データベース・システム 36－3
（1983. 7. 14）

米国における分散データベースシステム研究開発の現状

増永良文
（図書館情報大学）

筆者は米国カリフォルニア州サンホゼ（San Jose）市にあるIBMサンホゼ研究所の客員研究員として，昨年4月より今年3月までの一年間，分散型関係データベース管理システムR*（アールスターと発音）のプロジェクト員として，その研究開発に従事する機会に恵まれた。本報告はこの体験を通して見聞した，米国における分散データベースシステムの研究開発の現状を記したものである。

1. 緒言

分散型データベース管理システム（以下簡単に分散データベースシステムともいう）は1970年代後半に入り，急激に関心が高まるとともに，幾つかの研究開発例をみるようになった。その原因としては次の項目を挙げられよう。

- 分散向きといわれた関係データベース管理システムがシングルサイトで実動しだし，分散データベースシステム開発の技術的裏付けをえた。
- ARPANETに代表されるような計算機ネットワーク技術が十分に固まり，分散データベースシステム開発の基盤を固めた。
- データを意図的に複数サイトに分散させ，システムのavailabilityを向上させようとするコンピュータアーキテクチャ側からの要求が高まった。
- 分散データベースシステムを構築して，生産性を向上させたいという企業の要求が高まった。

現在迄，米国では約10システム程，分散データベースシステムが研究開発されている。なかには商品化されているものもある。それらを大体年代順にリストアップすると次のとおりである。

- SDD-1（CCA社，マサチューセッツ州）
- 分散型INGRES（カリフォルニア大学バークレー分校）
- System R*（IBMサンホゼ研究所，カリフォルニア州）
- ENCOMPASS（タンデムコンピュータ社，カリフォルニア州）
- DDTS（ハネウェル共同計算機科学センタ，ミネソタ州）
- プライムコンピュータ社のシステム（マサチューセッツ州）
- Multibase（CCA社，マサチューセッツ州）
- DDM（CCA社，マサチューセッツ州）

本稿では，次章でこれらシステムの研究開発状況，特徴等を要約する。System R*は報告者がそのプロジェクトの一員として研究開発に従事したこともあり，章を改め，若干詳しくその設計方針，アーキテクチャを述べる。

2. サーベイ

表－1に前出のシステムの特徴を概略する。システム開発において，設計でインプリメンテーションは時間の経過とともに変るのが普通である。なるべく表

に記載の事項が事実と異なるないよう，データはシステムの研究開発の最終あるいは最新のレポートからピックアップするよう努めた。以下表－1を補うべくシステム毎に特徴等を記す。

● SDD－1[1]：汎用分散型関係データベースシステムのプロトタイプであった。同時制御（concurrency control），分散型質問処理，信頼性等の技術課題で貢献した。同時制御のためのコンフリクト（conflict）グラフ解析のメカニズムとタイムスタンプに基づいた同期プロトコルを導入した。コンフリクトグラフ解析はロッキング（locking）よりもトランザクションの直列化可能性（serializability）が優れている。分散型質問処理では準結合（semi-join）演算を導入した。SDD－1では質問処理の最適化は，それを行なうために必要なサイト間のデータ転送量（バイト）と定義したので，アクセスプランニングの問題はトランザクションとデータベースの状態が与えられたもとでのデータ転送量最小の準結合演算列をプログラムすることとなった。トランザクション管理には2フェイスコミットの一種を使っている。このプロジェクトはソフト寿命に終結した。

● 分散型INGRES[2]：非分散型関係データベース管理システムINGRESの分散版である。汎用システムを目指している。同時制御，分散型質問処理，パフォーマンスの測定，解析等を主要技術課題にしている。分散型質問処理の最適化は応答時間最小と通信量最小である。質問処理にはINGRESで用いられた質問分解（query decomposition）の手法が拡張されている。トランザクション管理は2フェイスコミット方式，重複データの制御のために主コピーサイトという概念を使った2フェイスロック方式を採用している。現在パフォーマンスの評価のため種々のベンチマークテストを実施中である。分散型INGRESをシングルサイトで使うと，非分散型INGRESに比べて，ローカルデータベース処理で20％程遅くなると報告されている。今後非対称なワークロード（workload）がパフォーマンスの劣下に与える影響の測定，種々の同時制御法の評価，等を行なうのベンチマークテストを計画している模様である。

● System R*：サイトの自治権の確立を最大の目標として，システムの開発を行なっている。非分散型関係データベース管理システムとして開発されたSystem R（のちにSQL/DSとして商品化）の分散版である。汎用システムを目指している。現在3台のIBMプロセッサー上で実動しており，インプリメンテーションは最終段階にきている。システムアーキテクチャの説明は次章に譲る。

● ENCOMPASS[3]：タンデムNonStopコンピュータシステムのデータベース部として開発された分散型データベース管理システムである。レポートジェネレータ機能と関係データベース質問処理機能の融合を計っている点に特徴がある。ENCOMPASSはENSCRIBE，TMF，DDL，ENFORM，PATHWAY等のサブシステムからなる。ENSCRIBEはファイルシステムであり，データへのアクセスや分散時のロケーショントランスパーレンシー等を提供する。TMFはトランザクション管理システムである。DDLはデータディクショナリシステムである。ENFORMは上に述べたOAとDBの融合機能を提供している。これはコンパイラ/レポートライター（フロントエンド）と質問プロセサー（バックエンド）の二つのプロセサーから

なる。コンパイラ/レポートライタは処理要求の発生したサイトで起動されるが，質問プロセサーは質問処理のためのデータアクセスが最も重いサイトで起動され，ネットワーク上のデータ転送量を最小化しようとしている。ENFORMはユーザがその目的のために使用するユーザ言語の名前でもある。

● DDTS[4]: DDTSはDistributed Database Testbed Systemの略である。命名から判るように"testbed"として開発されている。プロトタイプは効率とシンプリシティを強調するのに対し，テストベッドはモジュラリティとフレキシビリティを強調する。種々の設計パラメータ(たとえば同時制御法)がパフォーマンスにどのような影響を与えるかを"量的"につかむことを目的としている。6層のアーキテクチャをとり，ローカルデータベースはCODASYL型のZDS/II，その上に関係モデルのローカルとグローバルの表現(シンタクティック)スキーマがのり，その上にECR(Entity-Category-Relationship)モデルで記述されたグローバル概念(セマンティック)スキーマがある。この アクセス言語はGORDAS(Graph ORiented DAta Selection language)と呼ばれており，現在この段階までしかインプリされているようである。

● プライムコンピュータ社の分散データベースシステム[5]: プライム社はCODASYLタイプの非分散型データベース管理システムを所有しており，それをPrimenetで結合，分散データベースシステムを開発しようとするものである。特に多バージョン(multi-version)分散読み出し専用トランザクションの管理技術の確立を課題とした。現在，インプリメンテーションはほぼ終了したと報告されている。分散質問処理の最適化等は未だ研究されていないようである。

● Multibase[1]: 異種(heterogeneous)分散データベースシステムである。一般にデータモデルの異なるローカルスキーマを共通データモデルにスキーマ変換し，それらを統合するというアーキテクチアをとっている。共通モデルにFunctional Data Model(FDM)を採用している。統合スキーマ上にグローバルスキーマが，質問言語DAPLEXのビュー定義機能を用いて定義され，ユーザはそことインターラクトする。したがってINGRESが導入した質問変形(query modification)の手法が質問処理に適用可能で，実際そうしている。1982年初めに一応のインプリメンテーションが終了したと報告されている。

● DDM[6]: データベース操作言語DAPLEXをプログラミング言語Adaに埋込んだADAPLEXをサポートできる汎用の分散型データベース管理システムDDM(Distributed Database Manager)として開発されている。DDMは機能的にはSDD-1やSystem R*を手本にして設計されたが，ADAPLEXのサポート，重複データの取扱い，読み出し専用トランザクションの最適化，障害時回復のアルゴリズム等で特色をあすという。システムをAdaでインプリメントしており，システムのポータビリティが狙いという。現在進行中のプロジェクトである。

システムは開発しているが，例えば分散型質問処理法，同時制御法等の理論を研究している研究教育機関は多くあり，例えばスタンフォード大学，ベル研究

所，ミシガン大学，メリーランド大学，ハーバード大学，MIT等を挙げれる。
上に報告したシステムの研究開発状況を大別すれば次の二つに分けられよう。

- 汎用分散型データベース管理システムの開発：CCA社のSDD1 → Multibase → DDMの流れ，IBMのSystem R → System R*の流れ，UC Berkeleyの INGRES → 分散型INGRESの流れ，等に代表される。
- OA（Office Automation）システム組込型分散型データベース管理システムの開発：この タイプの管理システムのアーキテクチャが十分研究されているとは思えないが，タンデム社のENCOMPASS上のOAのアプリケーションを主たる目的の一つにしている。シリコンバレーの小規模コンピュータ会社はこのあたりの市場を十分意識しているだろう。

分散データベースシステムが非分散のデータベースシステムと異なった取扱いをせねばならぬ技術的問題点は少なくとも次の二点を含むだろう。

- 分散型質問処理法
- 分散型のカタログ（DD/D）管理法，同時制御法，障害時回復法

現在迄のシステム開発を通して，決定的な解答が出ている訳ではない。典型的なシステムアーキテクチャを確立するために，単に実験室レベルでのパフォーマンス向上のみならず，そのプロトタイプを実際企業業務に適用して，システムアーキテクチャの改善と評価を行なってゆく必要があろう。

システム名＼区分	年代[*1]	マシン	通信網	インプリメンテーション言語	ユーザ言語	状況[*2]	タイプ	インプリメンター
SDD-1	1976~78(デ) 1979(終)	DEC-10とDEC-20で合計4台	Arpanet	BCPL 25,000行	Datalanguage(関係言語)	imp/inact	institutional	CCA マサチューセッツ州
分散型INGRES	1976~79(デ) 1979~(イ)	DEC VAX 11/780，DEC VAX 11/750各々5台	ETHERNET(3Mbit)	C 非分散INGRESの1.7倍	QUEL EQUEL	imp/act	↑	カリフォルニア大学 バークレー分校
System R*	1979~(イ) 1984(ヨ)	IBM 4341 Mod2 ×3台	SNA virtual circuit	PL/I	SQL PLI/SQL	↑	↑	IBMサンホセ研究 カリフォルニア州
ENCOMPASS	1981 商品発表	Tandem NonStop コンピュータ	long-haul net(56KB)	—	ENFORM	↑	commercial	タンデムコンピュータ社 カリフォルニア州
DDTS	1979~(イ)	Honeywell level 6 minicomp ×3台	Bus結合(coax cable)	C 20,000行	GORDAS(on ECR)	↑	institutional	ハネウエル/CCSC ミネソタ州
特別名なし	~1982~(イ)	プライムコンピュータなうなんでも	Primenet	—	—	↑	↑	プライムコンピュータ社 マサチューセッツ州
Multibase	1980~84(ヨ)	—	—	Ada	DAPLEX(on FDM)	↑	↑	CCA マサチューセッツ州
DDM	1982~84(ヨ)	DEC VAXマシン(on VMS)	—	Ada	DAPLEX ADAPLEX	des/act	↑	↑

表-1. 米国における分散データベースシステム概観

*-1. (デ)はデザイン，(終)はプロジェクト終了，(イ)はインプリメント，(ヨ)はプロジェクト終了予定。
*-2 impはインプリ完了かインプリ中，desはデザイン中，inactはプロジェクト終了 actは活動中。

3. システムR*(7~17)

3.1 システムR*の設計指針

システムR*が研究開発の技術的ゴールとしたのは次の三点である:(11,15)

- サイトの自治権の確立
- 便い易さ
- 高いパフォーマンス

サイトの自治権の確立はシステムのモジュラリティを上げ，システムの増殖をたやすくし，誤りに対する回復力を強める。しかし，このためには一切の集中制御の概念を捨て去った，完全に分散型のアーキテクチアを開発しなければならない。便い易さは新らしくR*に加入したユーザに，加入のための特別なロード(例えばカタログの書換，アプリケーションプログラムの変更)を一切かけないとするもので，他にシングルサイトイメージの保持，高レベル非手続的言語SQLのサポートを意味する。高いパフォーマンスは勿論サイトの自治権の確立や便い易さといった課題とは相矛盾する設計ゴールであるが，種々の最適化技法をとり入れて，それを達成しようとするものである。

3.2 システムR*の基本アーキテクチア

システムR*は均質(homogeneous)な関係データベース管理システムのあつまりである。ここに均質とは各システムが(versionは異なっても)同じ外部インターフェイスを持ち，共通の通信プロトコルに従っていればよいことをいう。図-1にR*のシステムの基本構造を示す。R*は大別して，●データベースマネジャー，●トランザクションマネジャー，●通信マネジャーからなる。データベースマネジャーは更にRDS(Research Data System, 分散データベース質問マネジャー)とRSS(Research Storage System, ローカルデータベース・マネジャー)

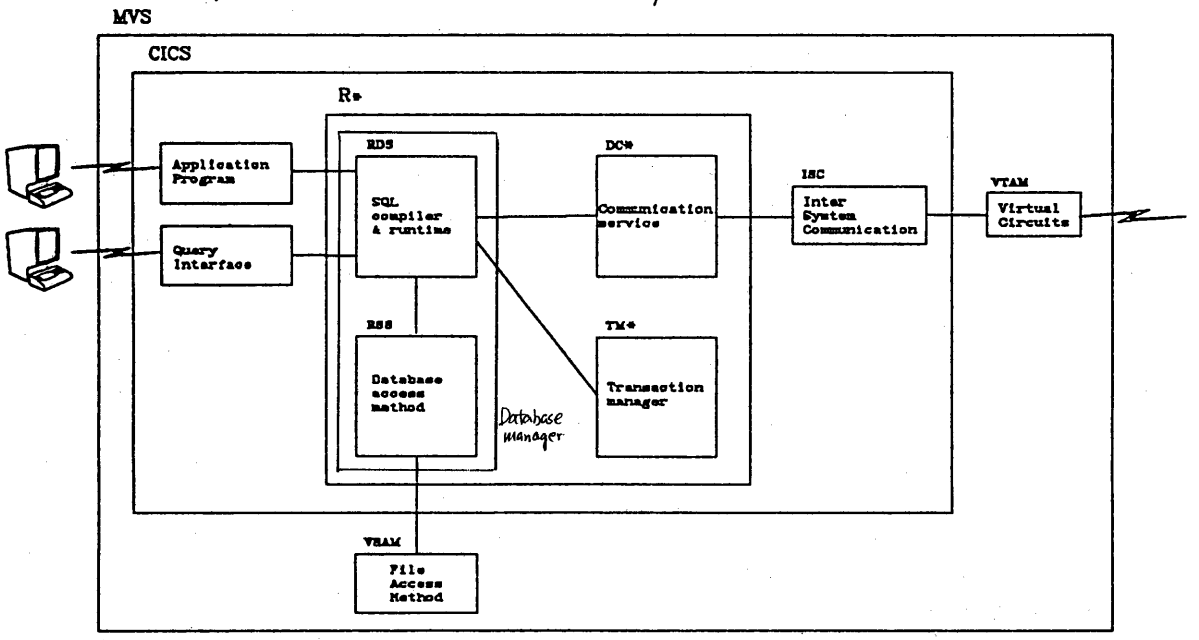

Figure 1: The R* System Structure (文献16より)

からなる。 RDSの仕事はネーム解読，権限チェック，アクセスパス選択，ビュー合成，コード生成である。 RSSはローカルトランザクション管理，デッドロック情報管理，同時制御，障害時回復，物理的ディスク管理の仕事をする。トランザクションマネジャーの仕事は分散された(多サイトの)トランザクションがそのトランザクションのプロセスが実行された全てのサイトでコミット(commit)されるか棄却(abort)されるかを保証することである。 2相コミットプロトコルの一変形である入れ子型のそれが使われている。 通信マネジャーはバーチャルサーキットを保持，入力要求をデータベース，トランザクション各マネジャーに適切にルーチングする。

3.3 システムR*のカタログ管理体系[9]

分散システムではデータ共有化実現のためのオブジェクト(関係テーブル，ビュー等)の命名法が必要である。 システムR*ではサイトの自治権確立が大きな技術的課題になっているので，次のような命名法をとった。 オブジェクトは

生成者名 @ 生成者サイト名 . オブジェクト名 @ オブジェクト誕生サイト名

の4項からなるSWN(System Wide Name)をもつ。 したがって，YOSHI@SJ.SAL@SJ と YOSHI@SJ.SAL@NY とはたとえ同じSALテーブルでも異なるオブジェクトとして認識される。 デフォールトは YOSHI がSJで logon し SAL という関係テーブルを作れば，自動的にYOSHI@SJ.SAL@SJというSWNを与える。 オブジェクトが格納されているサイトを格納サイトという。 格納サイトがSWNの構成要素に入っていないのは，元来オブジェクトは移動(migrate)するものであり，それをSWNに組入れるとオブジェクトのロケーショントランスパーレンシーが損なわれるからである。 オブジェクトの格納サイトはその生成時，SQL文の IN 句(例えば DEFINE TABLE SAL @NY IN SJ;)で指定できる。 オブジェクトの移動はMIGRATE文で指定できる。 シノニム(別名)はシステムRと同様定義できる。

分散データベースシステムのどのサイトでどういうオブジェクトが誕生，格納されているか，またどのようなアクセスパスを有するのか，等の情報を格納するため，システムR*は複数のその目的のための関係テーブルを用意しており，システムカタログと呼んでいる。 別の言葉ではDD/D(Database Dictionary/Directory)である。 システムR*のサイトの自治権確立というゴールから，システムカタログは完全に分散している。 つまりR*にはローカルカタログ，グローバルカタログといった区別は一切ない。 システムカタログのうちSYSCATALOGテーブルは，オブジェクトの生成，格納等の情報を保持している。 オブジェクトがあるサイトのSYSCATALOGテーブルにエントリーされるのは ●それがそのサイトで誕生したか，●そのサイトに格納されているか，あるいは●そのサイトでかって使われ，キャッシュ情報として保持される，かのいづれかのときである。 オブジェクトの誕生サイトのSYSCATALOGテーブルにはそのオブジェクトが消滅しない限り，それを記録する。 オブジェクトの統計量等はその格納サイトのSYSCATALOGテーブルに記入される。 他に，シノニム，ビュー，インデックス，権限等，等その目的のためにテーブルが定義されている。 このカタログ体系は次のことを実現できる。

- 新規に網に加入してもシステムカタログ，応用プログラムを変更しないでよい。
- オブジェクトの移動(migration)に応用プログラムは無関係である。
- 高々2回のリモートテーブル参照で，オブジェクトの格納サイトがわかる。(つまり，質問処理を受付けたサイトはまず自サイトのSYSCATALOGテーブルをみる。自サイトが誕生サイトならすぐに格納サイトが判る。もし格納サイトならそれもすぐ判る。もしキャッシュされていて，その情報がout-of-dateでなければ，直ちに格納サイトが判る。もしout-of-dateなら，指示されたサイトに要求してもそのオブジェクトは無いのだから，オブジェクトのSWNからその誕生サイトを識別し，誕生サイトに現在の格納サイトを問合せる。)
- キャッシュ情報を格納するためパフォーマンスが向上する。(キャッシュされたオブジェクトが他のトランザクションにより変更をうけた場合に，out-of-date化されるが，カタログにバージョン数を保持することにより，発見，訂正される。 ※3.4節参照。)

3.4 システムR*の分散型質問処理体系[8, 12, 13, 14]

PL1/SQLプログラムは一般にSQL文を含み，プレーコンパイル時にこのSQL文の表わす質問が分散コンパイルされる。システムR*では，そのプログラムが提示されたサイト(主サイトと呼ぶ)をコオーディネーター(coordinator)とし，そのプログラムの質問処理に関係するサイトを従サイトとし，分散型コンパイルを行なうという方式を採用している。この方式はシステムR*が設計目標にあげているサイトの自治权の確立にもあい，システムカタログの構造とも整合する。分散型コンパイルの結果，従サイトに低レベルのアクセスモジュールが格納される。これはランタイム時にアクセスインタプリータにより実行される。

主サイトでの質問コンパイレーションは，1. SQL文の構文解析とネーム解読，2. パーズ木を基にカタログ参照，オーソリゼーション検査，ビュー合成，3. グローバル(アクセス)プラニング(この結果，1.の時点でえられた従サイトへのSQL文と3.でえられるグローバルプランを従サイトに送る)，4. アクセスジェネレーションをして主サイトのサブセクションに格納，というステップをとる。従サイトでは送られてきたSQL文とグローバルプランからローカルアクセスプランを生成する。(そのSQL文に保護ビューの存在するときはグローバルプランも生成する。) 主サイトがグローバルプランを作成するので，それが使用したカタログデータがout-of-dateであれば，従サイトでそれらのバージョン数を検査して，主サイトに報告，主サイトはグローバルプラニングをやり直す。

グローバルプラニングは次のコスト関数Tcが最小となるように行なう。

$Tc = W_1 \times$ 送信されたメッセージ数 $+ W_2 \times$ 送信されたバイト数
$+ W_3 \times$ フェッチされたディスクページ数 $+ W_4 \times$ 要求されたデータ数

関係テーブルの(自然)結合をとる場合のコストを考えてみると，それは●結合順序，●結合法(システムR*では入れ子型かマージ型)，●各テーブルのアクセスパス，●結合場所(サイト)，●テーブルの送信法(全体か一部か)，等に依存する。例えば，二つのテーブルのマルチサイト結合を考えてみると，R*で

は次の場合を想定している。（入れ子型は正式には入れ子ループ型。）

- アウターテーブルをインナーサイトに送る。インナーサイトではアウターテーブルの全ての到着順に，入れ子型かマージ型で結合を開始する。
- インナーテーブルをアウターサイトに送る。アウターサイトではインナーテーブル全体を格納した後，マージ型結合を行なう。
- インナーテーブルをアウターサイトに送る。アウターサイトでは必要なインナーテーブル部分が到着次第，入れ子型かマージ型で結合を開始する。
- アウターテーブルもインナーテーブルも第三サイトに送りそこで結合する。
- 第三サイトにアウターテーブルを格納し，必要なインナーテーブル部分が第三サイトに到着次第，結合を行なう。

グローバルプランの生成はアクセスパス選択探索木を breadth-first に総あたりでサーチして決める。グローバルプランはミニプランの系列として表わされる。図-2はサイトCにあるテーブル1をサイトAに送り，それをアウターテーブルとして，入れ子型でテーブル3と結合し，次いでテーブル2がサイトBからサイトAに送られ，それが格納されたあと，テーブル1との結合結果とマージ法で結合するプランを表わしている。図中，ボックスが一つのミニプランに対応する。VT (Virtual table) はリモートサイトからテーブルをとりこむための placeholder である。グローバルプラン中，連続したミニプランの最大サブセットをサブセクションと呼び，VTは他のサイトのサブセクションへの call を表わしている。サブセクション全体がアクセスモジュールである。[13]

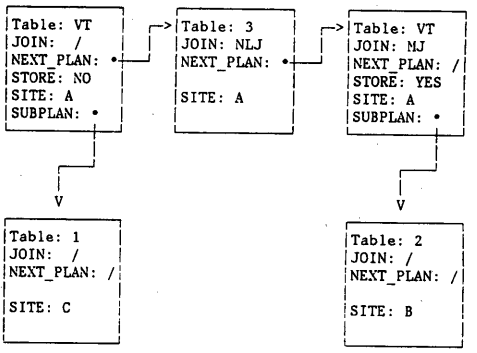

Figure 2. A Global Access Plan（文献13より）

システムR*のアクセスモジュールは●アクセスパスの欠落，●データオブジェクトの変化，●アクセス権の撤回，により無効化 (invalidation) される。依存性の損壊は一般にはアクセスモジュール全体に及ぶものではなく，その幾つかのサブセクションの範囲に収まることが多いことに注目する。無効化に自動的に対処するため，コンパイル時にアクセスモジュールの依存性がカタログ（たとえば，usage カタログ）に記録される。

アクセスモジュールを再有効化 (re-validate) するため，再コンパイルが行なわれる。システムR*ではセクション単位で再コンパイルをするグローバル再コンパイルと，サブセクション単位で再コンパイルをするローカル再コンパイルの二つの方式を用意している。入れ子型結合のインナーテーブルのインデックスが欠落していた時にはグローバル再コンパイルを，マージ型で同様の欠落があった時とアウターテーブルのインデックスが欠落していた時はローカル再コンパイルで良い。アクセス権の撤回による無効化はローカル再コンパイルで良い。ローカル再コンパイルで良いとしている時は，その結果得られたアクセスモジュールがほぼ最適であるからである。[12]

3.5 システムR*のトランザクション管理体系[7,11,16,17,18]

Ad-hocなSQLターミナルユーザではSQL文が，PL/SQLプログラムではBEGIN TRANSACTIONに始まりEND TRANSACTIONに終る一連のステートメントが一つのトランザクションをなすのはSystem Rと同じである。

システムR*のトランザクションが，システムRのトランザクションと異なる点はそれがマルチサイトトランザクションである点である。トランザクションはコミット(commit)されるかアボート(abort)される。マルチサイトトランザクションではそのトランザクションがデータベースアクセスと更新で関与した全てのサイトが一様に(uniform)コミットするかアボートしてほしい。そのためR*は入れ子型2-相コミットプロトコルを使用している。トランザクションのプロセス木を作る。根プロセスはそのトランザクションに対応し，根プロセスとオニレベルのプロセスはR*のバーチャルサーキットで結合している。以下のレベルも同様である。トランザクションのコミットかアボートかを決定するため，コオーディネータ(coordinator，つまりそのトランザクションのサイト)はオニレベルに"prepare to commit"要求を出す。(この結果サブセクションがアクセスモジュールに格納され実行される。)そして"vote yes"か"vote no"を返す。オミレベルがあれば，オニレベルから同様に入れ子になった型でオミレベルに"prepair to commit"を出す。これが"collecting"相(オー相)である。オー相でコオーディネータは投票の結果をみてコミットかアボートかの指令を出す。プロセス木のノード数をNとすると，メッセージの総数は4(N-1)となる。R*では読み出し専用の部分木に対してはオー相の会話を省略し，最適化を計っている。[16]

トランザクションリカバリにはデータグラムを使う。障害のおきたサイトとはバーチャルサーキットでの会話は行なえないので，それが出来上るまでデータグラムを使う。

トランザクションの同時制御はシステムRと同様，2-相ロック方式[18]を使っている。トランザクションにはR*全体で唯一識別可能なトランザクション番号(サイト名とローカル時刻の組)をつけている。この結果，同一サイトの異なるプロセスでも同一のトランザクション番号を持っていれば，データオブジェクトのロックを共用させてやることができる。

サイトの自治権確立というシステムR*のゴールに抵触しないデッドロックの検出と解消のアルゴリズムが開発されている。検出アルゴリズムでもトランザクション番号が必要である。デッドロックが起った場合，犠牲となるのはトランザクション番号が最も大きい(youngest)トランザクションである。このグローバルなデッドロックの検出と解消アルゴリズムは既にインプリメントされている。[17]

4. 結言

システムR*のプロジェクトは10人程のプロジェクトである。現在，プロジェクトリーダはBob Yost，メンバーはBruce Lindsay, C. Mohan, Guy Lohman, George Lapis, Paul Wilms, Laura Haas, Elisa Bertinoである。Pat Selingerは本年一月にプロジェクトをはなれた。本稿では，

ビューサポートについての記述を省略した。別の機会に譲る。テストプログラムもシステムR(及びSQL/DS)の遺産を手直して，膨大な数ある。システムR*がシステムRと同様，商品化されるかどうかの決定は未だなされていない。

【謝辞】著者はIBMサンホゼ研究所でシステムR*の一員として，その研究開発に携わる好機を与えて下さった日本IBM(株)に心から感謝の意を表する。著者はまた，気持良く客員研究員としての一年間を送らしていただいたR*プロジェクトのメンバーの皆様，そしてIBMサンホゼ研究所の皆様に心から感謝の意を表する。

【文献】

(1) J.B.Rothnie他, Introduction to a System for Distributed Databases (SDD-1), ACM TODS 5-1('80)
(2) M.Stonebraker他, Performance Analysis of Distributed Data Base Systems, IE³, Database Engineering 5-4('82)
(3) J.Nauman, ENCOMPASS: Evolution of a Distributed Database/Transaction System, ibid.
(4) S.k.Rahimi, A Structural View of Honeywell's Distributed Database Testbed System DDTS, ibid.
(5) D.DuBourdieu, Survey of Current Research at Prime Computer Inc. in Distributed Database Management Systems, ibid.
(6) A.Chan, Distributed Database Management Research at Computer Corporation of America, ibid.
(7) B.Lindsay他, Notes on Distributed Databases, IBM Res. Rep. RJ 2571 ('79)
(8) P.G.Selinger他, Access Path Selection in Distributed Database Management Systems, IBM Res. Rep. RJ 2883 ('80)
(9) B.Lindsay, Object Naming and Catalog Management for a Distributed Database Manager, IBM Res. Rep. RJ 2914 ('80)
(10) B.Lindsay他, Site Autonomy Issues in R*: A Distributed Database Management System, IBM Res. Rep. RJ 2927 ('80)
(11) R.Williams他, R*: An Overview of the Architecture, IBM Res. Rep. RJ 3325 ('81)
(12) P.Ng, Distributed Compilation and Recompilation of Database Queries, IBM Res. Rep. RJ 3375 ('82)
(13) D.Daniels, Query Compilation in a Distributed Database System, IBM Res. Rep. RJ 3423 ('82)
(14) D.Daniels他, An Introduction to Distributed Query Compilation in R*, IBM Res. Rep RJ 3497 ('82)
(15) L.Haas他, R*: A Research Project on Distributed Relational DBMS, IE³, Database Engineering 5-4 ('82)
(16) B.Lindsay他, Computation and Communication in R*: A Distributed Database Manager, IBM Res. Rep. RJ 3740 ('83)
(17) R.Obermark, Distributed Deadlock Detection Algorithm, ACM TODS 7-2 ('82)
(18) P.A.Bernstein他, Concurrency Control in Distributed Database Systems, Computing Surveys, 13-2 ('81)