# 分散データベース用ファイル管理システム

本田公男，田中英彦，元岡　達（東大・工）
堀口真志（日立），青木栄一（富士通）

## 1．はじめに

計算機網の目的の一つに資源――特殊なハードウェア，各地に蓄積されているソフトウェア，データ――の共有があげられる．これらの資源のうち，データの共有を実現するための分散ファイルシステムの一方式を提案する．このシステムは，網上に分散したすべてのファイルについて，ユーザに網を意識させない共通のアクセス法を提供することをその主たる目的としており，さらにこのシステムを利用して分散データベースを実現することをめざしている．

現在このシステムの第一版は研究用計算機網ＴＥＣＮＥＴに実装されており，ひきつづき，システム内に重複データを置き，信頼性，応答時間の向上をめざした第二版を開発中である．

本発表ではこのシステムの位置づけ，第一版の構成法，第二版の変更点を中心に述べる．

## 2．ＴＥＣＮＥＴの網向きＯＳ（ＮＯＳ）とプロセス間通信

研究用計算機網ＴＥＣＮＥＴの構成を図１に示す．

ＴＥＣＮＥＴの網向きＯＳ（ＮＯＳ）は，各ホスト内においてマルチプロセス処理を実現し，さらにプロセス同志が網を意識することなく情報を交換する手段としてプロセス間通信を提供している．プロセス間通信を利用するときは，各プロセスは，

(1) 相手プロセスのホスト番号，プロセス番号

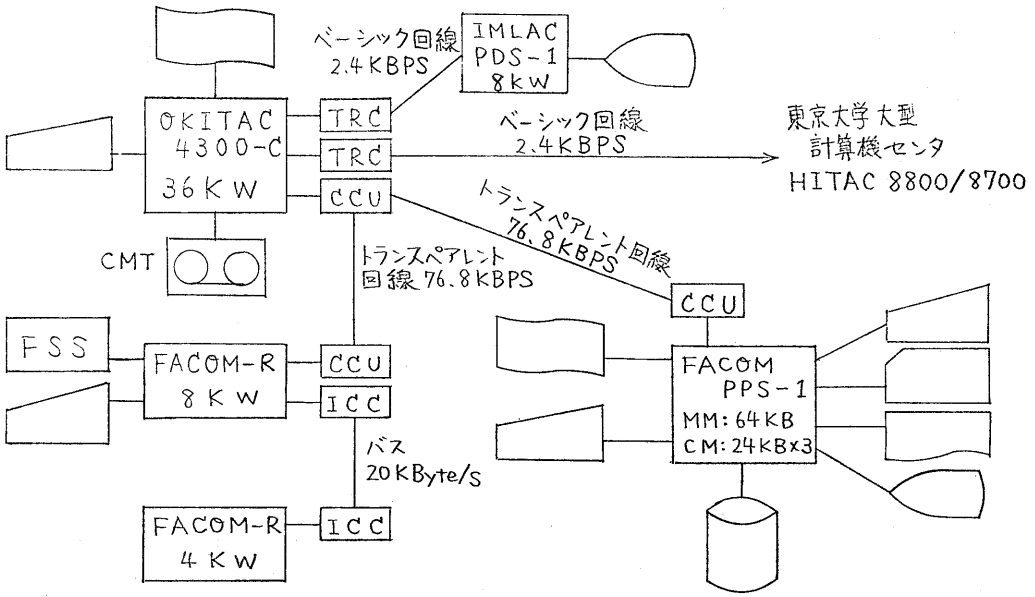



図１．ＴＥＣＮＥＴの構成

CCU: Communication Control Unit
TRC: Transmission Control
ICC: Intercommunication Controller

(2) メッセージ(バッファ)の先頭アドレス，バイト数
(3) ブロック指定(通信が終了するまで待つか否か)
を指定してＳＥＮＤまたはＲＥＣＶプリミティブを実行すればよい．この手続きは相手プロセスがリモートでもローカルでも同じである．

## 3．分散データベースと分散ファイルシステムの関係

一般にデータベースを構成するには，データベース管理システム(ＤＢＭＳ)が直接データにアクセスする方法(図2．(a))と，まずファイル管理システムを作成し，ＤＢＭＳはそれのアクセス法を利用する方法(図2．(b))の2通りが考えられる．さらに後者の方式を分散データベースに適用した場合，網上の各地にそれぞれローカルなファイル管理システムを置く方法(図2．(c))と，分散ファイルシステムを作る方法(図2．(d))が考えられる．

(c)の方式ではＤＢＭＳが網を意識し，各地のローカルなファイル管理システムを利用するが，(d)の方式では分散ファイルシステムが，網を意識せずに網上のすべてのファイルにアクセスできるアクセス法を提供しているので，ＤＢＭＳは網をほとんど意識しなくてよい．したがって，(d)の方式には，従来の(集中形)データベースのDBMSをほとんどそのまま用いることができる長所がある．

## 4．分散ファイルシステムの設計方針

分散ファイルシステムの設計に際しては以下の方針に従った．
(1) すべてのホストが平等な権利を持つ分散管理方式を採用する．
(2) ユーザの立場から見て，ファイル

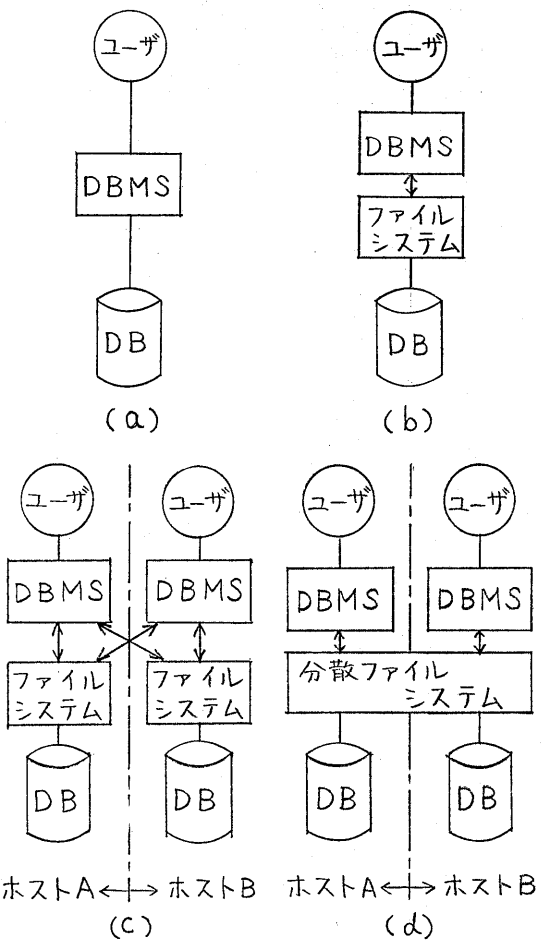



図2．データベースの構成法

をアクセスする際に網を意識する必要がない．すなわち網上のすべてのファイルが同じ手続きでアクセスできるという点に重点を置く．そのためにプロセス間通信の機能を最大限に利用する．
(3) ユーザが，自分の使用するファイルについては，自分の所有しているファイルでも，他のユーザに共用させてもらっているファイルでも，自由に名前をつけて呼ぶことができるようにする．
(4) リモート通信のオーバヘッドができるだけ小さくなるように考慮する．
(5) デッドロック対策および機密保護

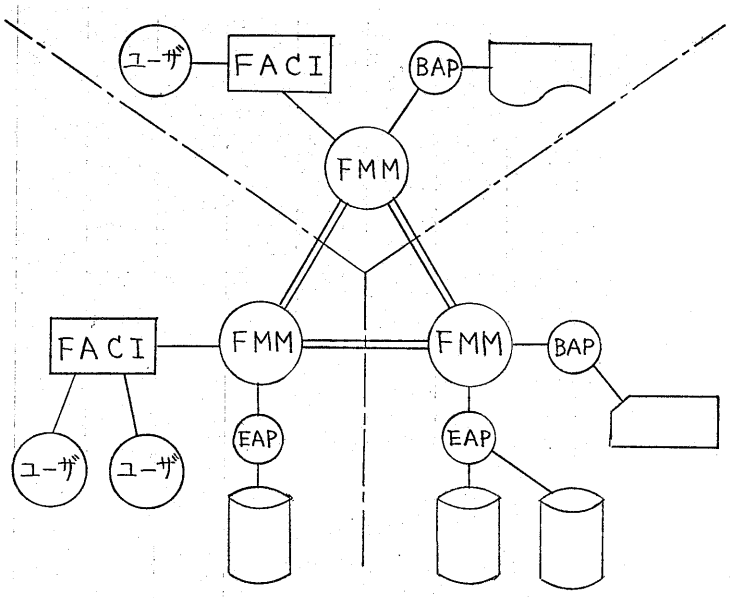



図３．分散ファイルシステムの構成

についても，実用上問題とならない程度に考慮する．

(6) 将来，分散データベースを実装することを考慮した構成をとる．

## ５．分散ファイルシステム（第一版）の構成

### ５．１ 構成

分散ファイルシステム（第一版）の構成を図３に示す．

(1) ＦＭＭ（File Main Manager）

ＦＭＭは，各ホストに１つあるプロセスであって，そのホストにあるファイルを一括管理している．ここでいうファイルとは，ディスク，磁気テープのファイルだけでなく，入出力機器も含んでいる．また，FMMが他のＦＭＭと通信することによって，網全体としてのファイル管理が行なわれるようになっている．

ＦＭＭの役割を以下に示す．

(i) 資源としてのファイルの確保，解放

(ii) ファイルの新規作成，消去

(iii) ファイルのオープン，クローズの管理

(iv) ディレクトリ管理

(2) ＢＡＰ（Basic Access Process）

ＢＡＰは各ホストのファイル編成ごとに設けられるプロセスで，ユーザプロセスと通信を行ない，その要求に従って実際にファイルにアクセスする．

(3) ＥＡＰ（Extended Access Process）

ＢＡＰはファイルのRead, Writeのような基本的な機能のみを行なうものであるが，これだけではたとえばリモートにあるファイルを複写したり，ＤＢＭＳがリモートにあるデータの検索を行なうような場合，通信のコストが非常に大きくなる．そこでＥＡＰというプロセスを設け，これに拡張コマンドを送ることができるようにする．ＥＡＰが処理するマクロコマンドとしては，ファイルの複写，比較の他にデータの検索や更新等を用意して分散データベースの便宜を図る．

(4) ＦＡＣＩ（File Access Command Interpreter）

ＦＡＣＩは一般のユーザプロセスのために設けられたファイルアクセスのためのリエントラントなサブルーチン群である．

## 5．2　ディレクトリ

分散ファイルシステムのためのディレクトリ構成法として，たとえば以下の方法が考えられる．

(1)　集中ディレクトリ
　網内のすべてのファイルのディレクトリを一ホストだけに置く．

(2)　分散ディレクトリ
　網内のすべてのファイルのディレクトリをすべてのホストに置く．

(3)　部分ディレクトリ
　各ホストに，それぞれそのホストにあるファイルのディレクトリを置く．

(1)は，一回の通信でファイルの格納位置を知ることができ，また集中管理であるため管理が能率的に行なえるという利点があるが，特定ホストの負担が増大し，またそのホストのシステムダウンが網全体に及ぶという欠点がある．

(2)は，通信を行なわずにファイルの格納位置を知ることができ，あるホストのシステムダウンが他のホストに及ぼす影響も小さいが，ディレクトリ更新の際の通信のオーバヘッドが大きい．

(3)は，あるホストのシステムダウンの影響は小さく，更新のオーバヘッドも小さいが，使用したいファイルが自ホストにない時は他のホストを尋ねて回らねばならないため，通信のオーバヘッドが大きくなる．

このような点を考慮して，このファイルシステムでは，図4に示す2階層テーブル方式と呼ばれる方式を採用した．この方式は部分ディレクトリの上に，ファイルの格納位置検索の能率を向上させるテーブルを設けた方式と考えられ，部分ディレクトリの欠点を解消している．以下，この方式について説明する．

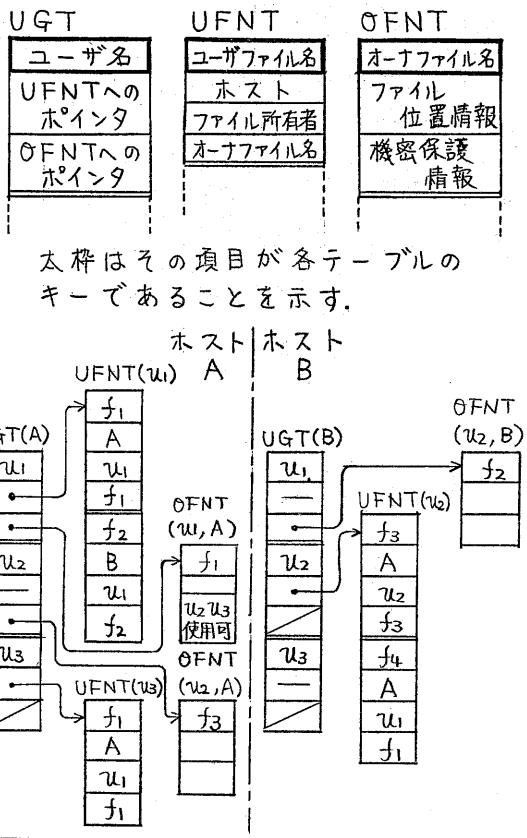
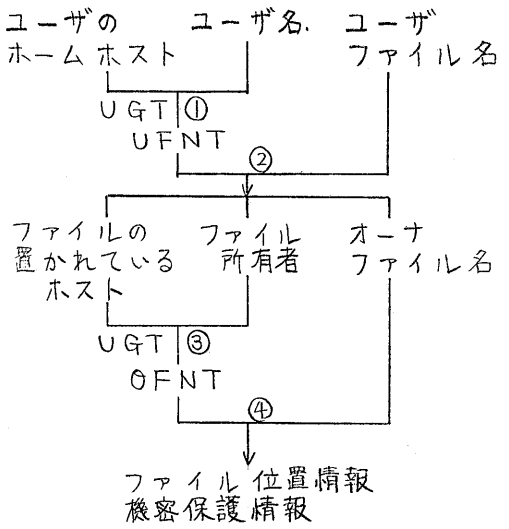



図4．ディレクトリの構成

ディレクトリの情報は次の3種のテーブルにおさめられている．

(1)　UFNT（User File Name Table）
　ファイルの利用者がつけた名前（以下ユーザファイル名という）から，ファイルの置かれているホスト，ファイルの所有者およびファイルの所有者がつけた名前（以下オーナファイル名という）を得るための表である．各ユーザごとに網内にただ1




① ユーザのホームホストのUGTをユーザ名をキーとしてサーチし，UFNTを見つける．
② UFNTをユーザファイル名をキーとしてサーチし，ファイルの置かれているホスト，ファイルの所有者およびオーナファイル名を求める．
③ ファイルの置かれているホストのUGTをファイル所有者をキーとしてサーチし，OFNTを見つける．
④ OFNTをオーナファイル名をキーとしてサーチし，ファイルに関する情報を得る．

図5．ディレクトリ検索の手続き

つあり（UFNTの置かれているホストのことをそのユーザのホームホストと呼ぶことにする），使用するファイルごとに1エントリある．

(2) OFNT（Owner File Name Table）
オーナファイル名から，そのファイルの格納されているデバイス，ボリューム等の位置情報および機密保護情報を得るための表である．各ファイル所有者，各ホストごとに1個設けられ，所有しているファイルごとに1エントリある．

(3) UGT（User Group Table）
UFNT，OFNTの検索の効率を考慮して設けられたもので，各ホストに1個ある．

このディレクトリから，希望するファイルの情報を得るための手続きを図5に示す．

5．3 プロトコル

ユーザプロセスがファイルを使用する際にファイルシステムに対して出すコマンドを以下に列挙する．

(1) GETFN（ユーザプロセス⇄ユーザのホームホストのFMM）
UFNTによって，ユーザファイル名からファイルの置かれているホスト，ファイルの所有者およびオーナファイル名を得る．

(2) OBTAIN（ユーザプロセス→FMM$_1$→FMM$_2$→…→FMM$_k$→ユーザプロセス）
ファイルの確保を行なう．また新規にファイルを作成する．

(3) FREE（ユーザプロセス→FMM$_1$→FMM$_2$→…→FMM$_k$→ユーザプロセス）
OBTAINコマンドによって確保されたファイルを解放する．また同時にOFNTへの登録，OFNTからの削除，ファイルの消去をも行なう．

(4) OPEN，CLOSE（ユーザプロセス⇄ファイルの置かれているホストのFMM）
ファイルのオープン，クローズを行なう．

(5) CATALOG，UNCATALOG（ユーザプロセス⇄ユーザのホームホストのFMM）
UFNTへの登録，UFNTからの削除を行なう．

(6) PERMIT，INHIBIT（ユーザプロセス⇄ファイルの置かれているホストのFMM）
他のユーザにファイルのアクセスを許可あるいは禁止するという情報

をＵＦＮＴに書きこむ．

(7) ＲＥＡＤ，ＷＲＩＴＥ（ユーザプロセス⇄ＢＡＰ）
ファイルのRead，Writeを行なう．

## ５．４ ファイルの確保および解放の手順

この分散ファイルシステムではデッドロック防止策として，実装の容易さからデッドロック回避法のうちの単純法を採用した．そのために，デッドロック対策が必要なファイルは，すべてジョブステップの最初で確保しておくことになる．その手順は次の通りである．（図６参照）

(1) ユーザ（実際はジョブ管理システムが代行）が必要なファイルの一覧表を作る．

(2) このうち，ユーザファイル名により指定されているファイルは，ＧＥＴＦＮコマンドによってUFNTをひいてファイルの置かれているホスト，ファイルの所有者およびオーナファイル名を得る．（図の①②③）

(3) ＯＢＴＡＩＮコマンドによりファイルの一覧表を網上の各ＦＭＭに回覧させる．（図の④⑤⑥⑦）この際，デッドロックを防ぐため，回覧の順序を決めておく．各ＦＭＭは，ファイルの一覧表のうち，自分の管理しているファイルを確保し，一覧表を次のＦＭＭに送る．最後に回覧したＦＭＭはユーザに応答を送る．
他のユーザが使用中であるために確保できないファイルがあった場合は，ユーザの指定に従って確保できるまで待つか，ただちにユーザに通知するか（図の⑥'）する．

(4) ファイル解放の手順は，ＧＥＴＦＮの手続きが不要なだけで，確保の手順とほとんど同じである．

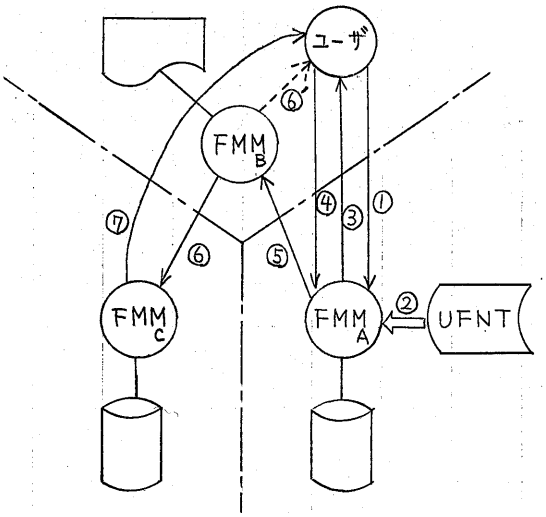



図６．ファイルの確保・解放の手順

## ５．５ ファイルアクセスの手順

確保されたファイルをユーザがアクセスするときの手順を以下に述べる．（図７参照）

(1) ユーザプロセスが（ＦＡＣＩのＯＰＥＮルーチンを通じて）ファイルの置かれているホストのＦＭＭにＯＰＥＮコマンドを送る．（①②）

(2) ＦＭＭはＢＡＰにファイルのオープンを指示する．（③）

(3) ＢＡＰはファイルをオープンし，ＦＭＭに応答を返す．（④⑤）

(4) ＦＭＭはユーザプロセスに応答を返す．このときユーザプロセスにＢＡＰのプロセス番号が知らされる．（⑥⑦）

(5) 以後はユーザプロセスが（FACIのＲＥＡＤ，ＷＲＩＴＥルーチンを通じて）ＢＡＰと通信することによって，ファイルのRead，Writeが行なわれる．（⑧～⑫）

(6) クローズの手順は，オープンの手順とほぼ同様である．（⑬～⑲）

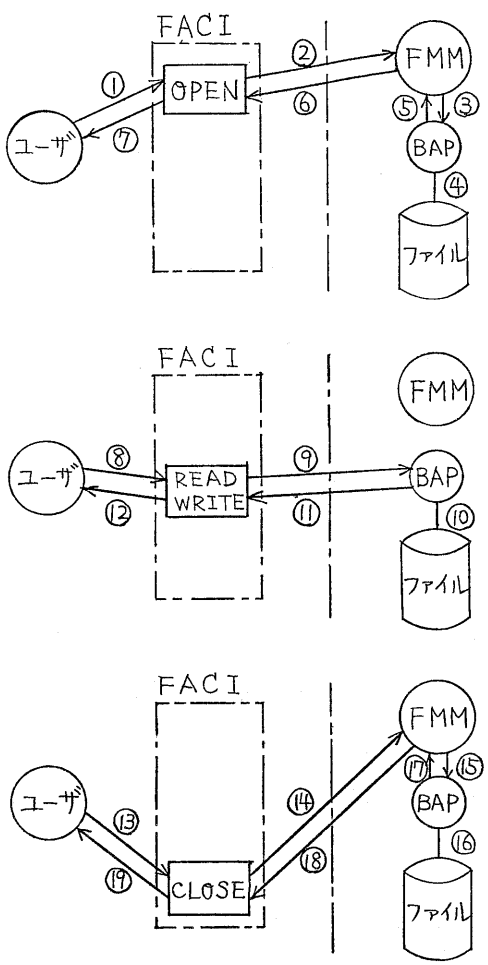



図7．ファイルアクセスの手順

5．6　ディレクトリへの登録およびディレクトリからの削除の手順

(1)　自分のファイルの場合

(i)　ユーザが，ファイルの置かれているホストのＦＭＭにＯＢＴＡＩＮコマンドを送る．

(ii)　ユーザが，ファイルの置かれているホストのＦＭＭにＦＲＥＥコマンドを送り，同時にＯＦＮＴへの登録を指定する．

(iii)　ユーザが，ホームホストのFMMにＣＡＴＡＬＯＧコマンドを送る．（このとき，ファイル所有者＝ユーザ，ユーザファイル名＝オーナファイル名とする）ＦＭＭはこれをＵＦＮＴへ登録する．

(2)他人のファイルの場合

(i)　ユーザが，ホームホストのＦＭＭにＣＡＴＡＬＯＧコマンドを送る．ＦＭＭはこれをＵＦＮＴに登録する．

以上ファイルをディレクトリに登録する方法について説明したが，ディレクトリからの削除についてもほとんど同様である．

## 6．分散ファイルシステム（第二版）の構成

### 6．1　分散データベースにおける重複データ

分散データベースにおいては，ファイルの信頼性，通信コストの低減，応答時間の短縮などの点からデータに重複を認めることが重要だといわれている．現在開発中の分散ファイルシステム第二版では，重複データを置き，性能改善をめざしている．

第一版が提供している，ユーザに網を意識させない共通のアクセス法は，従来の「データ独立」（ユーザにデータベースの内部表現を意識させない）の概念を拡張したと考えられるが，重複データを認める場合にも同様に，ユーザに，ファイルの新規作成時以外は，データの重複を意識させないことが望まれる．これを実現するためにはユーザから見える１つのファイル（以下，論理ファイルという）と網上に分散している複数のファイル実体（以下，これらのファイルの一つ一つを物理ファイルという）との対応づけを，ユーザから見えないように行なわなければならない．

ゆえに第二版では，上記の操作を行

ない，ユーザにデータの重複を意識させないアクセス法を提供することを，その主たる目的としている．

以下，第一版と異なる点を中心に説明する．

## 6．2　構成

第一版と比べて構成要素には変更はない．しかし，ＦＭＭとＦＡＣＩに変更が加えられた．詳細は，６．４，６．５，６．６で述べる．

## 6．3　ディレクトリ

第一版と同様に，２階層テーブル方式を採用したが，ディレクトリ情報をおさめるテーブルに変更がある．（図８参照）以下の説明では変更点に下線を付した．

### (1)　ＵＦＮＴ

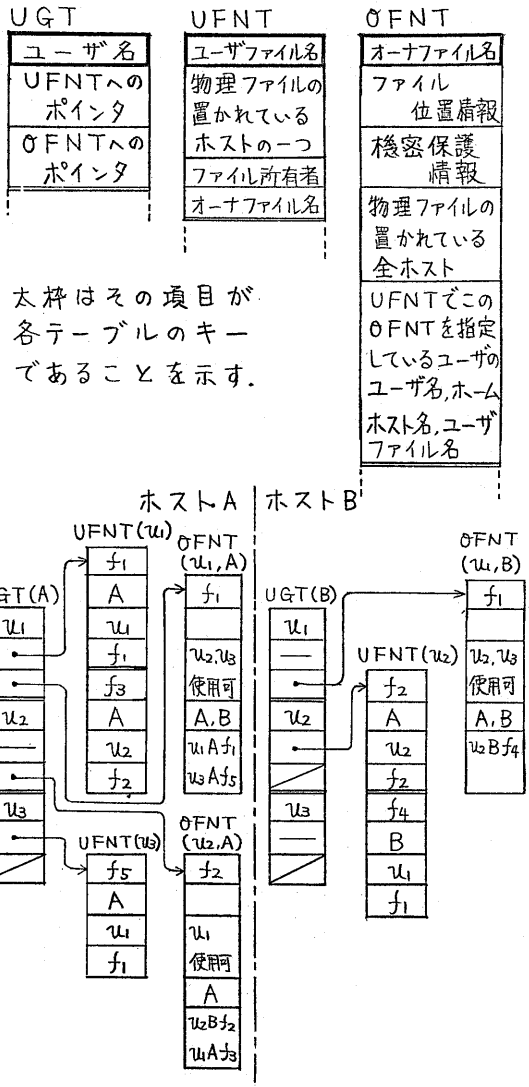
ユーザファイル名から，物理ファイルの置かれているホストの（任意の）１つ，ファイルの所有者およびオーナファイル名を得るための表である．各ユーザごとに網内にただ１つあり，使用する論理ファイルごとに１エントリある．

### (2)　ＯＦＮＴ

オーナファイル名から，その物理ファイルの格納されている位置情報，機密保護情報，重複データをもっている物理ファイル（コピー）の置かれている全ホストおよびＵＦＮＴでこのＯＦＮＴを指定しているユーザへのポインタを得るための表である．各ファイル所有者，各ホストごとに１個設けられ，所有している物理ファイルごとに１エントリある．

## 6．4　プロトコル

まず変更のあったプロトコルをあげる．変更点には下線が付してある．




- ユーザ $u_1$，$u_3$ のホームホストはＡ，ユーザ $u_2$ のホームホストはＢである．
- ユーザ $u_1$ は，Ａ，Ｂに $f_1$ を所有し，$u_2$，$u_3$ に使用を許可している．これを $u_2$ は $f_4$，$u_3$ は $f_5$ と呼んでいる．
- ユーザ $u_2$ は，Ａに $f_2$ を所有し，$u_1$ に使用を許可している．これを $u_1$ は $f_3$ と呼んでいる．

図８．ディレクトリの構成

### (1)　ＧＥＴＦＮ（ユーザプロセス⇄ユーザのホームホストのＦＭＭ）

ＵＦＮＴによって，ユーザファイル名から物理ファイルの置かれているホストの１つ，ファイルの所有者およびオーナファイル名を得る．

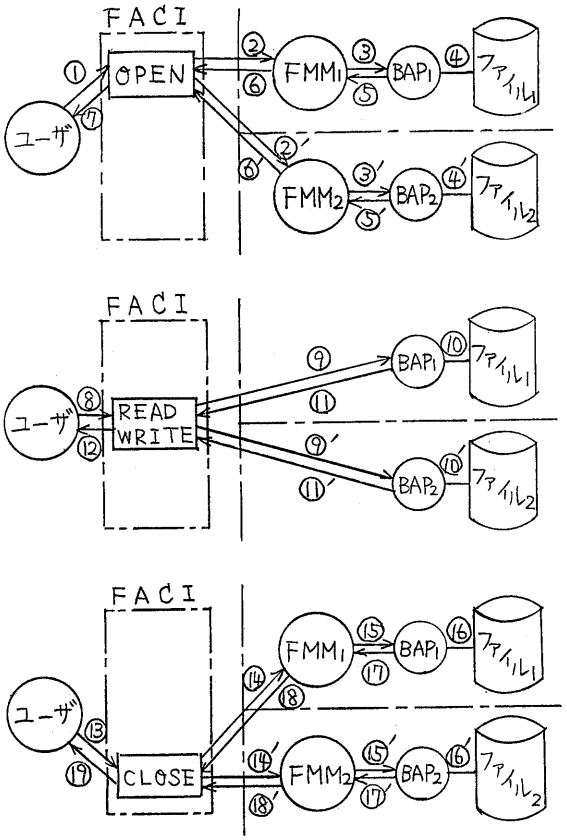



図9. ファイルアクセスの手順

(2) CATALOG, UNCATALOG (ユーザプロセス⇄ユーザのホームホストのFMM⇄UFNTの指すホストのFMM)
UFNTへの登録, UFNTからの削除と, それに伴なうOFNTの変更を行なう.

(3) PERMIT, INHIBIT (ユーザプロセス→物理ファイルの置かれているホストのFMM1→…→物理ファイルの置かれているホストのFMMk→ユーザプロセス)
すべての物理ファイルのOFNTを変更する.

次に第二版で新設したコマンドをあげる.

(1) ADDCOPY, DELCOPY (ユーザプロセス→物理ファイルの置かれているホストのFMM1→…→物理ファイルの置かれているホストのFMMk→ユーザプロセス)
重複データをもつ物理ファイルの作成, 消去を行なう. また同時に, OFNT, UFNTの変更も行なう.

(2) GETPS (ユーザプロセス⇄UFNTの指すホストのFMM)
OFNTによって, オーナファイル名から物理ファイルの置かれている全ホストを得る.
その他のコマンドには変更はない.

## 6.5 ファイルの確保および解放の手順

複数の物理ファイルからなる論理ファイルを確保するには, Readの場合はユーザから一番アクセスしやすい物理ファイルだけを, Writeも行なう場合は物理ファイルをすべて確保すればよい. その手順は次の通りである.

(1) ユーザが必要なファイルの一覧表を作る.
(2) このうち, ユーザファイル名によって指定されているファイルは, GETFNコマンドによって物理ファイルの置かれているホストの一つ, ファイルの所有者およびオーナファイル名を得る. さらに, GETPSコマンドによって物理ファイルの置かれている全ホストを知った後, Readの場合一番アクセスしやすい物理ファイルを, Writeも行なう場合はすべての物理ファイルを一覧表に加える.
(3) OBTAINコマンドによりファイルの一覧表を回覧させる.

ファイルを解放するには, (3)の一覧

表をＦＲＥＥコマンドにより回覧させればよい．

6．6　ファイルアクセスの手順

重複データが存在するときに，ユーザにそれを意識させるべきでないのは前述の通りである．第二版ではＦＣＢ（File Control Block）とＦＡＣＩに変更を加えてこれを達成し，また重複データ更新方式としては，更新権回覧方式を採用した．その手順を以下に述べる．（図9参照）

(1) ユーザ（実際はジョブ管理システムが代行）はＯＢＴＡＩＮコマンドによりファイルを確保した後，論理ファイルごとに，それを構成する物理ファイルに関する情報を含んだＦＣＢを作る．

(2) ユーザプロセスは（ＦＡＣＩのＯＰＥＮルーチンを通じて）物理ファイルの置かれているすべてのホストのＦＭＭにＯＰＥＮコマンドを送る．ＦＡＣＩはＦＣＢを参照しながらコマンドを送信するので，これらのコマンドの送信は，ユーザプロセスが1回ＯＰＥＮルーチンを実行するだけで行なわれる．（①②②′）

(3) 各ＦＭＭは，ＢＡＰにファイルのオープンを指示する．（③③′）

(4) ＢＡＰはファイルをオープンし，ＦＭＭに応答を返す．（④④′⑤⑤′）

(5) 各ＦＭＭはユーザプロセスに応答を返す．（⑥⑥′⑦）

(6) 以後はユーザプロセスが（FACIのＲＥＡＤ，ＷＲＩＴＥルーチンを通じて）各ＢＡＰと通信することによってファイルのRead,Writeが行なわれる．（⑧～⑫，⑨′～⑪′）

(7) クローズの手順は，オープンの手順とほぼ同様である．（⑬～⑲，⑭′～⑱′）

## 7．おわりに

この研究では，ユーザが網や重複データを意識することなくファイルにアクセスできることに重点をおいた分散ファイルシステムの構成法を追求している．現在実装されている第一版では，比較的小規模なプログラムによって，ユーザに網を意識させないアクセス法が提供されており，現在実装中の第二版では，さらにユーザにデータの重複を意識させないアクセス法が提供される予定である．

ユーザが網や重複データを意識しないてよいということは，網向きオペレーティングシステムの中で，ファイル管理という一つの階層を設けることでもあり，その上に構築すべき分散データベース管理システムの負担を軽くする意味がある．

したがって，新規に分散データベースを作成する場合のアプローチとして，プロセス間通信機構および分散ファイルシステムを作成するのは，有利な方法であると思われる．

〔参考文献〕

1. 堀口真志他：分散データベース用ファイル管理システム，第19回全国大会，1978，pp. 957～958.
2. 堀口真志，元岡達：分散データベース用ファイル管理システム，特定研究Ｂ－19班研究発表会資料.