S.12-5 データベースに於ける分散

田中英彦（東京大学工学部）

1. はじめに

従来，個々の業務別に作られていたファイルを統合し種々の業務から共用できるようにして重複を避け，その作成と保守を集中化することによって能率の向上を計ることを目的として様々なファイルが整理されて来ている．このような共用ファイルを一般にデータベースと呼んでおり，様々な分野での利用が広がって来ているが，データ通信の発達，コンピュータネットワークの発展とともに更にその発展形態として分散形のデータベースが云々されるようになって来た．分散データベースと言う場合，普通2通りのものを指して言っている．一つは，従来のファイルシステムを構成している要素ファイルが地理的に分散した形の分散ファイルであり，他は各々独立したデータベース管理システムが相互に結合された形のシステムである．

データベースを構成する場合一般に考慮するべき事項はデータと応用プログラムとの独立性（data independence），データベース作成及び利用上の言語としてのデータ記述言語（DDL），データ操作言語（DML）のサポート，データベースの信頼性及びデータ保護対策等である．分散データベースについてもそれは同様であるが，分散したファイルで一つのデータベースを作りあげたり諸データベースを論理的に結合し統合する為には，それ特有の問題があり解決せねばならないことが多い．ここでは分散データベース一般の目的，利点をあげるとともに，その構成形態，技術及び分散データベースの幾つかの実現例について述べる．

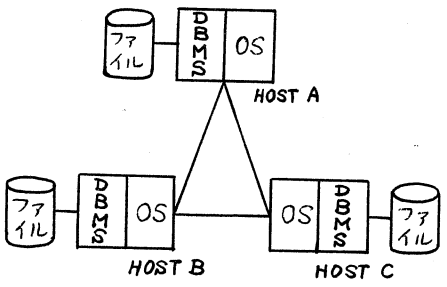

図1 分散データベース

2. 分散データベースの目的，利点

分散データベースを構成する場合の動機はシステムによって様々でありその目的も異なるが，一般に考えられている事柄は以下の通りである．

①データリソースの共有：大規模で特徴のあるデータベースを重複設置することはコスト的にも時間的にも不経済である．これは普通のデータベース自体の目的でもあるが，特に分散形の場合重要である．

②データアクセスの偏りの利用による経済化：各ファイルに対するアクセス頻度が地域的に偏在している場合，ファイルを分散設置する方が経済的である．

③各設置場所のニーズに対する適応性が高い：複数の事業所それぞれに適合したシステムを設けることが可能で，構成，保守，変更に適応し易く，又最初は独立したシステムから出発し徐々に機能を強化して回線で相互に結合し，論理的に結合して分散データベースを作りあげてゆくと言った発展的な設計が可能である．

④システム全体の性能の向上：複数のプロセッサを設けることによって同時処理によるスループットが向上する．特に，データの発生現場で処理をすることにより通信コストの低減が望め，応答性の良いシステムができ，処理の明確な機能分担が可能となる．また，プロセッサの台数を漸次増すことによりシステム全体の性能を必要に応じて増強できる．

⑤システム運用上の信頼性向上：大容量のデータを余りにも集中化することは，システムの信頼性の上から望ましくない．また，データ内容を保護しsecurityを確保することは各所のニーズに合わせて調整でき，より高い信頼性を持たせることが可能となる．

その他，行政情報システム，流通情報システムなど，特に全国規模にまたがるようなシステムからの要望があろう．

3. 分散データベースの構成形態

各要素データベース相互間の関係によって分散データベースを分類すれば次の3種が考えられる．

①内容複写による分散データベース：図2のようなもので，これは銀行システム，販売在庫管理システム等にその例が見られる．最上位システムは顧客全部のアカウントマスタデータベース

を持ち，それを幾つかに分割してそれぞれが受持つ．衛星プロセサはそのローカルな顧客のアカウント用データをマスタからコピーして持っている．このローカルなデータベース上で毎日の取引がおこなわれ夜の内に最上位へ報告し，マスタを更新して新たな翌日のマスタデータベースが作られる．

②アクセスパタンにより分割された分散データベース：基本的には複写を持たず，各要素ファイルに対して各プロセサから出されるアクセス要求の頻度に従って全ファイルの分割がおこなわれる．一般に要素システム相互は対等である．

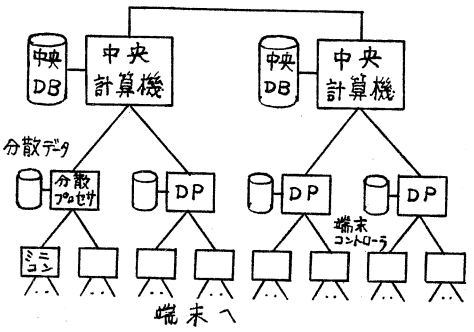

図2　コピーによる分散DB

③組織管理体制により分割された分散データベース：会社の事業体毎，又は会社毎に作られたデータベース相互を結合したものがこれに相当する．一般のシステムでは，処理の能率を考えてファイルのコピーをどの程度重複して持たせるかが問題となるし，DBMSを複数設ける場合はDBMS相互間の制御が必要で，システムが同種であるか否かにより結合の難かしさ及び適切なその程度も異なる．個々に発展したデータベースを後程結合する場合は普通③の形になり，DBMSは複数存在する．

4．分散データベースの構成技術

分散データベースを構成する技術は，一般に，各要素システムが同種のOSを持つ場合，分散したファイルを統合してユーザには単一に見せる為の分散ファイル構成技術と，各要素システム相互が異なる場合，更に出てくる問題としての異種システム相互の両立性技術とに分けられる．

4．1　分散ファイルの構成技術

①各システム内の構成：ユーザからのファイルアクセス要求は，まずそれがローカルファイルに対するものであるか否かの分類がおこなわれローカルである場合はそのファイル管理に引続かれるが，リモートファイルに対する要求である場合は一旦その要求は網標準のコマンド言語に変換され，計算機間通信機構（HOST/HOSTプロトコル，プロセス間通信機構）を経て相手のシステムに要求が送られる．従って，網標準言語の設定とそれを解釈実行する為のプロトコルを定める必要がある．この言語設定のレベルを普通のファイルアクセスコマンドのレベルで設けるかより高レベルのものにするかは応用によって異なり，一般に前者の方がきめ細かな制御ができるがシステム間の両立性に対する制約がきびしい．前者の例としてはARPANETのファイル転送プロトコルがあり，アクセス制御コマンド，転送パラメータコマンド，ファイル転送サービスコマンドの3種が設けられているが，ファイル構造としては無構造かレコード構造しか扱えずアクセスもシーケンシャルに限られている．これをより一般化した例としてはR. Peeblesのシステム提案[3]がある．

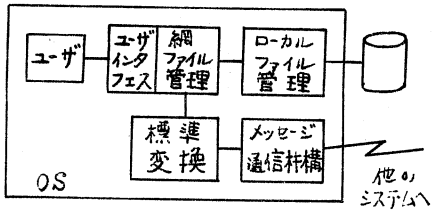

図3　各システム内構成

②ファイルディレクトリの構成：ディレクトリの構成法としては一般に，全ファイルのディレクトリを一ヶ所に集中するか，各ファイルのネームにその置かれた場所のネームを付けて用いるか，何階層かのテーブルを各所に設けるか，ローカルファイル以外のディレクトリは持たないかの何れかが考えられる．一ヶ所に置く方法は，各データベースの管理組織が異なると問題であるし，又大規模になると量的にも信頼性の上からも困難となる．場所ネームを併用する方法は最も単純であるが，このままではユーザに取って不便であり何らかのユーザサポートが必要である．リモートファイルに対してディレクトリを設けない方法は各システムに対して問合せを行うものであるが，一般にはオーバヘッドが大きくなり過ぎる恐れがある．従って，分散ファイルに対するディレクトリ構成としては図4のようなものが考えられる．これは何段かのテーブルを引くことによって求めるファイルが置かれたデバイスを求めるもので，網上でユニークに付けられるファイル名としては，HOST名とOwner名，及びオーナが付けたファイル名（OFN）の組み合わせを用いている．

③デッドロック対策：複数のプロセスがリソースを取り合う場では互いにリソースが空くのを待

つことによってデッドロックが生じる可能性があり，特にデータの保全性を保つ為に更新モードで用かれたレコードをそのユーザに専有させることにするとデッドロックの可能性が増す．分散形システムではリソースの量が増す他，それら各々の状態を調べる為のオーバヘッドが増し，リソース割当てをデッドロックフリーに行う回避対策の実装が非常に困難になる．従ってその対策としては，まずデッドロックを考慮せねばならないリソースをできるだけ限定すること，次に動的なリソース割当ては行なわずプロセス走行開始時にのみおこなうか，動的に行う場合はプロセスにチェックポイントを設け，デッドロックの発生をタイマ等で調べて検出をおこない，発生しておればどれかのプロセスのリソースを取り上げてそのプロセスの処理をチェックポイント迄戻す方法等が現実的であろう．動的な割当てをおこなう場合，各プロセスが将来必要とするであろうリソースが既知でないとデッドロックの発生を防ぐことは困難で，検出法に依るしかない．リソースが既知である場合はプロセス走行前にそれらをすべて確保するか，又はリソース要求が出た時点でそのリソースを割当てた場合の将来のデッドロック可能性をテストし，可能性がない場合に限り割当てをおこない，可能性がある場合には割当てを保留するかのどちらかである．静的にリソースを割当てる場合でも分散システムではデッドロックが無い訳ではなく，リソース確保の順番（プロセサの順番，及び各計算機内でのリソースの順番）を網上で唯一に定めておくか，又は確保の途中段階でリソース待ちに入ればそれ迄に確保済みのリソースすべてを解放してから待ちに入る等の考慮が必要である．

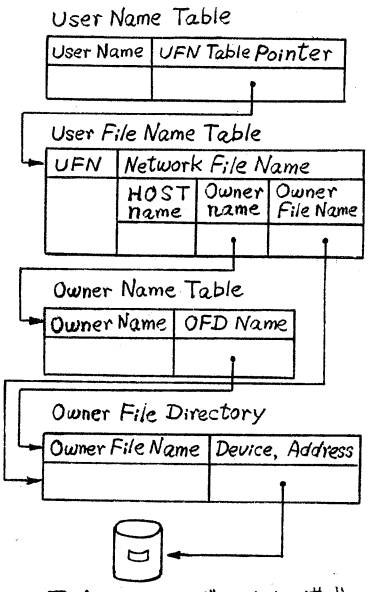

図４　ファイルディレクトリの構成

④ファイル分割と配置：計算機システムという入れ物だけが分散していた場合に，数多くのファイルをそれぞれどこに設備すれば最も経済的で能率が良いかという問題である．実際上これは各計算機システムの構成，OS 等によって定められることが多いが，同種の計算機によって構成されているような網の場合は現実に問題となる．この問題は線形計画や動的計画の問題として定式化できるが，その時考慮するべき事項としては，各ファイルへのアクセスパタンと頻度，プログラムファイルかデータファイルかの種別，そのファイルが使える計算機種，回線トポロジーと距離及び回線速度，各計算機上でのファイル蓄積コスト等である．プログラムファイルは特に走りうる計算機が限定されるし，又それが定まるとそのプログラムが使うデータファイルに対するアクセス頻度にも影響する．

４．２　異種システムの両立性技術

異種計算機システムからなる分散データベースに於いて特に問題となる点は，システム間でのデータ構造，データ操作コマンドが異なること，及びより応用プログラムに近い所から見れば複数のデータベース応用システムの利用手続きが異なること等である．

①データ構造の定義と変換：データ構造の差とは，語長，コード，フィールド位置，ブロック長，ブロック形式などの物理的な差の他，データ内部表現やファイル内データ相互の関係などの論理的な差のことで，システムが異なるとこれらは一般に異なる．これらのデータ構造は一般に，データベース（ローカルな）全体の記述（スキーマ）と，各応用プログラムの側から見たデータベースの記述（サブスキーマ）とで定まり，前者は更にデータの論理的なモデルの記述と物理的な記憶構造の記述に分けて考えることができる．システム間の差で問題になるのはこれらの記述がまちまちなことで，それがユーザ任せになっている初歩的なものから，それぞれのDDL を設定したもの迄様々である．これに対処するには，応用プログラムで個々にソフトウエアインタフェスを作るか，間にデータ変換システムを置き必要に応じて変換プログラムを実行させるか，共通のデータ記述言語を設定してスキーマの記述をおこない，OS にはサブスキーマ情報交換用のプロトコルを設定することによってデータベースのリモートアクセスを可能にするかの３通りが考えられる．根本的な対策は最後のものであろうが，共通のDDL は未だ

定まってもいず（ISO では現在の所，ANSI/SPARC 報告の方向が有望），これは将来の研究課題であろう．データ変換システムはこれらの妥協の産物であり，現実的にはARPANETのデータ再編サービスとして使われている他，それらをより拡張した形のFile Translation Language Processorの提案[3]がある．また，より高レベルでのデータ構造変換用の言語としてはIBM の CONVERT がある．

②データ操作言語とアクセス法：データベースの利用者はそれぞれのプログラム言語を介してデータベースを操作するが，このデータ操作言語をDBMSの中で実際に実行するのがアクセス法である．遠隔のデータベースを利用する場合DMLが異なればユーザに取って使い難いので，理想的には全システムが同一のDMLを用い，各システム内でのアクセス法を仮想化してローカルファイルへのアクセスとリモートファイルのアクセスとの区別を見えなくすることによって，より完全な意味での分散データベースが実現されよう．DMLとしては親言語（COBOLやFORTRAN）から使われるものとそれだけで自己完結形のものがある．

③コマンド言語：プログラミング言語のみならず種々のコマンド言語の非両立性も問題である．その最も良い例がジョブ制御言語であるが，システムの数が増すとユーザの負担が重くなり，これに対処するには一つの共通コマンド言語を設定してそれをユーザインタフェースとし，ユーザと分散システムとの間に各システムへ送る個々のコマンドへの変換をおこなう装置を設けることが考えられる．このような方向での研究としては，NBS の Network Access Machine，Rand の Rand Intelligent Terminal Agent，MIT の Connector for Networked Information Transfer などがある．

④応用の内容レベルでの両立性：異なったデータベースを用いた同種の応用システムを統合してより便利な仮想システムを作りあげる場合に出て来る問題である．例えば複数の情報検索システムを統合する場合，コマンド言語，インデクス語彙，カタログデータ構造等の統合が必要になるが，その為にはインデクス語相互間の関係（同義語，広義語，関連語等）が必要になったり，各システムのデータ要素を更に分解して基本データ要素としそれらの組み合わせとして全データを見た場合の構造が必要になったりする．

5．分散データベースのシステム例

分散データベースとして今までに述べて来た機能をすべて備えた本格的なシステムは未だ存在しないが，各組織体の目的に合わせてデータベースを特定の専用的な利用形態に限ったシステムとか，同一機種同一OS 間での分散ファイルシステムとかは存在する．前者の例としては販売在庫管理，保険業務，銀行業務等に使われている複写形の分散データベースがあるし，又その他特徴的なものとしては前述のCONITがある．これは複数の情報検索システムを結合したもので，ARPANETやTYMNETを用いMITのIntrex，Lockheed DIALOG，SDC ORBIT，NLM MEDLINE 等のデータベースを結合して一つの仮想的な情報検索システムとすることを狙ったもので，MITのMULTICS の上に実験用インタフェースが作られており図5のような構成をしている．又，同一機種間の分散ファイルとして有名なものはBBN 社の Resource Sharing Executive である．これはARPANET上のPDP-10 をHOST としTENEX OSを備えたシステムを論理的に結合するオペレーティングシステムで，各システム内のファイルアクセストリーを拡張し，ファイル名の前に計算機の場所名を付けることによって全ファイルをアクセス可能としたものである．その他，諸システムからの共有データベースとなることを目指したものにはBTL の XDMS，Computer Co. of America の Datacomputer がある．後者はARPANETに接続されておりPDP-10 で制御されているが，Ampex 社の大容量記憶システム Tera Bit Memory を備えており Datalanguage と呼ばれる高水準言語によってデータベースにアクセスをする．

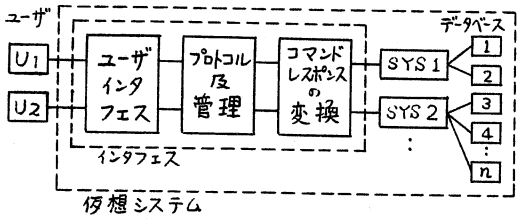

図5 CONIT の仮想インタフェス

（参考文献）

(1) 分散処理計算機システムの動向調査－新技術動向調査報告－，日本電子工業振興協会，Vol. 52-C-334，pp. 70-96，昭和52年3月．
(2) Richard W. Peebles, Ph.D Dissertation, Univ. of Pennsylvania 1973.
(3) G. M. Schneider, Ph.D Dissertation, Univ. of Wisconsin 1974.