データベース・システム 47－3
（1985・5・20）

関係型均質分散データベースシステム　RDB／DV

手塚正義　安達進　山根康男　中田輝生　武理一郎　岡崎卓
（富士通研究所）

1．　はじめに

1970年代後半から本格化してきた分散データベース管理システムの研究開発[1]は理論研究とSDD-1 以来のシステム開発技術の積み上げによって，近年System R*[2]を代表格とする均質分散システムとDDTS, Multibaseなどの異種分散システムの２方向に研究開発が定着してきた感がある。どのシステムも分散データベースが本来抱える複雑多岐の問題の総てを解決してはいないが，性能・信頼性・相互利用性などの本質的課題について種々の解決法を提案してきている。筆者達もSystemR*と同様，集中型関係データベースの分散化を図るための技術課題に焦点を絞り，トランザクション管理の分散制御と並列実行を目的としたシステムアーキテクチャ及び分散カタログ管理，問い合わせ処理のグローバルな最適化，同時実行制御などの機能を備えた基本システム RDB/DV (Relational Database System/ Distributed Version)の開発を行ってきた。このたび試作システムの一応の完成をみたのでここにシステムの基本アーキテクチャと各機能の処理方式について報告する。なお，現在システムは Mシリーズ大型コンピュータ同士の回線接続によるネットワーク上で稼働している。まだグローバルなリカバリ，重複データ処理，データの分割分散と統合ビューの機能は未完成である。

2．　設計方針

ＲＤＢ／ＤＶの設計目標は関係データベースの均質分散を前提として，ユーザが集中型データベース管理システムを使うのと同様な使い易さを提供出来るシステムとすることである。使いやすい分散データベースシステムの基本的な要件は以下のものであろう。

（１）アプリケーションに対するシングルサイト・単一データベースイメージのサポート
（２）サイト間の通信を伴うグローバル処理のレスポンスが従来の集中型処理のレスポンスと最小の通信遅延時間の和程度の高い処理効率性
（３）負荷分散，危険分散をねらうデータベースのトップダウン的分散化と既存データベースの相互利用を目的とするボトムアップ的統合化の両方に適用出来る汎用性

(1) は分散データベースシステムの主目的である位置透過性の実現である。本システムでは，分散データベース上での関係型非手続き言語 (Relational Query Language) のサポート，同時実行制御，機密保護，リカバリなどの基本的な分散制御技術の開発を第一の目標としている。(2) に関しては如何なる高速な通信路を用いても通信遅延時間は存在するので，複数サイトが関係するグローバル処理に対し，分散制御と並列実行方式，通信量の最小化を図るグローバル最適化手法などによって，出来る限り高い応答速度を達成することである。(3) はシステムＲ* の設計目標の１つに挙げられたサイトの自治権[3]の確立を前提とした上で，システムの動的な参入と離脱の容易性，データの重複／分割分散とそれらの統合化機能，異種データベース

の統合性，データの最適配置方法などデータベースの運用と深くかかわった未解決の問題を含んでいる。本システムでは異種問題は対象外とする。少なくとも (1)と(2) が実現できれば，実用化に向けた汎用分散データベースシステムの第一歩といってよいであろう。

3．基本アーキテクチャ

ＲＤＢ／ＤＶのシステム構成を図１に示す。各サイトのシステムは同一で，グローバル実行サブシステム(GES)，ローカル実行サブシステム(LES)，分散アクセス制御サブシステム(DACS)，の３つから構成される。GES は複数サイトのデータに対する処理要求を含むアプリケーションを受け付け，自サイト及び他サイトにデータ処理要求を出し処理全体の実行を制御する。LES は自サイトの処理要求を受け付けてデータベースをアクセスし検索・更新処理を行う。DACSは処理要求と転送データ等のメッセージ通信，他サイトからの処理要求に対する実行制御，他サイトの辞書情報の管理，グローバルな排他制御などの機能を持つ。これらのサブシステムの機能構成を図２に示す。LES は集中型データベース管理システムＲＤＢ／Ｖ１[4]とほぼ同様と見なしてよい。ネットワーク形態は論理的に，全サイトが相互に通信可能な対等型ネットワークを基本としており，広域網・LAN 上に容易に適用できる様なシステム・アーキテクチャとなっている。

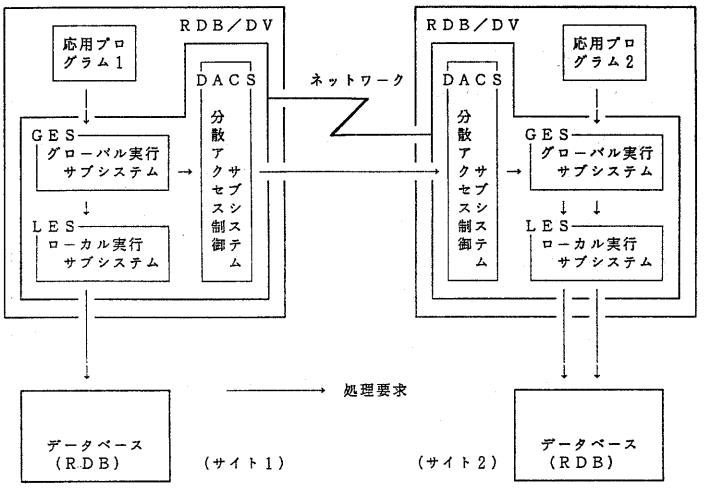


図　１　　システム構成

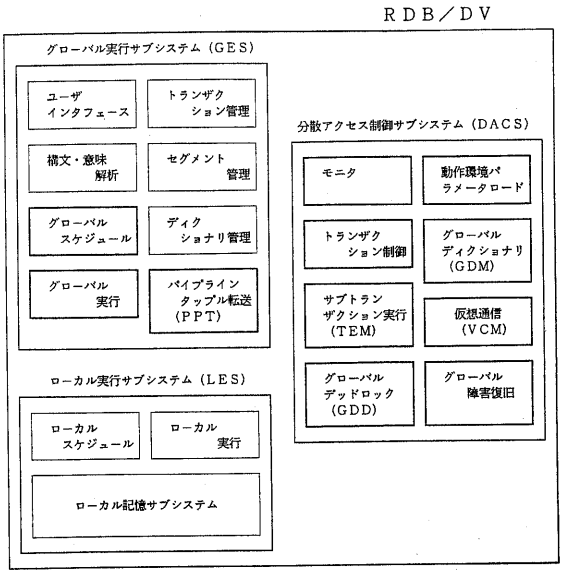
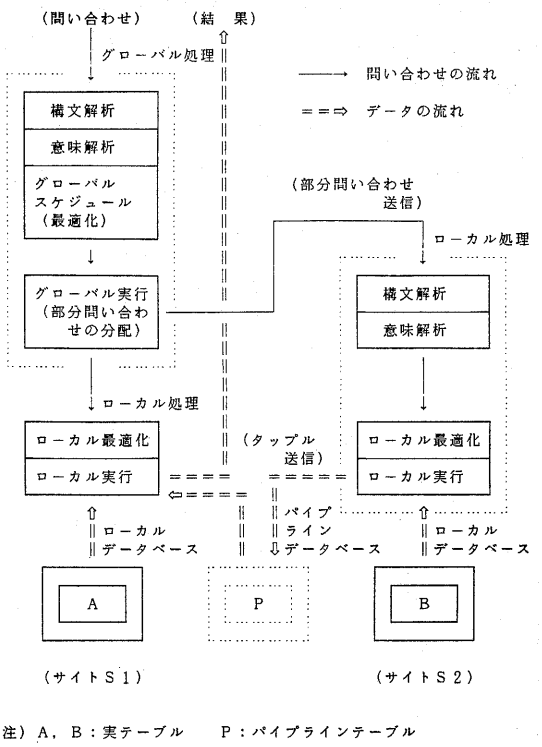


図　２　　機　能　構　成

3．1　問い合わせ処理方式[5]

問い合わせは図３に示す様に処理される。GES のユーザインタフェースで関係型非手続き言語RDB/QLの各種コマンドを受付け，構文

・意味解析部がコマンドを構文解析後，含まれるテーブルの属性情報をデータベースの辞書をアクセスして取り出す。グローバルスケジュールは複数サイトのデータアクセスを必要とする問い合わせをサイト全体の処理コストが最小になる様に，グローバル最適化を行い，各サイト毎の部分問い合わせに分解する。グローバル実行は部分問い合わせを各サイトに分配し，実行開始と終了処理を行う。複数サイトにまたがる処理（グローバルトランザクション）をシングルサイトイメージで処理するためにトランザクションとデータベースアクセスに関する管理機能を用意している。前者はトランザクションの開始・終了処理を行う。後者の主な機能としてセグメント（データベースと同義）やテーブル・インデックスのオープン／クローズ，サイト間のデータの授受を行うパイプラインタップル転送（PPT ）機能等がある。分解された部分問い合わせはQL言語のコマンドであり，LESが受け取る。LES はGES 同様，コマンドの構文・意味解析を行った後，ローカルスケジュールでインデックス等のアクセスパス選択と実行計画を立て，その処理手順に従いローカルに実行する。データベースアクセスは自サイトの場合，ローカル記憶サブシステムが行い，他サイトに対しては，PPT 機能を用いている。パイプラインテーブルはサイト間のデータ授受をサイト内の仮想的なテーブルへのデータアクセス（タップルの挿入・取り出し）に見せかけるとともに，1つの問い合わせを分解し複数のサイトに分配された部分問い合わせを並列的に実行することも意味している。



注）A，B：実テーブル　　P：パイプラインテーブル

図３　　問い合わせ処理方式

3.2　トランザクション実行方式[6)]

グローバルトランザクション（Tr）の実行方式を図４に示す。

Tr発生サイトのGES はTr開始時に各Trに対し，識別子（Trid）を与える。Tridは＜タイムスタンプ，発生サイトの識別名＞の組で構成され，グローバルな一意性を持つ。１つのTrはグローバルスケジュールで，サイト毎の処理（サブトランザクション: Str）に分解され各Str の実行サイトが決定される。一方，各サイトでは他サイトから送られてくるStr を実行するために，DACS中のサブトランザクション実行（TEM ）が前もって複数個起動されており，Str を受取る毎に１個のTEM が順次割当てられる。TEM はそのサイトのGES を使ってStr の開始，実行，終了処理を行う。このように１個のTr発生サイト（マスタサイト）と複数個のStr 実行サイト（スレーブサイト）が協調して１つのTrを並行処理する。

Tr実行時のサイト間のメッセージの流れを図５に，その意味を表１に示す。Trの開始時にマスタサイトからスレーブサイトにStr 開始要求（BEGIN ）が送られると，Str を実行するTEMが動的に割当てられる。BEGIN にはStr 実行に必要なサイト間のメッセージ通信のための宛名（メッセージキューノード: MQN ）が付与される。MQN は＜メッセージ受信サイトの識別名，Trid＞の組で与えられる。スレーブサイトでは届いたBEGIN のMQN からStr 毎のMQN を開設する。この方式はTEM の動的割当てにもかかわらずStr の非同期開始を可能にしている。つまりあるTrに属する複数のStr を協調して実行する場合，（例えばサイトにまたがるジョイン処理等）サイト間の同期開始方式に比べ同期メッセージ及び同期のための待ち時間を不要とするので効率が良い。

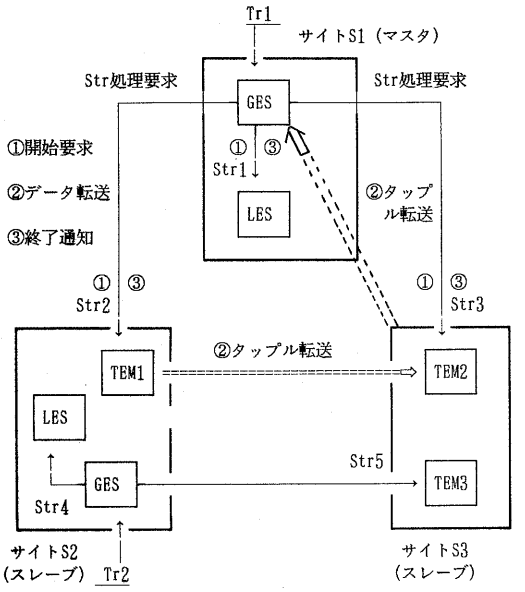


図４　トランザクション実行方式

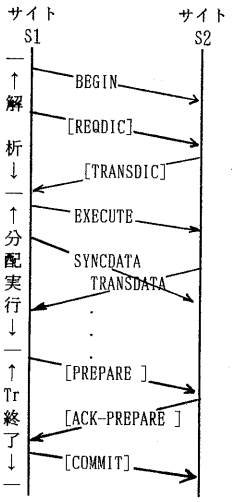


図５　メッセージフロー

制御	BEGIN	Str 開始要求
	PREPARE	コミット準備要求
	ACK-PREPARE	コミット準備完了通知
	COMIT	コミット要求
	ABORT	アボート要求
辞書	REQDIC	辞書情報転送要求
	TRANSDIC	辞書情報転送
実行	EXECUTE	部分問い合わせ実行要求
	SYNCDATA	タップル転送同期要求
	TRANSDATA	タップル転送

表１　メッセージの意味

マスタサイトではTrの解析に必要な辞書情報をグローバルディクショナリ機能（GDM ）を通して得る。GDM は辞書情報の転送回数を減らすため，メモリ内のキャシングを行っている。分解された部分問い合わせは実行要求（EXECUTE ）によってスレーブサイトのTEM に通知され，各サイトから実行結果であるタップルデータが返される。（TRANSDATA ）　タップル転送は転送パスに対応するパイプラインテーブルによって並列実行を基本として実現される。その機構はタップルの送受信操作を実テーブルの挿入・スキャン操作と同様に見せかけている。ここで転送バッファ領域のオーバフローを避けるために同期メッセージ（SYNCDATA）を使う。またタップルはブロック単位で転送するがバッファ領域が大きい場合には複数ブロックを受信側バッファ領域に蓄積できるSYNCDATA を複数ブロック毎　に送ることによってメッセージ数を減らしている。

Trの終了処理では分散データベースの同時更新による一貫性を保つために，２フェーズコミットメント（2PC ）方式を採用して，Trに関わるStr の終了処理の同期化を図っている。これはマスタからスレーブへのコミット準備要求（PREPARE ），スレーブからの準備完了通知（ACK-PREPARE ）及びTrの正常終了（COMMIT）又は異常終了（ABORT ）要求のメッセージで行う。終了処理以前にマスタが何かのエラーを検出した場合，直ちにTrのリストア処理を行い関連スレーブにABORT 要求を出しStr のリストア処理を行う。この方式ではスレーブはACK-PREPAREを出してからCOMIT を受け取るまでの間にマスタがシステムダウンを起すとStr を正常終了又は異常終了すべきかの判定ができない（ブロッキング）状態に陥るので，この期間を短縮するために，COMIT の直前に再度同期確認メッセージの授受を行う拡張方式も付加している。当然ながら全てのStr が検索処理の場合はCOMIT だけを送り他を省略しメッセージ数を減らしている。

４．　カタログ管理方式

本システムのＤＤ／Ｄ（ Database Dictionary/Directory ）は完全に分散している。セグメントおよびオブジェクト（テーブル，インデックス等）の情報は夫々，ディレクトリとディクショナリに分けて管理する。セグメントは表２に示す様に４種類ある。ベースセグメントは自サイト内に実在する。他サイトセグメントは自サイトに存在しないが，ベースセグメント同様に扱える様，仮想化するものである。

名　称	実在性	共用性	用　途
ベースセグメント	○	○	グローバル・ローカル両用
テンポラリセグメント	○	X	作業用，一時保存用
他サイトセグメント	X	○	グローバル専用
パイプラインセグメント	X	X	タップル転送用

表２　　セグメントの種類

システムディレクトリのエントリ ::=

＜グローバルセグメント名，マスタファィル名，ボリューム名，... ＞

ユーザディレクトリのエントリ ::=

＜ユーザセグメント名＝グローバルセグメント名　SITE（サイト識別名）
マスタファィル名＞

(注) グローバルセグメント名はシステムディレクトリ内で，ユーザセグメント名はユーザディレクトリ内で夫々，ユニークである。

図６　　ディレクトリの記述

ディレクトリはシステム用とユーザ用の２種類を用意している。システムディレクトリは各サイトのベースセグメントの中で，他サイトからのアクセスが可能なものに対してその論理的名称（グローバルセグメント名）と物理的なマスタファイルの名前の対応表である。一方，ユーザディレクトリは応用プログラムが使用するセグメントに関する定義表で，プログラム内で使用する名称（ユーザセグメント名）と前述のグローバルセグメントまたはマスタファイルの名前を対応付ける。（図６参照）本方式によれば，データベースをローカル使用からグローバル使用に切り換えるときシステムディレクトリにグローバルセグメント名を付けて登録すればよい。ユーザセグメント名またはマスタファイル名の変更も夫々，プログラム単位，サイト単位に独立に行なえるので各種変更に対する柔軟性がある。システムディレクトリはサイト毎に１個用意し，システムがその起動時に自動的に読み込み，環境設定を行う。一方，ユーザディレクトリは応用プログラムの実行時に，他の動作パラメータと共にオプションファイルのデータとして与える。

オブジェクトの属性情報を格納するディクショナリ（データ辞書）はパイプラインセグメント以外のものは，オブジェクトが存在するサイトのセグメント中にハッシュ編成のテーブル形式で分散して置かれている。グローバルトランザクションの解析で必要となる他サイトのオブジェクトの辞書情報は，メモリキャッシュに無い場合に他サイトに要求する。（REQDICメッセージ）また，パイプラインテーブルの辞書情報は参照（受信）側サイトで必要とするが，その情報は部分問い合わせの生成と同時に得られるので，その付属情報として付加し実行サイトに送っている。（EXECUTE メッセージ）辞書情報の構造と内容は集中型RDB と同一であり，LESは他サイトのテーブル，パイプラインテーブル等の仮想オブジェクトを特に意識することなく辞書を利用できる。また，辞書の中には統計情報（タップル数，ジョインキーの種類など）も含まれており，GES のグローバル最適化のパラメータとして利用できる。

5．　グローバル最適化方式

上述の問い合わせ処理方式では図７に示す様に，複数サイトに分散するテーブル同士のジョイン処理を分解し，グローバルな処理量が最小になるように実行計画をたてる。（グローバル最適化）この計画に従ってパイプラインテーブルを用いた部分問い合わせが生成される。

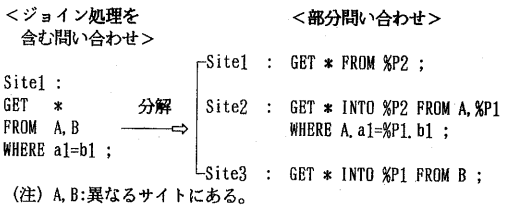


図７　　問い合わせ分解例

5．1　最適化処理手順[7)]

最適化の目標はサイト間のデータ転送量とサイト内の処理量の総和を最小にする処理手順を探し出すことである。本方式はデータ転送量を減少させる効果のあるセミジョイン処理を基本とし，真に効果的なセミジョインを選択するための改良を施している。その処理手順を図８に示す。

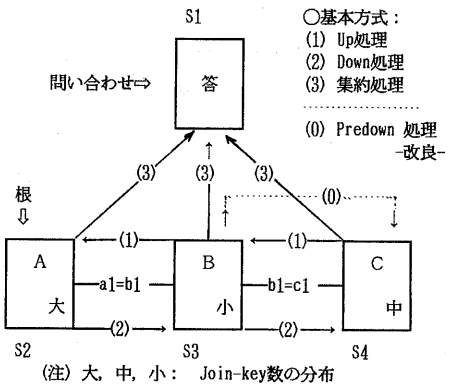


図８　　最適化処理手順

①根の決定：ジョインキーの数（又はタップル数）最大のテーブルに着目する。

②基本方式の適用：

ジョイン関係を示す問い合わせグラフに基づき根に向かうUp処理，その逆向きのDown処理，各サイトのセミジョイン結果の部分解を答のサイトに集める集約処理からなる有向グラフを作る。

③UP処理に先立ち，ジョインキー数の大小関係から逆向きのPredown 処理を追加する。

④Predown → UP → Down →集約処理の順に次の手続きにより処理法を決める。

ｉ）各処理の有向枝集合の中でセミジョイン効果最大（テーブル同士のジョインキー数の比：縮小率が最小）の枝から順に選択する。

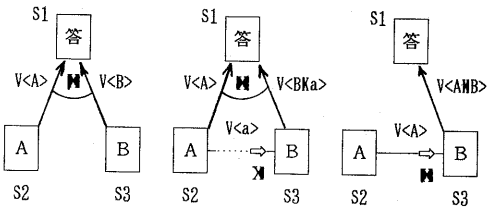
ii）選択枝について5.2 のコスト計算法により省略，セミジョイン，ジョイン化のいずれかの処理を決定する。

iii）一旦選択・決定した枝の処理法は変更しない。（バックトラックは無い）

5．2　コスト計算法

コスト計算は本来通信コストとサイト内処理コストの両方を見積るべきであるが，後者はシステム自身の性能とアクセスパスの選択に強く影響を受け，厳密に見積ることは困難である。またジョイン対象テーブルの数（サイト数）の増加に伴い，セミジョインを含めた処理手順の総数は指数オーダで増え，最適解の完全な探索は却って最適化の処理コストを増大させ，システムの性能を落しかねない。

本方式では上記処理手順④で縮小効果の大きい枝から優先的に選ぶことによって，極力効果の無い枝を刈る。またコスト計算は各枝の処理法の決定基準としてだけ使う。図９に１つの有向枝に対する３つの処理法を，表３に各処理に対するコスト計算式を示す。



(注) V: Volume of data, AMB: Join, BKa: Semi-Join
(a) 省略　(b) セミジョイン　(c) ジョイン化

図9　3つの処理方法（ジョイン条件：A.a=B.b ）

処理方法	コスト計算式	転送回数	処理量
省　略	(2CO+C1)V<B>	2	Join 1
セミジョイン	(2CO+C1)(V<a>+ V<BKa>)+C2	3	Semi 1 Join 1
ジョイン化	(2CO+C1)V<AMB>	2	Join 1

(注) 時間換算係数　CO: パイプラインテーブルの入出力処理，C1: 転送データ量，C2: 転送回数

表３　コスト計算式

6．同時実行制御方式

複数サイト・マルチユーザ環境でのデータ更新の秩序を正しく保つためには，グローバルな排他制御とマスタ・スレーブサイト間の同期更新機能が不可欠である。後者は3.2で述べた様に2PC方式で実現している。グローバルな排他制御方式は各サイトのローカルロック管理（LLM:Local Lock Manager）が個々に行う分散ロック方式を基本とし，GDD（Global Deadlock Detector）がサポートする複数サイトにまたがるデッド

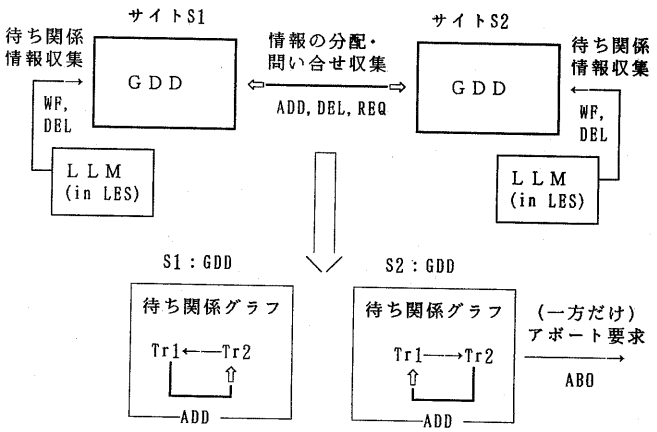


図１０　グローバルデッドロック検出方式

ロックの検出・解消方式[8]を組合せている。本GDD 方式は図10に示す様に，LLM から逐次通知されるTrの待ち関係情報を各サイトのGDD 同士がある時間間隔に送り合って待ち関係グラフを構成し，ループ検出を行いループ中のあるTrの中断（アボート）を要求するものである。

デッドロック検出処理手順

①分配処理

GDD はLLM から得たトランザクションT の待ち関係をサイト関数S(T)で決るサイトに送る。つまりT1→T2の情報はS(T1),S(T2) へ通知される。結果としてT に関する情報は総てサイトS(T)に集まる。

②ループ検出処理

GDD は収集された待ち関係グラフ中で責任ループ及び責任パスの検出を行う。責任ループとはそのループを構成するTrに対する評価関数W(Ti)を最大とするTiがそのサイトのTrである場合を云う。また責任パスとはパスT1→…→Tnに対してW(Ti)が最大となるTiがそのサイトのTrで，かつ両端のT1,Tn がそのサイトのTrでないときを云う。責任パスがあるとき責任ループの可能性があるのでTnに関する情報をサイトS(Tn) に問い合わせる。（REQ メッセージ）S(Tn) サイトはTnに関する情報があれば問い合わせ元に返答する。（ADD メッセージ）

責任ループがあるとき，ループ内のTrを１つ選び，アボート要求（ABO メッセージ）を出す。ABO 要求はTr発生サイトのTr管理に通知される。その Tr 管理が直ちに関連するStr 実行サイトにABO 要求を出す。

本方式の特徴はサイト関数S(T)を適当に選ぶことにより完全な分散制御から集中制御まで可変であり，グローバルロック検出の最適負荷配分及び情報の分配・問い合わせに必要なメッセージ数の削減ができることである。また評価関数W(T) によって１つのループの構成サイトを１個に制限し，複数サイトでの重複した検出を避けることができる。

７．まとめ

汎用的な分散データベースの出現を期待する声は強まりつつあるが，均質分散システムにおいては高速な通信路及び高速なデータベースエンジンに支えられてこそ集中型トランザクション処理の性能に見合うシステムの実現が期待できるであろう。

本システムは集中型RDB システムの分散化による汎用的な分散データベースシステムの実現可能性を追求しており，本報告の最適化，並列実行，同時実行制御に関する方式等は充分，実用に役立つものと確信する。今後の課題として，本システムの性能評価，分散リカバリ方式の開発，LAN へのシステム適用検討などがある。また，新規に異種分散システムへの展開を目指している。

参考文献

1)増永：米国における最近の分散型関係データベースシステム技術，情報処理，Vol.25 No.5 pp. 443-450
2)Williams, et al.: R*: An Overview of the Architecture, Proc. Int. Conf. on Database Systems, pp. 1-27 (1982)
3)Lindsay, et al.: Site Autonomy Issues in R*, Reseach Report, RJ2927, IBM Reseach Lab. (1980)
4)牧之内，他：関係データベース管理システム RDB/V1,情報処理論文誌，Vol.24 No.1 pp. 47-55 (1983)
5)安達，他：分散データベースシステム RDB/DV,情報全大26回 (1983)
6)安達，他：分散データベースシステム RDB/DV の試作，情報全大29回 (1984)
7)中田，他：RDB/DVにおけるジョイン処理方式，情報全大29回 (1984)
8)武，他：最適負荷配分が可能な分散型グローバルデッドロック検出方式，情報全大31回投稿予定