3N-5

ルールベースに基づく最適化処理の設計

陳漢雄* 于旭* 山口和紀**
北川博之** 大保信夫** 藤原譲**
(*筑波大学工学研究科 **筑波大学電子情報工学系)

概要:
近年、エンジニアリング・データベース設計に関する研究がさかんである。しかしながら、各応用に特有なデータ・モデルを統合支援する目的での汎用データベース・システムの設計は極めて困難である。その原因としては、それらのデータ・モデルに依存する新たなアクセス手段の提供、現存のアクセス手段の有効使用などの困難な点が挙げられる。これらの問題を改善するために、拡張可能なデータベース管理システムの設計の必要性が明らかになりつつある[1][2]。特に問い合わせ処理における最適化設計のコストを軽減するために、ルールベースに基づく最適化処理の設計[3]が有効とされる。今回、抽象データ型(Abstract Data Type:ADT)を定義可能な拡張関係データ・モデルを取り挙げ、ルールベースに基づく最適化処理の設計について論じる。

本文:
現在,我々は各応用データ・モデルを統合支援することを目的とする拡張可能データベース管理システム(Extensible DBMS) MODUSを研究・開発中である。今回、取り挙げた拡張関係データ・モデルは、従来の関係演算と共に抽象データ型に対する演算を有する。抽象データ型の定義は次のように与えられる。抽象データ型Aは、Aのデータ構造Sとデータ構造Sに付随する関数群Fから構成される。従来の関係演算に関しては、代数規則のルールベースによる記述が可能である[3]。抽象データ型に関して、一般的に1つのデータが非常に大きくなりうるため、直接的なインデクスの実現は困難である。システムの現存のアクセス手段をより有効に利用する目的で、各抽象データ型に付随する関数間の関係および関数と基本定義域間の関係を記述するルールを導入する。

ルールベース

1.代数規則
代数規則は関係型DBMSの関係演算を形式化したもので、主に

transform rule: (t1 -> t2) と
implement rule: (t1 -(C)-> t2)

の2種類がある。前者は等価な代数規則の書き換えを表し、後者は代数規則とアクセス方法との結び付きを表す。以下にその例をあげる。

((JOIN t1 (... t2 ...)) -> (JOIN t1 (... ...) t2))
((TJOIN (... p1 ...) (TJOIN (...) t1 t2) t3) -(C)->
(TJOIN (... ...) (TJOIN (p1 ...) t1 t2) t3))
with C=(T(p1) in (T(t1) or T(t2)))

ただし、TJOINはJOINの中間(intermediate)結果で、条件によってLoop-JOIN またはMerge-JOINで実現される。Cは条件、t1,t2,t3はJOINされる式、p1はセレクション述語、T(s)はsの関連するリレーションを返す関数である。前者は書き換え規則で、連結文を変形する。後者は実現規則であり、セレクション述語をpush downする役割を果たす。

2. 抽象データ型を扱う規則
抽象データ型は、属性規則、関数規則の2組の規則により支援される。
属性規則は関数と基本定義域間の関係を記述するもので、例として次のようなものがある。

(f(t1) op f(t2) -(C)-> (x op f(t2)))
with C=((Ind(x,T(t1)) and ({x} in A(f)))

ただし、t1,t2が抽象データ型で、Cは条件を表し、fは抽象データ型上の関数、関数Indはxが T(t1)の表すリレーション上のインデクス・キーである時Trueを返し、A(f)はfの値域となる属性を返す。このルールではfの値域にインデクスが付随してあるとき、直接検索するかわりに、インデクスを用いられることを表す。
実現上の問題と実行時の効率を考慮して、我々は抽象データ型の関数を2種類に分ける。1つは基本関数で、各値域に対し、システム側にインデクスを付

Design of Rule-Based Optimization
Hanxiong Chen*,Xu Yu*,Kazunori Yamaguchi**,
Hiroyuki Kitagawa**, Nobuo Ohbo**, and Yuzuru Fujiwara**
*Program in Engineering Sciences, Univ. of Tsukuba.
**Institute of Information Sciences and Electronics, Univ. of Tsukuba.

随する。もう1つはユーザ定義関数で、可能なかぎり基本関数におとされる。関数規則はこれらの関数間の関係を記述する規則で、

(equal(t1, t2) -(C)-> (equal(t1, t2) and P(f1,f2,.. fn)))
with C=(equal(t1,t2) => P(f1,f2,...,fn)) と
(equal(t1, t2) -(C)-> P(f1,f2,.. fn))
with C=(equal(t1, t2) <=> P(f1,f2,...,fn))

のようなものがある。ただし、B =>AとA <=> Bはそれぞれ AがBの必要、必要十分条件であることを表す。(B => A は同時にBがAの十分条件であることを表す)。Pはn項述語であり、fは関数である。上のルールはある条件の必要条件を追加する、あるいは必要十分条件に置き換えることにより、絞り込みの効果の向上を図っている。

以上のような規則は代数規則の先頭に付け加えられる。いいかえると、抽象データ型に関する条件、データ構造を全部展開してから代数規則に関するルールの処理を行う。代数規則と抽象データ型とのルール間の関係は明確である。

CAD応用上のソリッド・データベースに対する問い合わせを実例として示す。(簡単のため、2次元にした)

RANGE OF t IS Struct
RETRIEVE (t.Quad-tree)
WHERE
isomophic(t.Quad-tree, *const*(1 0 (0 1 0 0) (1 0 0 0)))

ただし、システム側にareaに関してインデクスが張ってあり、フィールドareaが関数*area*(Quad-tree)の値と同一であるものとする。この問い合わせに対して、前述の抽象データ型のルールが適用可能である。具体的には、条件*area*(t.Quad-tree)=*area*(1 0 (0 1 0 0) (1 0 0 0))が*isomophic*の必要条件であるので、関数規則によりWHERE文の条件に追加される。さらに属性規則によって、area上のインデクスを用いると、次の形になる

WHERE
isomophic(t.Quad-tree, *const*(1 0 (0 1 0 0) (1 0 0 0)))
and area=*area*(1 0 (0 1 0 0) (1 0 0 0))

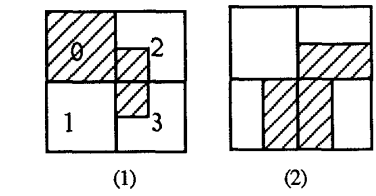
この結果は、*area*の値をキーとして、area上のインデクスを用いて検索することができる(図1)。

実現：

現在は拡張関係データベースGBASE上のLISPの環境でMODUS開発プロジェクトの一環としてこの機構を構築している(図2)。

おわりに:

従来の関係演算に対応するルールをもとにし、抽象



a).図形のQuad-treeによる表現

Struct

S-Id		Quad-tree
1	.	(1 0 (0 1 0 0) (1 0 0 0))
2	.	(0 (0 0 1 1) (0 1 0 1) (1 1 0 0))
(key)	.	... ...

b). a)に対応する表

ptr	area
.	6/16
.	6/16
.	.

c). システム側のインデクス

図1.実例

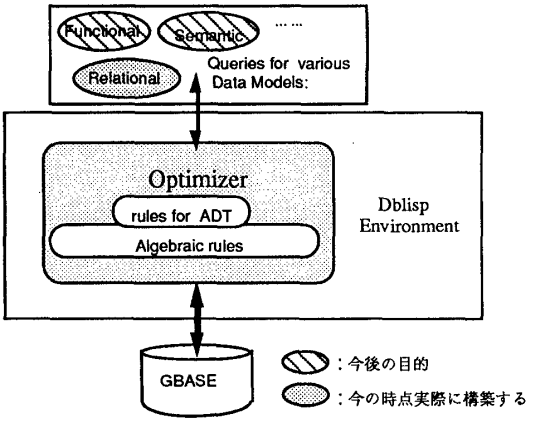


図2. システムの構成

データ型を扱うルールを加えて得られたルール・ベースに基づく最適化機構の設計を試みた。

現在は拡張関係データベース上で実現中である。将来の目的としては、他の色々なデータ・モデル(例えば、関数型モデル、セマンティク・モデル)への拡張を考えている(図2)。

参考文献：

[1]. D.S.Batory, et al : "GENESIS: An Extensible Database Management System." in *IEEE TRANSACTION ON SOFTWARING. VOL.14. NO.11.Novemble 1988.*

[2]. Goetz Graefe, David J. DeWitt : "The EXODUS Optimizer Generater." in *SIGMOD'87 PROCEEDING pp.160-171.*

[3]. Johann Christoph Freytag, et al : "A Rule-Based View of Query Optimization." in *SIGMOD'87 PROCEEDING pp.172-180.*