データベース・システム　75－2
（１９９０．１．１８）

エクステンジブルＤＢＭＳにおける最適化処理についての考察

関根　裕†，鎌田伸一‡，林　知博†，弘末清悟†
†富士通㈱　ソフトウェア事業部
‡富士通神戸エンジニアリング㈱

近年，エクステンジブルデータベースシステムに関する研究者が増えてきている．また，初期プロトタイプも作成されてきている．こうしたシステムのキーポイントでもある最適化処理の構築技術について，ウインストンらのルールシステムを応用したルールベースオプティマイザの試作を踏まえて検討したので報告する．

A PROTOTYPE IMPLEMENTATION OF QUERY OPTIMIZATION FOR AN EXTENSIBLE DATABASE SYSTEM

Yutaka Sekine†　Shinichi Kamata ‡　Tomohiro Hayashi†　Seigo Hirosue†
†Software Division, FUJITSU LIMITED
‡FUJITSU KOBE ENGINEERING LIMITED

140 Miyamoto, Numazu-shi Shizuoka, 410-03 JAPAN

In last decade, number of the researchers of extensible database systems has been growing, and their early prototype versions has been developed. The implementation techniques of query optimization is a key to those systems. We discuss an implementation of a rule-based optimizer prototype using Winston and Horn's rule system.

1．はじめに

１９８６年のＳＩＧＭＯＤ国際会議において，エクステンジブルデータベースシステムと題するパネル討論が行われた模様である［３］．筆者らは，この会議に参加していないので，どのような様子であったかはわからないが，この方面に多くの研究が行われていることをプロシーディングから窺い知ることができる．

エクステンジブルデータベースシステムの研究には，大きく２つの流れがあるように思われる．ひとつは，広い応用分野に適用可能な万能ＤＢＭＳの完成をめざすものであり，もうひとつは，適用分野向きのＤＢＭＳを作り出す，もしくは同等の機能を提供するための基盤を整備しようとするものである．

ＰＯＳＴＧＲＥＳ［14］は，リレーショナルモデル上で複合オブジェクトをサポートし，ポータルと呼ばれるデータベース操作言語や，アラータ，トリガなどのサポート，光ディスクを念頭に，工夫された格納機構などを特徴とするデータベースシステムである．ここでのエクステンジビリティは，データ型やオペレータ，組み込み関数，およびアクセスメソッドの利用者定義を可能にすることで実現しようとしている．このために最適化処理をテーブル駆動型にするといった工夫がある．

ＥＸＯＤＵＳ［４］は，完成したＤＢＭＳではなく，むしろ特定応用向き高性能ＤＢＭＳを早期に開発するための“ｔｏｏｌｋｉｔ”と考えられている．これは，ひとつのＤＢＭＳでは，広汎な適用分野の機能的／性能的な要求を満たすことはできないという考えによる．また，データベース操作言語の多様性をも認め，それぞれに最適化処理が必要になるという考えから，最適化処理のジェネレータを用意している．これは，データベース操作言語のオペレータとこれらに合理的に適用できる変換，および各オペレータを実行できるメソッドの記述とから，最適化処理をＣソースコードの形で出力する．

ＳＴＡＲＢＵＲＳＴ［９］は，特定応用向きにＤＢＭＳをカスタマイズできるような“オープンデータベースアーキテクチャ”を目指している．ここでは，ルール形式で最適化処理を記述することによって，エクステンジビリティを実現しようとしている．

この他にも，ＰＲＯＢＥ，ＧＥＮＥＳＩＳなど［３］があり，それぞれの立場からエクステンジビリティを標榜している．

最適化処理に対しては，可変要素を単に変数にするものから，その仕様記述によって最適化処理を生成しようとするものまで，いくつかのアプローチがある．いずれにしても，最適化処理の構築技法がひとつのキーポイントになると考えられる．

2．データベースにおける最適化処理

データベースにおける最適化の問題領域を手短かに整理しておきたい．ここでは，データベース操作言語と実行手順，データベースの格納機構，および発見的手法による探索という３つの側面から述べる．

2．1　データベース操作言語と実行手順

データベース操作言語として，ＳＱＬ［11］の標準化が達成されている．最適化処理には，このようなデータベース操作言語によって記述される，いわゆる宣言的なデータベース操作をできるだけ効率良く実現する手続き的な実行手順に変換することが要求される．

ＳＱＬに関しては，より多くの機能を持ったＳＱＬ２／３［１］へと引き続き標準化作業が続けられており，最適化の対象となるデータベース操作言語は，日々進歩して行くと考えてよいだろう．この例に，アウタジョイン［１］やビュー更新［13］などがある．

宣言的なデータベース操作を実現する手続き的な実行手順も工夫されてきた．たとえばジョイン技法でも，ネステドイテレーション［12］から，マージジョイン［12］，さらにハッシュジョイン［５］などが考案されている．これらは，その局面毎に適したジョイン技法として最適化処理による選択の対象になる．

２．２　ＤＢＭＳの格納機構

最適化処理は，対象となる格納構造をもとに手続き的な実行手順を作成する．したがって，格納構造が要求されたデータベース操作指示に適合していないなら，最適化処理がその実行手順をいくら工夫しても，データベースアクセスの絶対性能はでない．

たとえば，論理構造で表現される複数のデータが，強い相関を持ってアクセスされる場合，物理的にはこの相関に基づいて格納することによって良い性能が導かれる．格納構造は，こうした特定のアクセスパターンの頻度や相対的な重要性によって決められる．

従来から，商用のデータベースシステムは，このために種々の格納構造を提供してきている．Ｂatory［２］は，こうしたデータベースシステムを分析して，格納構造のモデルを提案している．ここでは，格納構造を，概念－内部マッピング，ファイル構造，レコードリンク機構を組み合わせた『格納機構』ととらえている．商用のデータベースシステムは，それぞれこのモデルで表現され得る『格納機構』を部分的にしか実現していない．

最適化処理を構築する上で，こうした格納構造のモデル化は極めて重要と思われる．宣言的に記述されたデータベース操作を忠実に実現する格納構造のアクセス手順を生成するためには，格納構造に対する十分な知識が必要であり，この知識を一般的に表現するためには，何らかのモデルが必要だと考えるからである．

限定された格納構造の範囲だけで最適化処理を考えるなら，この知識をプログラムコードに直接書き下すことも可能だろうが，こうした最適化処理は，極めて拡張性の乏しいものとなるだろう．

２．３　発見的手法による探索

最適化の難しさは，きわめて単純なデータベース操作を除いて，一般に生成可能な実行手順が膨大な量になり得ることである．対象となる格納構造のアクセスパス，探索条件のアクセスパスへの適用方法，および最適化の持つ実行手順のパターンの組合せによって，与えられたデータベース操作を実現することのできる実行手順の数は，現実の計算機が合理的な時間で生成し得る範囲を超えるものになってしまう［12］．

したがって，最適化処理は実際には，このような可能な実行手順の全数を調べ上げることはせずに，いわゆる発見的手法と呼ばれる人為的な探索空間の限定を行う．このことから，最終的に選択する手順は，必ずしも最適であるとは限らない．むしろ最悪を避け，平均的なプログラマが思い付く程度の実行手順を合理的な時間の範囲でいかに生成するかが目標になる．

3．ルールベースシステムと最適化処理への応用

最適化処理を直接，ルールベースシステムとして記述しようとする試み［6，7，9］だけではなく，ＥＸＯＤＵＳ［8］のように最適化処理のジェネレータへの入力としてもルール記述が使われているなど，エクステンジブルデータベースシステムでは，何らかの形で最適化処理とルールが関わっている．

ルールベースシステムは，それ自体固有のシステムが存在するわけではなく，同じような性質を持つ問題解決のためのひとつの手段と考えられている［10］．また，ルールベースシステムの持つ共通の重要な性質として，以下の５つが挙げられている．

① 知識のｉｆ－ｔｈｅｎ形式をした条件による表現．
② 知識ベースの拡張に比例したルールベースシステムの性能向上．
③ 適切なルールの選択と結果の誘導によって，広範で複雑な問題に対処可能．
④ 状況に応じたルールの適用順序の選択．
⑤ ルールの適用と結論に対する説明．

最適化処理の性質は，こうしたルールベースシステムの持つ性質に酷似している．対象となるデータベース操作と対応する実行手順の知識，格納構造の知識などから，適切な実行手順を選択する過程は，ルール適用の過程に置き換えることができるだろう．

また，データベース操作や格納構造などを入力として，可能な実行手順をすべて生成できるようなルール群に対して，適用ルールの適切な選択と結果の誘導を行うことが，いわゆる発見的手法に対応付けられることになろう．

4．ルールベースオプティマイザの試作

ウインストンら［15］のルールシステムを拡張して，最適化処理に適用することによって，簡単なルールベースオプティマイザの試作実験を行ったので，この概要を説明する．

4．1　試作ルーベースシステムの概要

ウインストンらのルールシステムは，ＬＩＳＰ上のパターンマッチングを主体としている．ここでは，このシステムにバックトラッキングの組み込みなどの拡張を行っている．図１に試作したルールベースシステムの概要を示す．プログラムの規模としては，ＬＩＳＰで約１００関数，約０．５Ｋ行である．

ルールドライバは，まずルールをひとつ取り出し，前提部に書かれたパターンとアサーション群とのマッチングをとる．パターンマッチングが成立した場合，アソシエーションと呼ばれる一時記憶にマッチングデータを蓄える．前提がすべて成立すると，結論としてアサーションを生成する．このとき，一般にアソシエーションの情報が付け加えられる．

すべてのルールを繰り返し適用し，新しいアサーションが生成されなくなった段階で停止する．作成されたアサーションの中に最終的な結果が得られている．

4．2　最適化処理の機能範囲

試作の対象とした最適化処理の機能範囲をひとことで言うと，B^+ ｔｒｅｅファイル構造にいくつかの二次インデックスを持つだけのフラットな概念－内部マッピング格納機構

と，ｎテーブルジョインだが，条件式にはＡＮＤで結合された等値比較しか持たないような検索，およびジョイン手法としてマージ技法に限定したものである．

合わせて２７個のルールを作成し，これらは，ＬＩＳＰで約１Ｋ行になった．

４．３　ルール適用の過程

ルール適用の過程を簡単な例（図２）を使って説明する．ルールには，キーアクセス可能なアクセスパスの抽出ルールを使う．このルールは，問い合わせの構造を認識するためのパターンと格納構造の持つアクセスパスを認識するためのパターンの２つの前提部と，パターンマッチングが成立した結果として，特定のアクセスパスを使った問い合わせの実現を示すアサーションを生成するためのひとつの結論部から成っている．

ここでは，C1，C2，C3という３つのカラムからなる，T1というテーブルから，C1の値が1000であり，かつ，C3の値が 100であるようなレコードの，C1とC2の値を取り出すという，簡単な問い合わせを対象とする．

T1には，次のような３つのアクセスパスが存在するものとする．

(1) PATH1 …… レコードＩＤ(TID) を与えると，対応するレコードのカラム値のすべてを取り出すことのできるパス

(2) PATH2 …… C3のカラム値を与えると，対応するレコードのC3の値と，レコードＩＤを取り出すことのできるパス

(3) PATH3 …… C1のカラム値を与えると，対応するレコードのカラム値のすべてを取り出すことのできるパス

パターンマッチの仕掛けとして，“(> 変数)”と“(+ 変数)”がある．これらは，それぞれ何らかのアトム，またはリストにマッチングするパターンであり，これら以外はすべて即値として比較される．

たとえば，ルールの前提部の前半は，問い合わせの構造を解釈するものであり，パターンマッチの過程で，SELECT-LIST という変数に，(T1.C1 T1.C2) というリストが対応付けられる（アソシエーション）．

変数に対応付けられた値は，“(< 変数)”というパターンで取り出すことができる．前提部にあれば，パターンマッチの対象となり，結論部にあれば，生成するアサーションのために使われる．

この例では，T1.C1 に関する等値条件からPATH3 ，T1.C3 に関する等値条件からPATH2 という２つのキーアクセス可能なアクセスパスが抽出されている．なお，このルールでは，プロジェクションに関するチェックを行っておらず，必要なカラムデータが取り出せないアクセスパスであってもよいものとしている．

４．４　試作実験における考察

今回の試作実験では，最適化処理のルール記述に関する感触を得ることを第一の目的としていた．この目的は，ほぼ達成されたと思う．試作したプログラムでは，検索木を初期入力に順次ルールを適用して，より具体的な検索木（実行手順）へと変換することによって，最終的に実行可能な手順を得ることができる．

試作システムでは，ルールを駆動するルール，いわゆるメタルールは作らずに，ルール

ドライバによる単純な全件処理しかしていない．実際には状況に応じたルールの適用順序の選択や，ルールの選択による適用知識の限定などがかなり必要と思われる．これ自身知識であることから，やはりルールと同じようにモジュール化するべきであろうが，ルールと同じ土俵で考えることができるかどうか（すべきかどうか）はこれからの課題である．

また，試作の過程で，ルールベースシステムのデータ構造が大きな問題になることがわかった．つまり，アサーションとして，どのような意味付けをもったものを用意するかである．

いくつかのアサーションを入力として，パターンマッチングを経由して，いくつかのアサーションを作り出すという単純なプロセスでは，ひとつの意味ある機能を実現するために，いくつかのルールを組み合わせて構成しなければならないことが多い．

したがって，ルールにも構造ができ，ルール間のインタフェースであるアサーションにもまた内部アサーション，外部アサーションなどといった構造ができる．試作システムでは，すべてフラットになっているが，こうした構造を明示的に表現できるルールベースシステムである必要があろう．

また，アサーション間の関係には，問い合わせ（または，実行手順）の木構造を表現するものと手順の代案を表現するものがあり，これが錯綜してしまった．このような関係もまた，明示的に表現し操作できるものである必要がある．

このような問題は，伝統的な最適化処理の作成方式でも同様に存在し，ルールベースにしたからといって何ら変わるものではなく，単に伝統的なプログラミング言語が“ルール言語”に変わっただけのようにみえる．ただし，適当なルール言語であれば，伝統的なプログラミング言語に比べて，記述量の大幅な減少は期待できる．

データ構造の設計を伝統的なプログラミング言語と同じように，適切に行う必要があるとすれば，ルールベースとすることの意味は，このような記述量の減少こそが，エクステンジビリティに寄与するのではないだろうか．むしろ，オプティマイザ記述言語を作るというスタンスで，ルールベースシステム化を考えるべきなのだろう．ただ，具体的にどのようなものであればよいかは，現時点ではオープン問題と考える．

5．まとめ

データベース操作言語や格納機構などが複雑になっていくにつれ，最適化処理の構築も難しくなってきている．従来のプログラミング言語による書き下しでは，もはや，ニーズに迅速に対応することができなくなりつつある．

これからのデータベースシステムが，『エクステンジブルデータベースシステム』でなければならないのかどうかは別にして，『エクステンジブルオプティマイザ』の構築技法を早急に確立しなければならないことには間違いない．

このひとつのステップとして，ルールベースオプティマイザの試作実験を行った．この経験を活かして，実用になる『エクステンジブルオプティマイザ』の実現に向けて努力していきたい．

参考文献

[1] ANSI X3H2-89-110 ISO DBL CAN-3, "ISO-ANSI(working draft) Database Language SQL2 and SQL3", February 1989.

[2] Batory, D. S., "Modeling the Storage Architectures of Commercial Database Systems", ACM Transactions on Database Systems, Vol.10, No1.4, December 1985.

[3] Batory, D. S., Mannino, M., Carey, M., Dayal, U., Mohan, C., and Rowe, L., "Panel on Extensible Database Systems", ACM SIGMOD International Conference on Management of Data, 1986.

[4] Carey, M. J., DeWitt, D. J., Frank, D., Graefe, G., Muralikrishna, M., Richardson, J. E., and Shekita, E. J., "The Architecture of the EXODUS Extensible DBMS", Proceedings of Object-Oriented Database Workshop, 1986. (Also in Readings in Database Systems, Stonebraker(ed)).

[5] Dewitt, D. J., Katz, R. H., Olken, F., Shapirio, L. D., Stonebraker, M. R., and Wood, D., "Implementation Techniques for Main Memory Database Systems", ACM SIGMOD International Conference on Management of Data, 1984.

[6] Freytag, J. C., and Goodman, N., "Rule-Based Translation of Relational Queries into Iterative Programs", Proc. SIGMOD 86, 1986.

[7] Freytag, J. C., "A Rule-Based View of Query Optimization", ACM SIGMOD International Conference on Management of Data, 1987.

[8] Graefe, G., and DeWitt, D. J., "The EXODUS Optimizer Generator", ACM SIGMOD International Conference on Management of Data, 1987.

[9] Haas, L.M., Freytag, J.C., Lohman, G.M., and Pirahesh, H., "Extensible Query Processing in Starburst", ACM SIGMOD International Conference on Management of Data, 1989.

[10] Hayes-Roth, F., "Rule-Based Systems", Communications of the ACM, Vol.28, No.9, September 1985.

[11] ISO/IEC 9075: 1989(E), "Information processing systems - Database Language SQL with integrity enhancement", Second edition 1989-04-01.

[12] Jarke, M., and Koch, J., "Query Optimization in Database Systems", ACM Computing Surveys, Vol.16, No.2, June 1984.

[13] Keller, A. M., "Choosing a View Update Translator by Dialog at View Definition Time", Proceedings of the Twelfth International Conference on Very Large DataBases, August 1986.

[14] Stonebraker, M., and Rowe, L. A., "The Design of Postgres", ACM SIGMOD International Conference on Management of Data, 1986.

[15] Winston, P. H., and Horn, B. K. P., "LISP, Second Edition", pp.269-283, Addison-Weslay, Reading, MA, 1984.

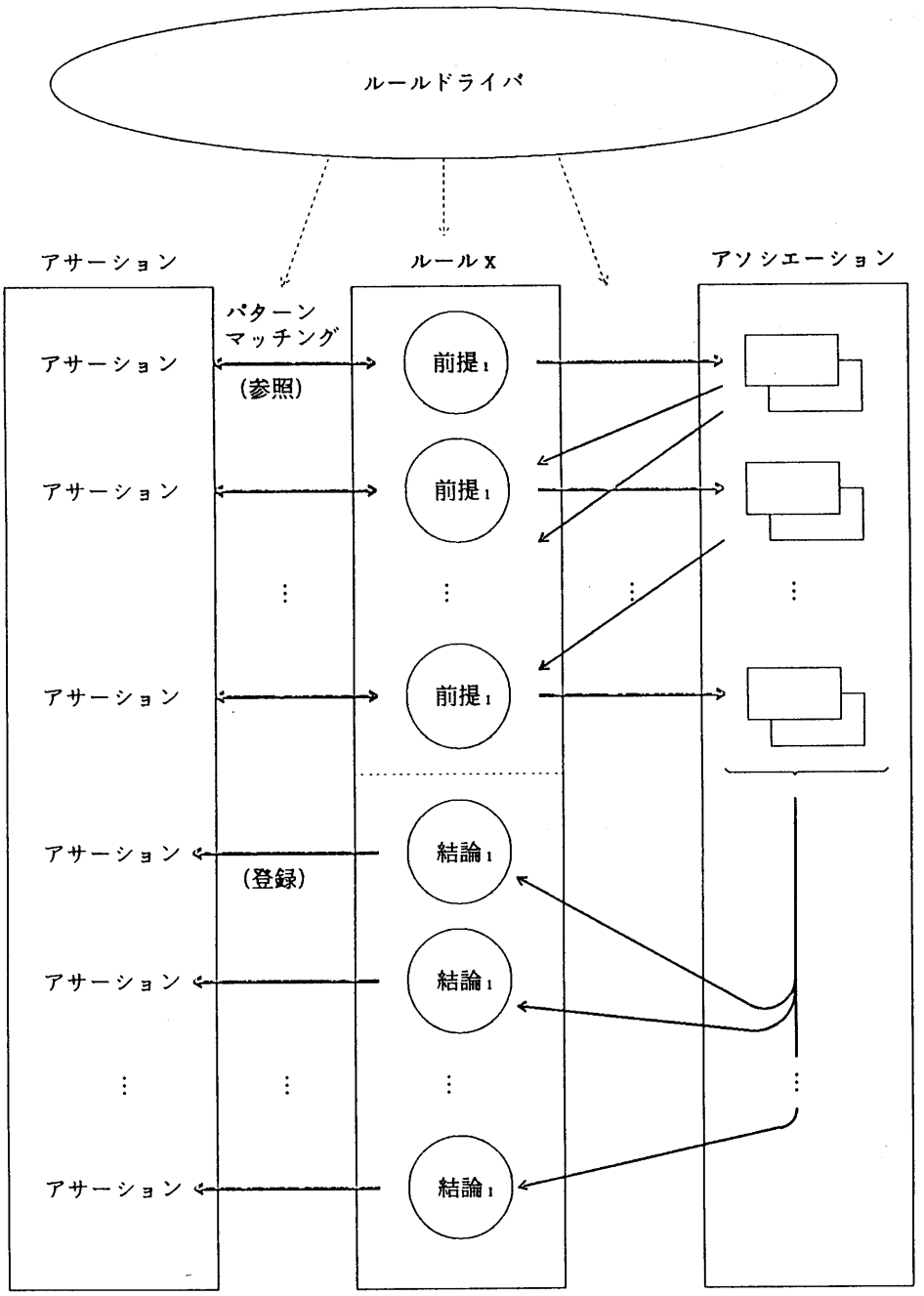

図1　試行ルールベースシステムの概要

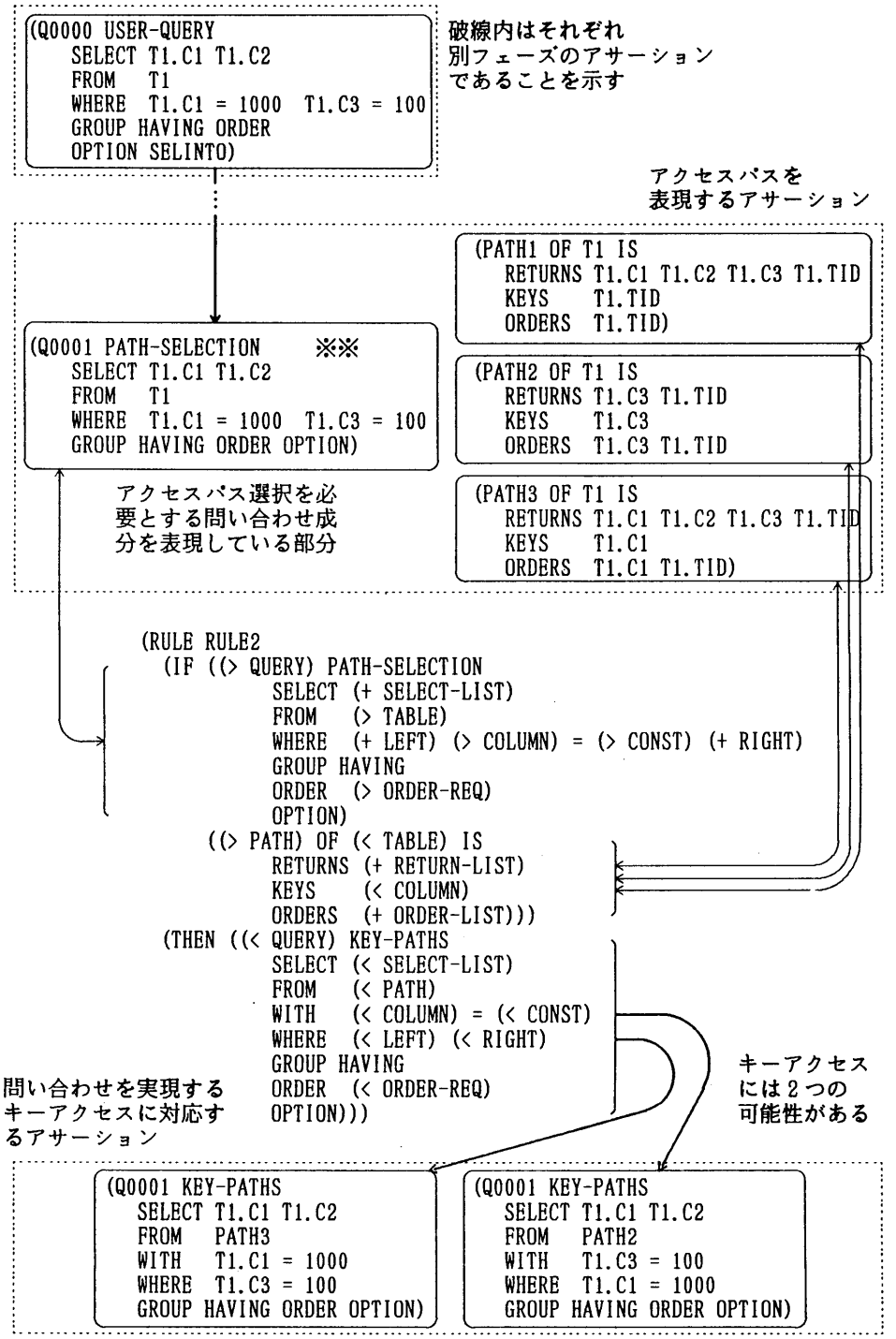

図2　ルールの適用例：キーアクセス可能なアクセスパスの抽出