Vol. 28 No. 4　　情　報　処　理　　Apr. 1987

解　説

4.3 分散型データベースシステム†

増　永　良　文††

1. 序　論

分散型データベース管理システム（分散型 DBMS と略記）とは地理的あるいは論理的（たとえば同一の OS 上で異なったユーザジョブとして稼動している場合）に分散したデータベース管理システム（DBMS）を計算機ネットワークを用いて結合し，データベースのユーザにはその分散を意識させることなく（分散の透明性を達成しているという），全体があたかも一つのデータベースであるかのように見え，アクセスが可能である DBMS をいう．図-1 にその概念を示す．分散型 DBMS を構成している個々の DBMS をメンバ DBMS ということにする．

さて，分散型 DBMS は 1970 年代中ごろより研究開発が始まり，現在まで世界で二十数例の開発報告がなされている[Masu 86]．その時期に分散型 DBMS の研究開発が始まった理由としては次をあげることができる[Masu 84]．

（1） リレーショナル DBMS がシングルサイトで実動しだした．

（2） 計算機ネットワーク（たとえば ARPANET や ETHERNET）技術が十分固まった．

（3） 分散処理システムへの理論的，実践的期待が高まった（たとえば高可用性の実現）．

（4） ユーザからの分散型 DBMS 構築要求が高まった．

具体的には 1976 年に米国 CCA 社で開発された SDD-1[RBF. 80] をさきがけとして，分散型 INGRES [StNe 77], System R*[WDH. 82], SIRIUS-DELTA[FeSt82], VDN[Muns 79], DDM と MULTIBASE[ChRi 82] や日本の JDDBS[Taki 83], DEIMS-3[MoYS 85], RDB/DV[TAY. 85], 分散型 FREND[HiKH 85] などさまざまである．

分散型 DBMS の構築にあたっては，その要求の背景や構築環境を把握しておくことは是非必要である．

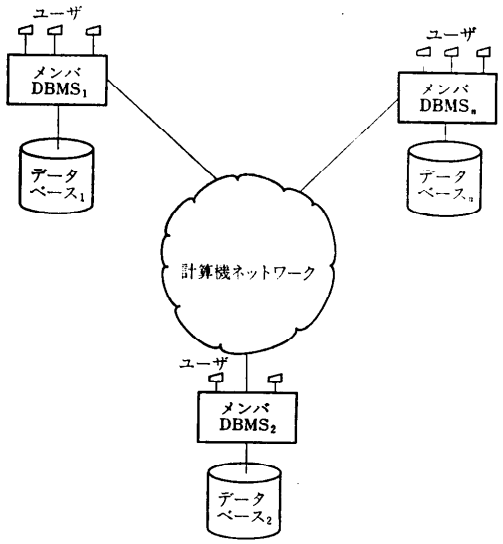


図-1 分散型データベース管理システム

少なくとも次の項目をあげることができよう．

（a） 分散型 DBMS 構築アプローチはトップダウン的なのかボトムアップ的なのか．すなわち分散した既存の DBMS を統合して一つの分散型 DBMS を作り上げてゆく（ボトムアップ）のかそうでないのか．

（b） 同種（homogeneous）分散型 DBMS を構築するのか異種（heterogeneous）システムを構築しようとしているのか．分散型 DBMS のメンバ DBMS がすべて同じとき同種システムという．そうでないとき異種である．異種性には弱い異種性と強い異種性があろう（第 3 章参照）．

（c） WAN（Wide Area Notwork）を計算機ネットワークとして使用するのか，LAN（Local Area Network）を使うのか．WAN では数十 Kbit/秒の伝送速度の点間（point-to-point）通信で LAN ではメガビット/秒の速度と放送機能が使える．ほかに衛星通信が使えよう．この場合通信遅れが大きくなる．

（d） 分散型 DBMS の構築目的はなんなのか．汎用システムの開発か，応用依存性のあるシステムか

† Distributed Database Management Systems by Yoshifumi MASUNAGA (University of Library and Information Science).

†† 図書館情報大学

(たとえばビジネス応用を指向した ENCOMPASS [Naum 82]), あるいはワークステーション環境整備のためか.

本稿の目的は分散型 DBMS の構築技術を解説することにある. 以下三章に分けて, 汎用分散型 DBMS の構築技術 (第 2 章), 異種分散型 DBMS の構築技術 (第 3 章), 分散型ワークステーション DBMS の構築技術 (第 4 章) を解説する. 第 5 章は結論である.

2. 汎用分散型 DBMS の構築技術

2.1 アーキテクチャ概観

ビジネス応用を中心としたさまざまな応用分野への適用を目的とした分散型 DBMS を汎用ということにする. 汎用分散型 DBMS はシングルサイト用に開発された汎用 DBMS を設計変更して, 分散版にした場合が多い. したがってシステムは同種構成となる. たとえば System R* や分散型 INGRES, あるいは RDB/DV などはその典型例である. 特徴としてはリレーショナルデータモデルに基づいたシステムである点を指摘できる. それはほかのデータモデル, たとえば CODASYL のネットワークモデルやハイアラキカルモデルに比べて, リレーショナルモデルは(i)データ (リレーション) の分散が容易に行える, (ii)通信に強い (すなわちリレーションがデータベース演算とデータ伝送の単位), という特長を備えているからである [CoDa 74].

分散, 非分散を問わず, DBMS を構築するには次の三つの技術体系を明らかにせねばならない.

(1) メタデータ管理体系
(2) 質問処理体系
(3) トランザクション管理体系

以下本章では, まず (2.2~2.4 節) 汎用分散型 DBMS として完成度の高い System R* を例として, 上記三つの体系がどのようであるかを概観する. (System R* のアーキテクチャの詳細は文献 [Masu 84] を参照.) 次いで (2.5 節と 2.6 節) SDD-1 で実装され, その後の分散型 DBMS 構築に強いインパクトを与えた二つの技術: (a)準結合 (semi-join) を用いた分散型質問処理体系, (b)時刻印 (time stamp) を用いた分散型トランザクション管理体系 (とくに同時実行制御) の概略を示す.

2.2 分散型メタデータ管理体系

分散型 DBMS 全体にどのようなデータが格納されているのか, だれがアクセスできるのか, どのようなアクセスパスが設けられているのか, 等々のデータをメタデータと呼ぶ. システムを実動させるためには必要にして十分なメタデータを組織化・管理・運用しなければいけない. DD/D (データ辞書) とも呼ばれる概念であり, 通常リレーショナル DBMS ではシステムカタログと呼ばれている. システムカタログは複数 (System R* では 20 数個) のカタログテーブル (カタログリレーション) からなっている. System R* の各メンバ DBMS はたとえば SYSCATALOG というカタログテーブルを有している. これはそのサイトで誕生したリレーション, 格納されているリレーション, あるいはキャッシュされたリレーションのデータ (リレーション識別番号, リレーション名, 生成者名, 生成者サイト名, 格納サイト名など) を格納するためのものである.

なお, リレーションやビューなどの R* のオブジェクトがネットワーク全体で一意に識別可能であることがメタデータ管理・質問処理・トランザクション処理のため必要である. そのため, オブジェクトの命名法がいる. System R* では次の 4 項系列法が採用されている.

生成者名@生成者サイト名. リレーション名@リレーション誕生サイト名

たとえば TOKYO サイトでログオンした YOSHI 氏が KYOTO サイトに EMPLOYEE リレーションを生成すれば, 次のシステムワイド名が与えられる. YOSHI @ TOKYO. EMPLOYEE @ KYOTO. なお生成したリレーションをどのサイトに格納するかは IN 句で別に定める.

2.3 分散型質問処理体系

2.3.1 質問処理の最適化

リレーショナル DBMS ではデータベースに対する質問 (query) は非手続的 (non-procedural) に表現される. これはリレーショナルデータモデルがネットワークなどのほかのモデルと大いに異なる点で, 結果として求めたいデータ (タップル) の集合をデータベースから切り出すために, なに (what) を求めたいのかを書けばよいのであって, how, すなわちそれをどのように求めるのかの手続きを書く必要はないということである. 一方, ANSI/X3/SPARC [TsKI 78] のスキーマ表現に従うと, 概念スキーマとして組織化されたデータベースは, 内部スキーマに変換され二次記憶装置に格納される. しかし内部スキーマは一般に B-

tree ファイルであったり，ISAM ファイルであって手続き的にデータアクセスをしなければならない．そのため，リレーショナル DBMS ではユーザの発した非手続き的に表現された質問を手続き的な質問に変換する必要がある．しかし，what を how に変換する仕方は一般には複数あることが想定され，それら代替案は実行に必要なコストで評価される．つまりコスト最小の代替案を見つけることが必要で，これを質問処理の最適化 (query optimization) と呼んでいる．本節では System R* の最適化技法を解説する．なお，リレーショナルデータ言語 SQL[IBM 82]や，SQL を PL/I に埋め込んだ親言語方式の質問言語 PLI/SQL で書かれた質問はインタ・プリタやコンパイラを使って処理されようが，いずれの場合も質問処理の最適化は行わないといけない．なお，質問処理の最適化をより大局的に捉えると，平均的質問処理コストを最小にする意味でのリレーションの最適分散の問題がある．またこのような環境をより積極的に支援するための，リレーションの重複生成，水平・垂直分割の研究も行われている[Masu 85]．

2.3.2 分散型コンパイレーション

分散型 DBMS で，コンパイレーション方式で，質問を処理しようとする場合，さまざまな方式を考えることができる．簡単に述べれば(i)あるサイトで集中コンパイルする，(ii)質問を受け付けたサイトでコンパイルする，(iii)質問を受け付けたサイトの DBMS が主となり，質問に関係するリレーションが格納されているサイトが従サイトとなり主サイトと協調して，コンパイレーションを行ってゆく，あるいは(iv)完全分散でコンパイルする等々である[Dani 82]．

System R* のようにサイトの自治権を尊重する場合には代替案(iii)が適当である．図-2 にその分散型コンパイレーションの概念図を示す．主サイトで SQL 文をどう処理するかのプラン（グローバルアクセスプラン）をたてる．主サイトで行うべきデータベースアクセスモジュールが生成・格納される．一方，関係する従サイトでも主サイトから送られてきたデータにより自サイトのアクセスモジュールを生成する．なお，コンパイレーション方式では，データベースアクセスモジュールが生成された時点と，それが実行される時点が一般には異なるので，コンパイルした時点でのさまざまな状況に変化が生じ，モジュールが実行不可能になったり，最適な実行が保証されなくなる恐れがある．(たとえばアクセス権が撤回されていたり，当初

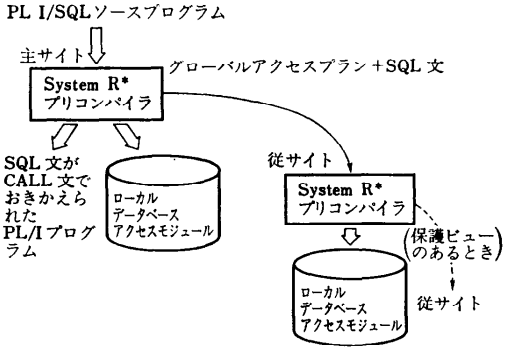


図-2 分散型コンパイレーション (System R*)

存在したインデックスが欠落していたりというようなこと.) したがって，アクセスモジュールの実行時には有効性を検査することが必要で，そのような状況が検出されれば SQL 文の再コンパイレーションを一般に行う必要がある．ただ，SQL 文をはじめからコンパイルし直すとコストがかさむので，質問処理の最適化の観点からは，なんとかアクセスモジュールの一部を再コンパイルするだけで準最適が保証されるようにならないものかという考え方が生まれる．System R* では計算機シミュレーションの結果，SQL 文全体を再コンパイレーションしないといけない場合，一部を再コンパイレーションすればよい場合というのを切り出している[Ng 82]．

2.3.3 ジョイン質問処理の最適化

SQL の質問文を大別すると次の三種になる．(i) 単一リレーション質問，(ii)ジョイン質問，(iii)入れ子型質問 [あるいは部分質問 (subquery)]．分散，非分散を問わず，質問処理の最適化で大事なことは，ジョイン質問をどう処理するかである．ここにジョイン質問とは二つ以上のリレーションの（自然）結合を必要とする質問である．単一リレーション質問処理の最適化は困難ではなく，また入れ子型質問を最適化するには，ジョイン質問の最適化ができていないといけない．

さて，ジョイン質問の最適化を考えるために，それを処理するに必要なコストを計算する．従来の非分散の環境でのコスト算出式が基礎となり，System R では次のようである[ABC. 76]．

$$C = N_1 + W \times N_2$$

ここに N_1 は与えられたジョイン質問を処理するために二次記憶装置からフェッチするデータとインデック

スの入っているページ数，N_2 はその質問を処理するために使用する CPU 時間（これは質問の結果，なん本のタップルが生まれるかの数で当てる），W は N_2 を1ページあたりのフェッチコストに変換するための重み係数である．W は DBMS を搭載している計算機が，CPU バウンドなのか I/O バウンドなのかにより，大きく異なってくる[SAC.79]．もちろん，ジョインコストはリレーションの結合順序，結合法，各リレーションのアクセスパスにより，異なってくる．

分散環境では次の二点で，ジョイン質問の最適化の考え方が，非分散の場合に比べて，大きく変わってくる．

（1） 分散したサイト間でのメッセージとデータの転送に新たにコストがかかる．

（2） 二つのリレーションをジョインする場合，一般に三つのサイトが関係してくる．

とくに(2)項は，サイトAにリレーションR，サイトBにリレーション S があり，自然結合 R*S をとるには，大別して(i)サイト A に S を送って R * S を行うか，(ii)サイト B に R を送って R*S を行うか，(iii)新たに第三のサイト C が関与し，C に R と S を送って R*S を行うか，という代替案が出てくることを意味している．

なお第一項の通信コストに関しては，いかなる計算機ネットワークを使おうとしているのかで，コストは大きく異なる．WAN では伝送速度が遅いので，通信コストが全コストに占める割合は大きくなる．一方，LAN では小さい．速度の面からみた両者の差異で，採用する質問処理最適化の技法への考え方が変わってくる．（準結合の提案はその一例である．**2.5** 節参照．）

System R* ではリレーションの自然結合は次のいずれかの方法で行う．それらは(i)入れ子型ループ法と(ii)マージ・ジョイン法である．二つのリレーション R と S を自然結合しようとするとき，ごく普通に，まず R のタップル一本を固定し，それと S タップルすべてとのジョインの可能性をためし，このプロセスを繰り返すだろう．そのとき R をアウタリレーション，S をインナリレーションと言うことにする．入れ子型ループ法では，アウタリレーションのタップル一本ごとにインナリレーションのタップルを総あたりでジョインする無駄を避けるため，インナリレーションのジョインカラム上にインデックスがついていることを前提としている．リレーショナル DBMS ではインデックスもまたリレーションである．System R* の分散環境ではリレーション S を他サイトに送る場合，S 上に定義されたインデックスは送られない．したがってサイトAにRがありサイトBにSがあり，SをサイトAに送り入れ子型ループ法で自然結合はできないという制限が付けられている．**表-1** に System R* で可能な5つの自然結合代替案を示す．

2.4 分散型トランザクション管理

2.4.1 トランザクション

トランザクション (transaction) とはデータベースの問合せと更新作業（リレーションへのタップルの挿

表-1 System R* における二リレーションの三サイトジョイン（5つの代替案）

方式＼項目	リレーションの送信先	リレーションの送信法	ジョインサイト	ジョイン法
方式 1	アウタリレーションをインナリレーション格納サイトへ送る	インナリレーション格納サイトの要求に基づきアウタリレーションのタップルを一本ずつ送信する	インナリレーション格納サイト	入れ子型ループ法とマージジョイン法が可能
方式 2	インナリレーションをアウタリレーション格納サイトへ送る	インナリレーションを一括してアウタリレーション格納サイトへ送信する	アウタリレーション格納サイト	マージジョイン法
方式 3	同　上	アウタリレーション格納サイトの要求に基づきインナリレーションの一部を送信する	同　上	入れ子型ループ法とマージジョイン法が可能
方式 4	インナリレーションもアウタリレーションも第三サイトへ送る	インナリレーションもアウタリレーションも一括して第三サイトへ送信する	第三サイト	マージジョイン法
方式 5	同　上	アウタリレーションは一括，インナリレーションは第三サイトの要求に基づきその一部を送信する	同　上	入れ子型ループ法とマージジョイン法が可能

入，削除，書換え）の単位である．通常一SQL 文はトランザクションであり，PLI/SQL プログラムにおいてはトランザクション開始文で始まり，一連の更新作業の終了後，データベースの更新をトランザクションマネジャにコミット (commit) するか，あるいはそれらを棄却 (abort) して終了するのがトランザクションである．ここにコミットとは具体的にはデータベース更新の論理的ログ (log, 記録) を物理的ログに書き換えることをいう．分散環境では，一般にあるサイトで受け付けたトランザクション（グローバルトランザクションと呼ぼう）を実行するにはほかのサイトのデータベースのトランザクションを生む．これをサブトランザクションと呼ぶ．たとえばサイトAからサイトBに送金するトランザクションをサイトAが受け付けた場合，サイトBに口座の預金額を増やすためのサブトランザクションが定義される．この意味で分散環境ではトランザクションはマルチサイトトランザクションである．

トランザクションの概念を持ち込むことにより，データベースの障害時回復や，分散型 DBMS では必ず問題になる多数のユーザが同時にデータベースの同じ部分を読み書きしようとするときに発生する衝突をどう制御するかという同時実行制御 (concurrency control) 問題解決の理論的基礎を与える．

大別して，次の方式が提案されたり実用化されている．

- 障害時回復：二相コミットプロトコル
- 同時実行制御：二相ロック法，時刻印法，楽観的制御法[KuRo 81]

2.4.2 二相コミットプロトコル

二相コミットプロトコルは分散環境においては System R* を始めとして最も広く使われている障害時回復法である．二相コミットプロトコルに従うとグローバルトランザクションがコミットするためにすべてのサブトランザクションにコミットできるか否かをコミット準備命令を送信して問う（第一相）．その結果全員がそうであれば，コミットをさせ，グローバルトランザクションがコミットしたことになる．サブトランザクションが一つでもコミットできなければ，グローバルトランザクションは棄却される．System R* では，グローバルトランザクションからサブトランザクションが生まれてゆく過程を木 (tree) 構造として捉えており[LHM. 83]，コミット準備命令とそれに対する返答はこの木構造に従い階層的に伝播されるよう管理されている．

2.4.3 分散型二相ロック法

分散型，非分散型を問わず，多数のトランザクションをできるだけたくさん同時に実行できるよう制御することはたいへん重要な問題である．そのための十分条件がトランザクションの直列化可能性 (serializability) という概念である[EGLT 76]．つまり，トランザクションの集合が直列化可能とは，同時に実行されたトランザクションのおのおのの結果が，かりにそれらトランザクションをある順番で順に実行させたとした結果にちょうど等しくなるようなときをいう．

二相ロック法では，トランザクションが実行可能となるのは，それが必要とする資源（たとえばリレーション）がすべて施錠（ロック）されたときである．この方法は明らかに直列化可能性を保証する．またインプリメンテーションが容易なため広く使われている．ただ，デッドロック（いくつかのトランザクションが資源をとり合い，いずれもが必要とするすべての資源を施錠できず，お互いが他の実行終了を待つという永久待ちの状態に入ってしまう状況）が起こり，その検出および解消法を必要とする．

分散型二相ロック法は資源の施錠をサイト間にまたがって行わせるよう拡張された方法である．System R* はこの方法を採用している．手続き最小でデッドロックが検出できる分散型デッドロック検出法が開発され実装されている．デッドロックが検出された場合，システムワイドに唯一に割り振られたトランザクション番号（ローカルタイムとサイト識別番号の対）を参照し，一番若いトランザクションを犠牲にする．なお，分散型二相ロック法ではデッドロックを検出・解消するためにサイト間での通信が必要で，その分コストがかかる．

2.5 準結合法によるジョイン質問の処理

ジョイン質問の処理法として準結合 (semi-join) という手法が SDD-1 で提案・実装された[BGW. 81]．たとえばサイトAにあるリレーション R(X, Y) と，サイトBにあるリレーション S (Y, Z) の自然結合をとろうとする場合，RかSを他サイトに送らないといけないが，準結合方式ではそうするのではなく，R〔Y〕(RのY上の射影演算の結果) をまずサイトBに送りSと自然結合させ，その結果として生ずるSの部分リレーションをサイトAに送らせ，Rと結合させようとするのが準結合の基本的アイディアである．Sをそのままサイトに送付しRと自然結合させるコストより，こ

の方が安いのではないかということである．これは通信コストがボトルネックとなるような場合に有効となる．準結合の考え方を巡回質問のクラスに適用可能とするための一般化が行われている[KaYY 82]．

2.6 時刻印法による同時実行制御法

時刻印に基づいた同時実行制御法の基本的動作は次のとおりである[CePe 84]．まず，データ項目 x（たとえばリレーション）を読み出したり，書き込んだりする操作にはそれらを要求したトランザクションの生成時に付与されたと同じシステムワイドで唯一の時刻印を付与する．一方データ項目には読み出された最近の時刻 RT(x) と書き込まれた最近の時刻 WT(x) が記録されている．さて，

（1） データ項目 x に時刻印 TS の読み込み操作が発せられたとする．もし TS<WT(x) なら，その読み出し操作を行わせず，かつそのトランザクションに最新の時刻印を与え再スタートさせる．TS≧WT(x) なら読み出しを行わせ RT(x) を RT(x) と TS のうち大きな値で置き変える．

（2） データ項目 x に時刻印 TS の書き込み操作が発せられたとする．もし TS<RT(x) か TS<WT(x) なら，その書き込み操作を拒否，そのトランザクション番号に最新の時刻印を付与し，再スタートさせる．それ以外なら書き込み操作を行わせ，WT(x) を TS で置き換える．

明らかに，この方法により，トランザクションの直列化が保証される．二相ロック法と異なり，あるトランザクションがほかのトランザクションの終了を待つということはなく（待つ代わりに再スタートする），デッドロックがない．したがって，分散型 DBMS においてはデッドロック検出のための通信のオーバヘッドがない．しかし，常に古いトランザクションが犠牲となるので，大きなトランザクションはなかなか終了できないという危険性がある．

SDD-1 では上記の基本アルゴリズムの変形が実装された[BeSR 80]．しかしそのアルゴリズムではデッドロックが生じる可能性があることが指摘されている[Mc-L 81]．

3. 異種分散型 DBMS の構築技術

3.1 異種分散型 DBMS のアーキテクチャ

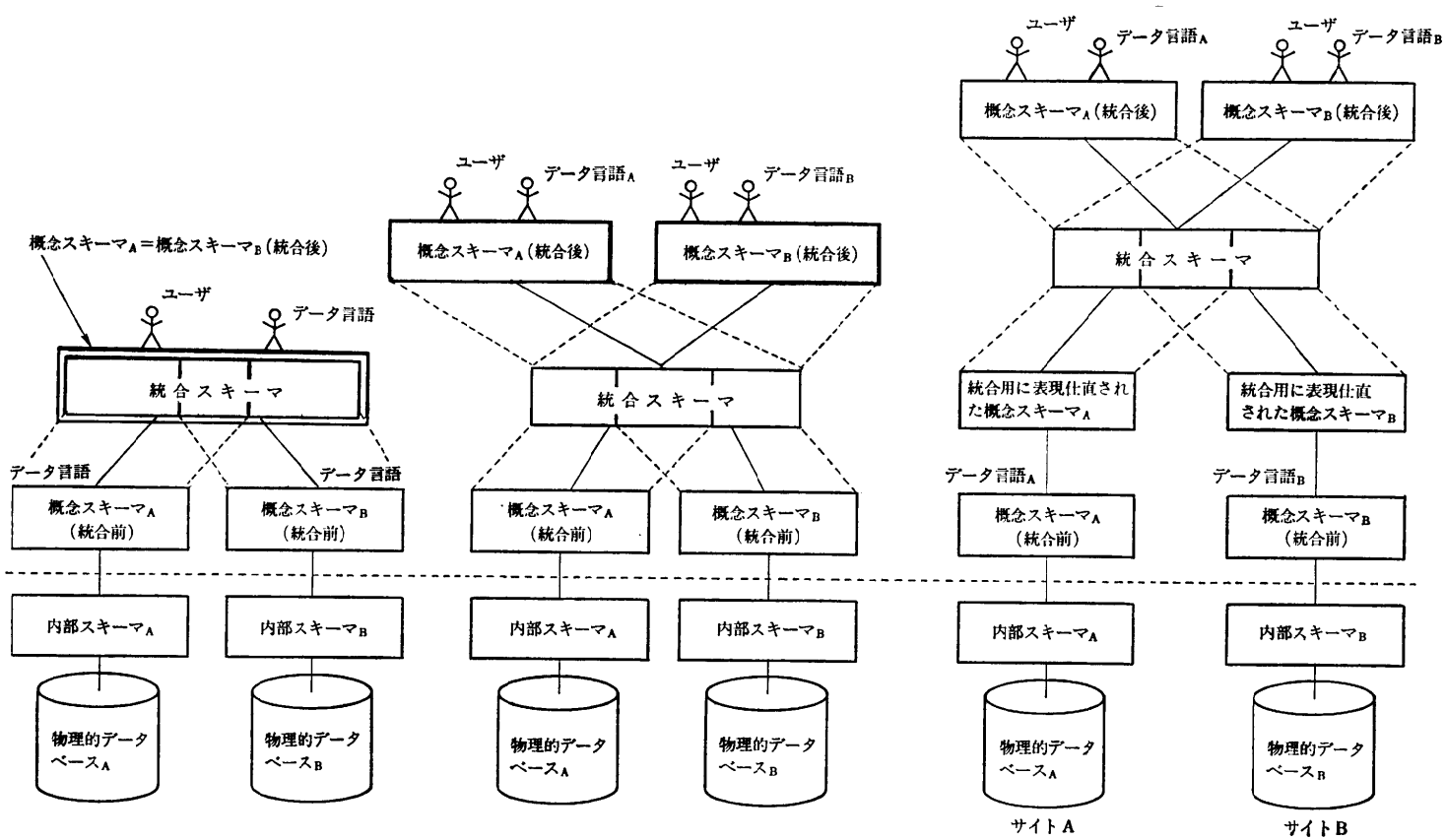
マルチベンダの DBMS を統合するには DBMS の異種性を克服しなければならない．異種性には強，弱の区別のあることはすでに指摘した．弱い異種性では分散型 DBMS を構成するメンバ DBMS のデータモデルは同じ（たとえばリレーショナル）であるが，インプリメンテーションが異なり，したがってデータ言語（データの定義，質問，更新のための言語）が異なる．一方強い異種性では，メンバ DBMS のデータモデルが一般に異なり，システムのアーキテクチャも異なってくる．**図-3** に同種，異種分散型 DBMS アーキテクチャの概念を示す．強い異種性の場合（図-3 (c)）に言及する．たとえばサイトAのメンバ DBMS はリレーショナル，サイトBのそれはネットワークモデルとする．各サイトの概念スキーマは統合のため共通の統合データモデルを使って変換，表現し直される．次にそれらが統合され，統合スキーマが構築される．統合スキーマは再び各サイトのデータモデルに変換され，拡張された概念スキーマがサイトAではリレーショナルモデルで，サイトBではネットワークモデルで提示される．なお，ユーザがその上にユーザの視野（view）を定義するのは自由である．

従来開発された異種分散型 DBMS も基本的にこのアーキテクチャを採っている．統合用の共通データモデルとしては，リレーショナルモデルを採る場合が多い[FeSt 82], [Taki 83]．ほかに関数データモデル[ChRi 82]や実体-関連モデル[RaSL 82]を採用している例もある．問題点はデータモデルにまたがったデータベース変換がデータやその意味を保存して行えるかどうかという点である．（たとえば CODASYL データベースでは意味のあるレコードの順番をリレーショナルにとり込めるのか．）理論的限界がある．

DBMS が異なれば使用する同時実行制御法が異なることが当然予想される．二相ロック法と時刻印法を統合する問題が議論されている[Kako 85]．

3.2 RDA

ISO で RDA (Remote Database Access Service and Protocol) が検討されている[ISO 86]．これは DBMS を開放型 (open-ended) にして，DBMS 同士を相互に接続・運用可能とするためのプロトコルである．同じく ISO の7層の通信プロトコルの最上位，応用層上に実装される．DBMS 同士が通信する際使用するデータモデルはリレーショナルモデルで，データ言語としては SQL が予定されている．つまり分散した DBMS に SQL インタフェースを実装して，データベースの相互アクセスを可能としようとするものである．データベースの統合という概念は入っていないが，異種分散型 DBMS の一種と見なすことができる．



(a) 同種分散型 DBMS　　(b) 弱い異種分散型 DBMS　　(c) 強い異種分散型 DBMS

図-3 同種と異種分散型 DBMS アーキテクチャ

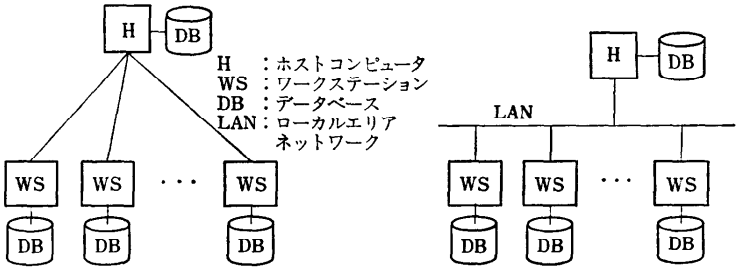
4. 分散型ワークステーション DBMS の構築技術

一機関一台の計算機の時代から一人一台のワークステーションの時代に入ってきている．ワークステーションでは従来のさまざまなデータ処理機能や日本語処理機能に加えて，データベース機能を充実させておくことが大事である．問題点は二つあり，一つはワークステーション上に機能的にみてどのような DBMS を搭載するのが良いかということと，もう一つはどのような分散型 DBMS 環境を実現すべきかということである．前者については，ワークステーション DBMS に求められているのはより高度なソフトウェアや設計作業の支援環境の実現であり，現状の UNIX 上の DBMS 機能ではいずれも不十分である．いつでも必要な機能を追加できるという意味での DBMS の拡張可能性 (extensibility) がワークステーション DBMS では必要となる．STARBURST[SCP. 86]はこの点で興味深い．

さて，分散型ワークステーション DBMS の構成はワークステーションとほかの計算機システムとの間に分担する作業の性格の違いから，一般的にネットワークが階層的に構成されないといけないという点で，従来開発されてきた分散型 DBMS とは異なろう．

図-4 に LAN を使用した階層的ネットワーク構成と，二点間接続に基づく階層的ネットワーク構成による分散型ワークステーション DBMS の概念を示す．

いずれの場合も議論しないといけない大問題はデータベースのダウンロード (down load) とアップロード (up load) の問題である．つまり，ワークステーション環境での最も典型的なデータベースアクセスの形態は，ホストデータベースからその一部（と必要とあらばそれの上のアクセスパス)をダウンロード(コピーを自分のデータベースに格納して使用に供すること)し，一連の作業の終了後，アップロード（ホストデータベースに返す）するというものである．この状況は従来の分散型データベース環境では発生しなかった現象で，したがってワークステーションとホスト DBMS の役割分担から始まって，DBMS 機能（メタデータ管理，質問処理，トランザクション管理）全般にわたりアーキテクチャを見直す必要がある．最近，図-4 (a)のネットワーク構成に従った一例として ADMS ±[RoKa 86]というシステムの設計が報告されている．



(1) 集中型を拡張したアーキテクチャ　(2) LAN を使用した分散型アーキテクチャ

図-4 分散型ワークステーション DBMS

5. 結　論

分散型 DBMS の構築技術を解説した．まず，汎用分散型 DBMS の構築技術を System R* と SDD-1 を事例として捉え，分散型メタデータ管理，質問処理，およびトランザクション管理の体系を解説した．続いて，異種分散型 DBMS のアーキテクチャの概略を解説した．最後に，これからますます重要となってくると思われる分散型ワークステーション DBMS の構築技術について基礎的解説をした．

計算機技術，通信技術の進歩は著しく，今後ますます分散処理の要求は高まり，またデータベースはあらゆる情報活動を支える源泉としての意義を増し，分散型 DBMS の構築要求は，増大する一方であろう．

参考文献

[ABC. 76] Astrahan, M. M., Blasgen, M. W., Chamberlin, D. D., Gray, J. N., Griffiths, P. P., King, W. F., Lorie, R. A., McJones, P. R., Mehl, J. W., Putzou, G. R., Traiger, I. L., Wade, B. W. and Watson, V.: System R: Relational Approach to Database Management, ACM TODS, Vol. 1, No. 2, pp. 97-137 (1976).

[BeSR 80] Bernstein, P. A., Shipman, D. W. and Rothnie, Jr,. J. B.: Concurrency Control in a System for Distributed Databases (SDD-1), ACM TODS, Vol. 5, No. 1, pp. 18-51 (1980).

[BGW. 81] Bernstein, P. A., Goodman, N., Wong. E., Reeve, C. L. and Rothnie, Jr., J. B.: Query Processing in a System for Distributed Databases (SDD-1), ACM TODS, Vol. 6, No. 4,

pp. 602-625 (1981).

[CePe 84] Ceri, S. and Pelagatti, G.: Distributed Databases Principles and Systems (book), 393 p., McGraw-Hill Book Co. (1984).

[ChRi 82] Chan, A. and Ries, D. R.: Distributed Database Management Research at Computer Corporation of America, IEEE Bulletin on Database Engineering, Vol. 5, No. 4, pp. 14-19 (1982).

[CoDa 74] Codd, E. F. and Date, C. J.: The Relational and Network Approaches: Comparison of the Application Programming Interfaces, Proc. of the Workshop on Data Description, Access and Control, pp. 83-113 (1974).

[Dani 82] Daniels, D.: Query Compilation in Distributed Database System, Research Report, RJ 3423, IBM Research Laboratory, San Jose, CA (1982).

[EGLT 76] Eswaren, K. P., Gray, J. N., Lorie, R. A. and Traiger, I. L.: The Notions of Consistency and Predicate Locks in a Database System, Comm. ACM, Vol. 19, No. 11, pp. 624-633 (1976).

[FeSt 82] Ferrier, A. and Stangret, C.: Heterogeneity in the Distributed Database Management System SIRIUS-DELTA, Prtoc. 8th VLDB Conf., pp. 45-53 (1982).

[HiKH 85] Hikita, S, Kawakami, S. and Hanyuda, H.: A Stepwise Approach to Distributed Database Systems by Database Machine, Proc. 1985 ACM SIGSMALL Symposium on Small Systems, pp. 18-24 (1985).

[IBM 82] SQL/Data System Application Programming, SH 24-5018-1, IBM, 327 p. (1982).

[ISO 86] Remote Database Access Service and Protocol, Draft paper, ISO/TC 97/SC 21/WG 3 (1986).

[KaKo 85] Kambayashi, Y. and Kondoh, S.: Integration of Concurrency Control Mechanisms, Proc. of the 31st Annual Convention of Information Processing Society of Japan, 2B-4 (1985) (in Japanese). (上林弥彦，近藤誠一：並行処理制御方式の統合，情報処理学会第31回(昭和60年後期) 全国大会論文集，2B-4).

[KaYY 82] Kambayashi, Y., Yoshikawa, M. and Yajima, S.: Query Processing for Distributed Databases Using Generalized Semi-Joins., Proc. ACM SIGMOD Conf., pp. 151-160 (1982).

[KuRo 81] Kung, H. T. and Robinson, J. T.: On Optimistic Methods for Concurrency Control, ACM TODS, Vol. 6, No. 2, pp. 213-226 (1981).

[LHM. 83] Lindsay, B., Hass, L., Mohan, C., Wilms, P. and Yost, R.: Computation and Communication in R*: A Distributed Database Manager, Research Report, RJ 3740, IBM Research Laboratory, San Jose, CA (1983).

[Masu 84] Masunaga, Y.: Recent Technology of Distributed Relational Database Systems in U. S. A., Journal of Information Processing Society of Japan, Vol. 25, No. 5, pp. 443-450 (1984) (in Japanese). (増永良文：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 (1984)).

[Masu 85] Masunaga, Y.: Management of Table Partitioning and Replication in a Distributed Database System, Proc. International Conference on Foundations of Data Organization, pp. 93-103, Kyoto, Japan (1985).

[Masu 86] Masunaga, Y.: Distributed and Multimedia Databases, The 12th VLDB Conf., Tutorials, pp. 33-56 (1986).

[Mc-L 81] McLean, Jr., G.: Comments on SDD-1 Concurrency Control Mechanisms, ACM TODS, Vol. 6, No. 2, pp. 347-350 (1981).

[MoYS 85] Mori, M., Yoshida, K. and Suzuki, K.: Development of a Distributed Database Management System, Review of the Electrical Communication Laboratory, Vol. 33, No. 3, pp. 443-449, Nippon Telegraph and Telecommunication (1985).

[Munz 79] Munz, R.: Gross Architecture of the Distributed Database System VDN, Proc. IFIP TC 2 Conf. on Database Architecture, Venice (1979).

[Naum 82] Nauman, J.: ENCOMPASS: Evolution of a Distributed Database/Transaction System, IEEE Bulletin on Database Engineering, Vol. 5, No. 4., pp. 37-41 (1982).

[Ng 82] Ng, P.: Distributed Compilation and Recompilation of Database Queries, Research Report, RJ 3375, IBM Research Laboratory, San Jose, CA (1982).

[RaSL 82] Rahimi, S. K., Spinrad, M. D. and Larson, J. A.: A Structural View of Honeywell's Distributed Testbed System: DDTS, IEEE Bulletin on Database Engineering, Vol. 5, No. 4, pp. 47-51 (1982).

[RBF. 80] Rothnie, Jr., J. B., Bernstein, P. A., Fox, S., Goodman, N., Hammer, M., Landers, T. A., Reeve, C., Shipman, D. W. and Wong, E.: Introduction to a System for Distributed Databases (SDD-1), ACM TODS, Vol. 5, No. 1, pp. 1-17 (1980).

[RoKa 86] Roussopoulos, N. and Kang, H.: Preliminary Design of ADMS±: A Workstation-Mainframe Integrated Architecture for Database Management Systems, Proc. 12th VLDB Conf., pp. 355-364 (1986).

[SAC. 79] Selinger, P. G., Astrahan, M. M.,

Chamberlin, D. D., Lorie, R. A. and Price, T. G.: Access Path Selection in a Distributed Relational Database Management System, Proc. ACM SIGMOD Conf., pp. 23-34 (1979).

[SCF. 86] Schwarz, P., Chang, W., Freytag, J. C., Lohman, G., McPherson, J., Mohan, C. and Pirahesh, H.: Extensibility in the Starburst Database System, IBM Research Report, RJ 5311 (Sep. 1986).

[StNe 77] Stonebraker, M. and Neuhold, E.: Distributed Data Base Version of INGRES, Proc. of the 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, pp. 19-36 (1977).

[Taki 83] Takizawa, M.: Distributed Database System-JDDBS, JARECT, Vol. 7, Computer Science and Technologies, Edited by Kitagawa T., Ohmsha and North-Holland, pp. 263-283 (1983).

[TAY. 85] Tezuka, M., Adachi, S., Yamane, Y., Nakada, T., Take, R. and Okazaki, T.: Homogeneous Distributed Relational Database System RDB/DV, Paper of Working Group on Database Systems, Information Processing Society of Japan, WGDBS 47-3 (May, 1985) (in Japanese). (手塚正義, 安達 進, 山根康男, 中田輝生, 武理一郎, 岡崎 卓: 関係型均質分散データベースシステム RDB/DV, 情報処理学会研究会報告, 85-DB-47-3 (昭和 60 年)).

[TsKl 78] Tsichritzis, D. and Klug, A.: The ANSI/X 3/SPARC DBMS Framework Report of the Study Group on Database Management Systems, Information Systems, Vol. 3, pp. 173-191 (1978).

[WDH. 82] Williams, R., Daniels, D. Hass, L., Lapis, G., Lindsay, B., Ng, P., Obermarck, R., Selinger, P., Walker, A., Wilims, P. and Yost, R.: R*: An Overview of the Architecture, Proc. International Conference on Database Systems, Jerusalem, Israel, pp. 1-27 (1982).

(昭和 61 年 12 月 11 日受付)