データベースマシン　GRACE

＝二次記憶構造とJOIN＝

4P-1

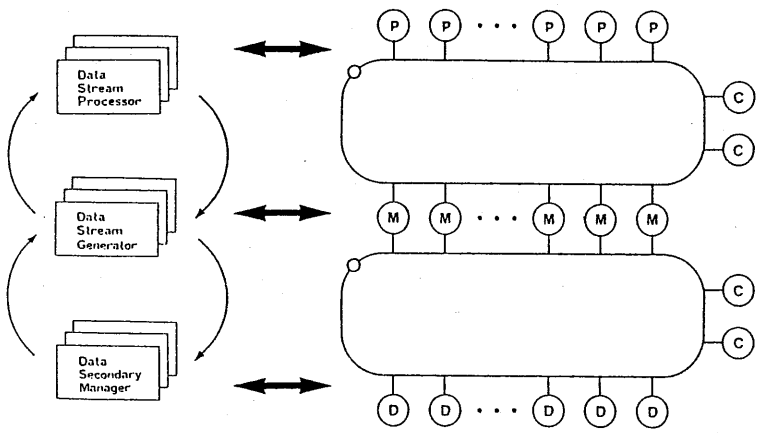
喜連川　優　　伏見　信也　　田中　英彦　　元岡　達

（東京大学　　工学部）

1.　はじめに

GRACE［1］はHash とSort に基づく処理技法により、Join 、Projection 等処理負荷の重い関係代数演算を高速に処理する事が出来、Join 負荷の重い環境に対しても高い性能が得られる点に特徴がある。図1，2に示される如く、GRACE は 3種類の構成要素 から成り、SDMにリレーションが格納されている。

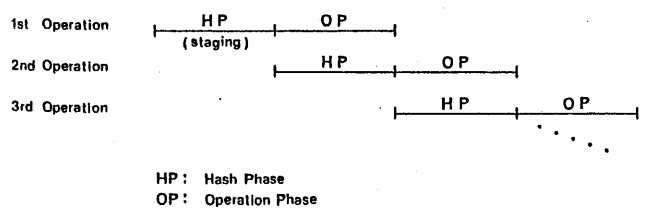
リレーションの格納形式は、問い合わせ処理に大きな影響を与えると考えられる。従来、多くのマシンでは1タプルを1レコードとする記憶構造を採用しているが、これに対してアトリビュト毎にリレーションを分割する構成も考えられる。今回は、GRACE上で従来からのタプル指向型編成に加え、アトリビュート指向型編成に対するJoin 処理方式について検討したので報告する。



P : Processing Module　C : Control Module
M : Memory Module　D : Disk Module

Fig.1 Abstract Architecture Of GRACE

Fig.2. Global Architecture Of Data Manipulation Subsystem In GRACE



HP : Hash Phase
OP : Operation Phase

Fig. 3 Operation Level Pipeline Processing
(Overlapped Processing Of Hash Phase & Operation Phase)

2.　二次記憶構造とJoin

2.1 タプル指向型ファイルとJoin

GRACEでのJoin は先ず2つのリレーションをそのJoin アトリビュートに関してHash を施し、互いに独立な幾つかのバケットに分割した後、各々のバケットをO（N）ハードウェアソータによりバケットシリアルに処理してゆくという “Hash と Sort を用いた関係代数処理アルゴリズム” により実現されている。この際、モジュールにわたってバケットのパイプライン処理がなされ、バンクパラレリズムを反映した処理が可能となっている。

1つの演算子の処理は、Hash による前処理と、Sort を用いた関係代数処理の2つに大別出来、多段の関係代数木として表わされる問い合せはこれらの処理を繰り返し行なう事により実現される事になる。　この際、リレーションをタプル毎に格納する従来のタプル指向型ファイルでは当該Join の処理対象タプル内に次Join のアトリビュートが連結されている為、次オペレーションのHash による前処理を当該オペレーションの関係代数処理に重畳化することが出来、これによって効率の良いJoin 処理が可能になっている（図3）。

2.2 アトリビュート指向型ファイルとJoin

リレーションをアトリビュート毎に分割し、各々にタプルID（Tid）を付加した形で格納するものとする。この様な構成ではアトリビュート間の対応関係はTidで示されているにすぎず、その結合は動的に行なう必要がある。即ち、1リレーション内で各アトリビュートの値は静的に結合されてはおらず、Tid を用いてJoin 操作を施さねばならない。図4に表わされる如き多くのJoin を含む問い合せ（リレーションR（i = 1, … ,5 ）に関してR_i とR_{i+1}(i = 1, …, 4) はA_{i+1}, A_i なるアトリビュート間で結合可能とする）の処理では、2.1と同様に処理を進めようと

すると、A_{i-1}からA_{i+1}を求める為にはTid Joinが毎回オーバヘッドとして加わる事になり、略倍のデータフローサイクル（DSGの発生したデータ流がDSPを通って処理され他のDSGに取り込まれるまでの1サイクル）を要する事になってしまう。

これに対して、値に関するJoinを先ず実行し、以後Tidの空間で処理を進めてゆく手法が考えられる。図5の例でも示される様に、始めにR_iとR_{i+1}のJoinを全て実行し、Tidペアを生成する。次にTidの空間でJoinを繰り返し、結果タプルに対応するTidペアを作る。そして最後に求められたTidのペアを用いて値の結合を行なう。以上3つの段階を経る事になる。

最初の値に関するJoinについては2.1と同様にアトリビュートの値にHashを施し、クラスタリングする事で従来同様の処理が可能であるが、アトリビュート毎に分割した構成では、各々値に関して静的にクラスタリングしておく事が可能であり、この値に関するJoinはマージ操作とする事が出来る。又、複数のリレーションにわたって同一ドメインを有するアトリビュートをまとめて1つのファイルとする事も可能であり（Outer Join）、この場合にはファイルの操作時間は少々増加するものの、処理は一層簡単になる。（この処理機能をSDMに持たせる事は容易である。）

次にTid空間でのJoin処理を行なうが、これは従来の処理方式と同様にして効率良く実現出来る。何回かのTid Joinを繰り返す事により、ターゲットアトリビュートを有するリレーションのTidからなる対 Tid $=(Tid_1, Tid_2, \cdots\cdots, Tid_n)$が生成される。

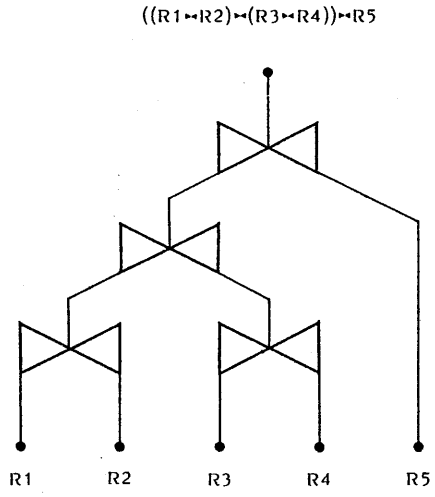


Fig.4 An Example of Relational Algebra Tree with 5 Joins

最後に結果リレーションを構成するアトリビュートに関してその値を結合する。生成されたTidペアはその要素である各々のリレーションに関し必要とするアトリビュートを結合した後、Tid 全体にクラスタリングを施し、多項のJoinを行なう事によりターゲットアトリビュートの値を一度に結合する事が可能である。

この様に、GRACEはTidと値からなるアトリビュート指向型のファイルに対しても効率の良いJoinの処理が可能である事がわかる。ProjectionやRestriction処理はアトリビュート間の対応が必須となるが、これは結果タプルに関する値の結合処理に埋め込む事が可能である。

3. おわりに

データベースマシンに於ける二次記憶系の物理的構造としての最適なファイル構成、リレーションのページ分割は、当該リレーションの値の分布、問い合わせの性質に依存すると考えられ、種々のファイル構成に対して柔軟に対処可能なアーキテクチャが望まれる。GRACEは、タプル指向型のファイルに対しても、又アトリビュート指向型のファイルに対してもデータの流れに沿った効率の良い処理の出来る事が判った。今後より詳細な検討を進めてゆく予定である。

［参考文献］
［1］喜連川 優，田中 英彦，元岡 達
「Relational Algebra Machine: GRACE」
京都数理解析研講究録 SSF 1982

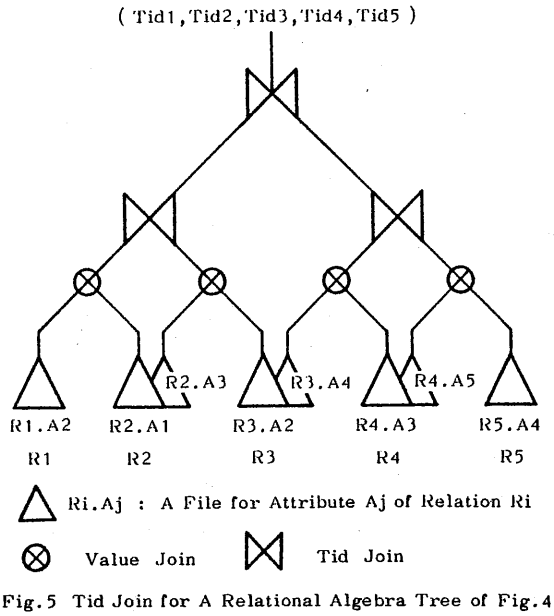


Fig.5 Tid Join for A Relational Algebra Tree of Fig.4