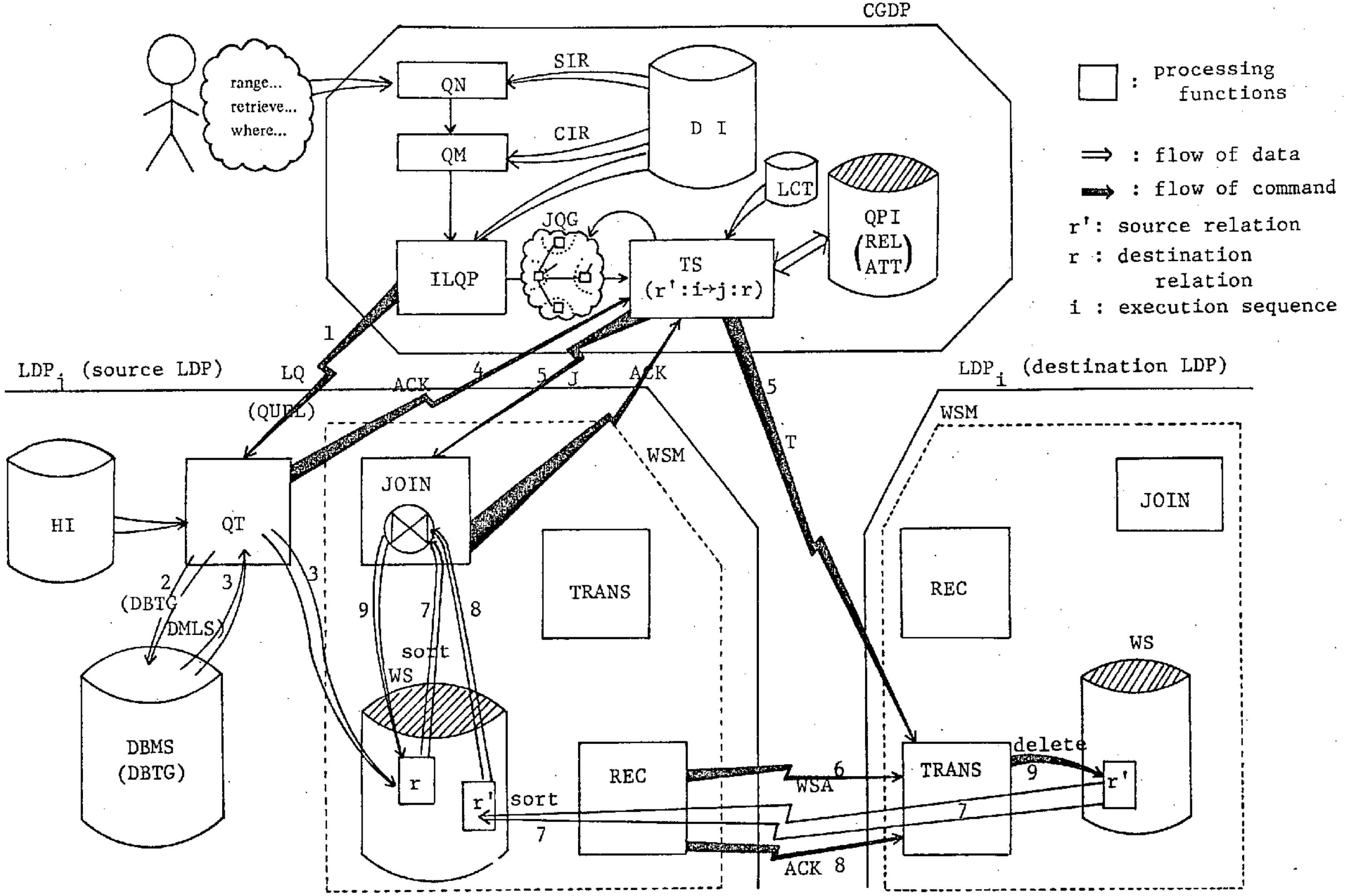


Fig. 4.7 The Architectures of the CGDP and LDPs