データベースプロセッサ ＲＩＮＤＡの最適化方式

5Q-8

中村仁之輔、板倉一郎、芳西　崇、黒岩淳一
ＮＴＴ情報通信処理研究所

１．はじめに

データベースプロセッサＲＩＮＤＡは、リレーショナルデータベース処理性能の向上をねらいとして開発したものであり、ハードウェアとして、①内容検索処理を行うＣＳＰ、②ソート、ふるい落とし処理を行うＲＯＰからなり、それらをデータベース管理プログラムで制御している。ＲＩＮＤＡの最適化は、従来のＲＤＢで採用されている、実行前にアクセスパスを決定する方式（静的最適化方式）に加えて、実行時に事前の処理結果を参照してアクセスパス決定を行う動的最適化方式を採用している。本稿では、ＲＩＮＤＡの動的最適化方式の内容を示す。

２．ＲＩＮＤＡ最適化

２．１　静的最適化方式の問題点

従来のＲＤＢで採用されている静的最適化方式は、アクセスする前に、すべてのアクセスパスを決定する方式であり、代表的な方式にインデックスの付与状態、条件式の種類、表内の行数等により、Ｉ／Ｏ回数を予測し、もっとも少ないＩ／Ｏ回数と想定されるアクセスパスを選択するコスト評価方式がある[1]。この方式では、検索結果の行数等は予測によって求められており、Ｉ／Ｏ回数が正確に見積もれないため、最適化によって最も処理時間の短いアクセスパスが選択されるとは限らないという問題があった。

２．２　ＲＩＮＤＡ処理方式の特徴

ＲＩＮＤＡの基本的関係演算処理は、ＣＳＰによる検索結果を一旦一時表上に保存し、次の演算処理ではその結果を入力として処理を続ける方式をとっている。結合処理時の処理方式の概要を図１に示す。

２．３　ＲＩＮＤＡ最適化の着眼点

２．２で示したＲＩＮＤＡの処理方式の特徴は、ＣＳＰ，ＲＯＰの検索結果の情報（行数等）を知ることができるため、次の処理に利用する情報を正確に知ることができる点である。この点に着目し、ＲＩＮＤＡでは、複数の表を操作する関係演算（結合処理、副問合せ処理）において、事前にアクセスした結果情報を利用して次のアクセスパスを決定する動的最適化方式を採用している。動的最適化により、正確な情報によるアクセスパス選択が可能となり、全体の処理時間短縮が実現できる。

以下、関係演算処理で処理時間の短縮が最も要求される結合処理について、動的最適化方式の内容を述べる。

３．ＲＩＮＤＡの結合処理方式

ＲＩＮＤＡの結合処理は、まず二つの表間で結合を行い、その結合結果と残りの表の内の一つの表との結合を行う。これを全ての表がなくなるまで繰り返す方法をとっている。

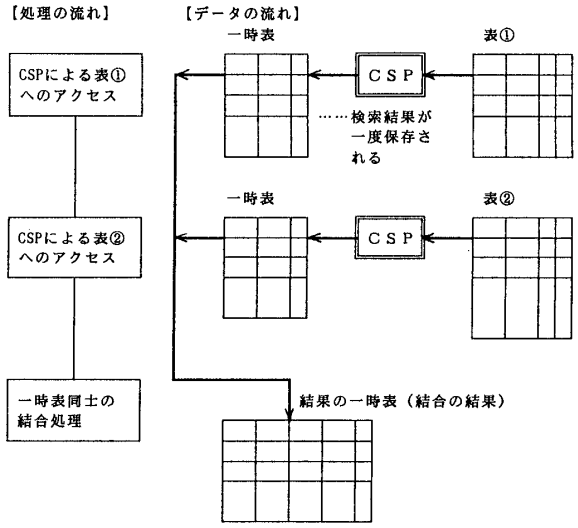

図１　ＲＩＮＤＡ処理方式の概要

二つの表間の結合処理は、一旦ＣＳＰによりアクセスされた結果の一時表に対して行い、以下の３つの方式のいずれかで行う。

（１）ネステッドループ結合方式

二つの一時表同士を、一方の表の一行に対し、もう一方の表のすべての行と比較し、それをすべての行が終了するまで続ける。

（２）ソートマージ結合方式

一般に知られているソートマージ結合方式のうち、ソート時に、ＲＯＰのふるい落とし機能を利用して結合対象行数を減少させてマージ結合を行う方式であり、以下の２種類の方式がある。

（２－１）片ハッシュソートマージ結合方式

一方の一時表（Ｔ１）の行すべてをＲＯＰにて設定ソート（ソートキー（ジョインキー）値に対しビットアレイの設定とソートを行う）し、その結果を一時表（Ｔ１１）に保存する。次にもう一方の一時表（Ｔ２）の行すべてをＲＯＰにて参照ソート（設定ソートで設定されたキーと同じものだけをソートして出力する）し、結果を一時表（Ｔ２１）に保存する[2]。最後にＴ１１，Ｔ２１の行をマージ結合する。

（２－２）両ハッシュソートマージ結合方式

一時表Ｔ１について、ＲＯＰにて結合キーの設定（ソートキー値に対しビットアレイの設定のみを行う）を行う。次に一時表Ｔ２について、ＲＯＰにて参照設定ソート（設定で設定されたキーと同じものだけをソートするとともに、新たにビットアレイの設定を行う）し、結果を一時表Ｔ２１に保存する。続いて、Ｔ１をＲＯＰにて参照ソートし、結果を一時表Ｔ１１に保存する。最後にＴ１１，Ｔ２１の行をマージ結合する。

ＲＩＮＤＡ－Relational Database Processor:Query Optimization
Jinnosuke NAKAMURA, Ichirou ITAKURA, Takashi HONISHI, Junichi KUROIWA
NTT Communications and Information Processing Laboratories

4．動的最適化方式

以下に示す手順で、アクセスパス決定および結合処理を行う。

【言語解析時】

検索結果の行数が少ないと想定される表の順に結合順序を決定し、１番目の表へのアクセスパスを決定する。

【実行時】

ＳＴＥＰ１：１番目の表にＣＳＰによりアクセスし、検索結果を一時表Ｔ１に保存する。

ＳＴＥＰ２：一時表の行数をみて、表１に従って２番目の表へのアクセスパスを決定する。

ＳＴＥＰ３：２番目の表へのアクセス結果を一時表Ｔ２に保存する。

ＳＴＥＰ４：Ｔ１とＴ２の行数、テーブルサイズにより、表２に従って結合方式を決定する。

ＳＴＥＰ５：ＳＴＥＰ４で決定された結合方式に従って結合処理を行い、結果を一時テーブルＴ３に保存する。

ＳＴＥＰ６：全ての表の結合が終われば処理を終了する。結合対象の表が残っていれば、次の結合対象の表をＴ２と、Ｔ３をＴ１とよみかえて、ＳＴＥＰ２にいく。

実行時の例を図２に示す。

表１　２番目の表へのアクセス方式決定法

一時表の行数	アクセス方式
1	他の検索条件に、結合キーに対する比較述語（列名　比較演算子　定数）を付加して表にアクセスする
ＩＮ述語内の定数数の制限以内	他の検索条件に、結合キーに対するＩＮ述語（列名　ＩＮ（定数，定数，・・・））を付加して表にアクセスする
上記以外	結合キーに対する条件を除いた条件で表にアクセスする

表２　結合方式判定法

判定条件	結合方式
C(T1)=1 OR C(T2)=1 OR (C(T1)*C(T2)⟨=n1 AND (S(T1)⟨=n2 AND S(T2)⟨=n2))	ネステッドループ結合方式
MIN(S(T1),S(T2))⟨O(ROP) AND (C(T1)/C(T2)⟨=n3 OR C(T2)/C(T1)⟨=n3)	片ハッシュソートマージ結合方式 ・C（T１）＞C（T２） 設定ソート：Ｔ２，参照ソート：Ｔ１ ・C（T１）≦C（T２） 設定ソート：Ｔ１，参照ソート：Ｔ２
上記以外	両ハッシュソートマージ結合方式 ・C（T１）＞C（T２） 設定：Ｔ２，参照設定ソート：Ｔ１，参照ソート：Ｔ２ ・C（T１）≦C（T２） 設定：Ｔ１，参照設定ソート：Ｔ２，参照ソート：Ｔ１

C(T1)：一時表T1の行数、C(T2)：一時表T2の行数、n1,n2,n3:定数
S(T1)：一時表T1のﾍﾟｰｼﾞ数、S(T2)：一時表T2のﾍﾟｰｼﾞ数
MIN(T1,T2):T1,T2のいずれか小さい方、O(ROP):ROPのｵｰﾊﾞﾌﾛｰ条件

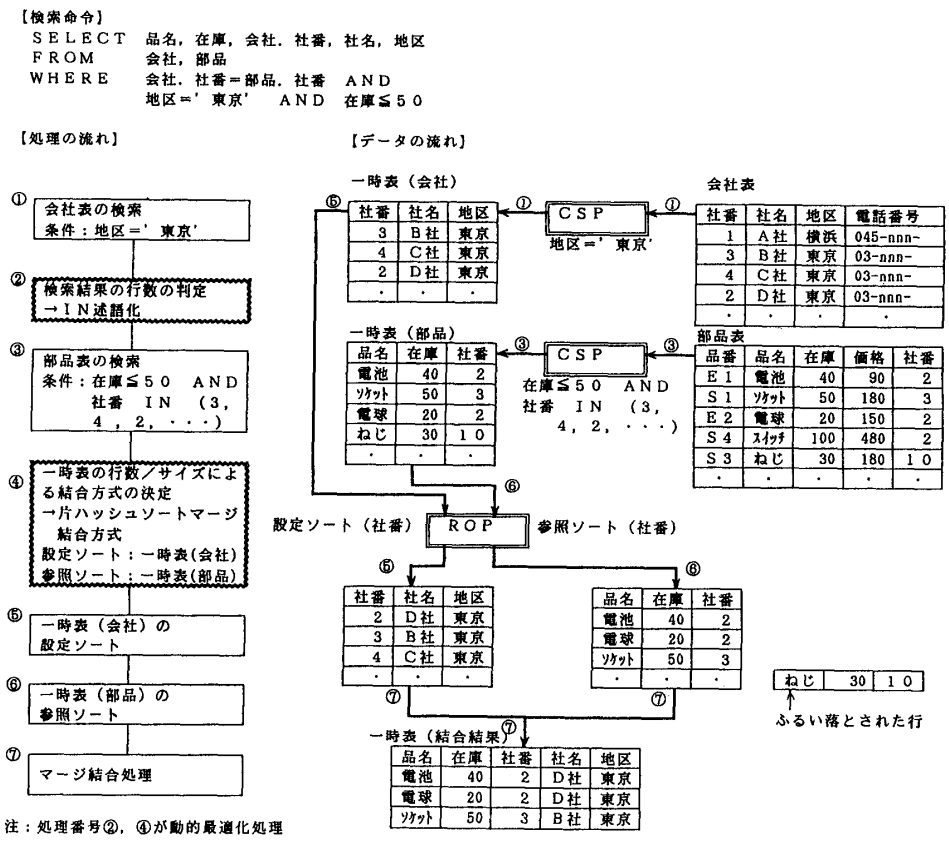

図２　結合処理の動的最適化の例

5．おわりに

ＲＩＮＤＡで採用した動的最適化方式の概略を示した。動的最適化方式は、実行時にアクセスパスの決定を行うため、言語解析時にアクセスパスを決定する方式に比べて、実行時のオーバヘッドは増える欠点はあるが、動的最適化の対象が大量データを扱う結合、副問合せ処理であり、全体の処理時間に比べれば動的最適化処理のオーバヘッドはほとんど無視できる。また、アクセスパスの決定を正確な情報に基づいて行っていることから、従来の方式に比べて大幅な性能向上が図れる。

【参考文献】

［１］M.M.Astrahan et al. :System R:Relational Approach to Database Management,ACM Transaction on Database Systems,Vol.1,No.2,June 1976, pp97-137

［２］武田他「データベースプロセッサＲＩＮＤＡの関係演算方式」、情処37全国大会、1988