ＯＺ：対象指向開放型分散システムアーキテクチャ
--- オブジェクト記憶用サーバ ---

6H-6

塚本享治（電総研）　中込昌吾（ＡＢＣ）
田中伸明（松下電器）　吉江信夫（住友電工）　近藤貴士（シャープ）

１．はじめに

ＯＺプロジェクトは、システムを変更することなく、システム機能をユーザが簡単に拡張・変更できる、真の意味の開放型分散システムを実現することを目指している[1]。データ、手続き、プロセス、メッセージなどすべてのものを継承機能を有するオブジェクトで表現し、オブジェクトの相互関係を保存して転送することが基本となっている。すでに単一ユーザ環境用のシステムは実現されているが[2]、そのままでは複数ユーザが拡張定義してオブジェクトを相互に交換することができない。これを可能にするには、ユーザ間で相互にオブジェクトを認識したり、共有することが出来なければならない。本稿では、これを可能にするオブジェクト管理システムで使用する記憶サーバについて述べる。

２．ＯＺのオブジェクト記憶サーバ

現在、ＯＺにはファイルサーバとディクショナリサーバの２つの記憶用サーバを実現している。

２．１　ファイルサーバ

ファイルサーバは、順編成アクセス（ｓａｍ）、索引順編成アクセス（ｉｓａｍ）の２種類のアクセス法を提供する。このうち前者は、ｕｎｉｘファイルシステムと互換性のあるモードであり、ソースファイルにアクセスする時などに使用する。また、後者は、キーを持った可変長レコードを記憶するファイルに対してキーを用いてランダムにアクセスするためのものでＯＺではオブジェクト名やオブジェクトｉｄキーとして、パッケージにアクセスする時などに使用する。
ファイルサーバが支援する機能を表１に示す。

表1　ファイルサーバがサポートする機能

共通	open	ファイルをオープンする
	close	ファイルをクローズする
	remove	ファイルを削除する
SAM	read	1レコード分入力する
	write	1レコード出力する
	skip	アクセスポジションを指定レコード数分進める
	bread	シーケンシャルに指定バイト分入力する
	bwrite	指定バイト数分出力する
	bseek	アクセスポジションを指定バイト分移動する
ISAM	read	指定されたキーを持つレコードを入力する
	write	指定したキーのレコードを追加出力する
	rewrite	指定したキーのレコードを置換出力する
	delete	指定したキーのレコードを削除する
	find	指定したキーのレコードの有無を調べる
	get	キー順に1レコード入力する
	put	ファイルに1レコード出力する

２．２　ディクショナリサーバ

ディクショナリサーバは、バイトコードインタプリタ（ｂｃｉ）、分散カーネルに対してデータベース的役割を果す。

現在、すべてのｕｎｉｘマシン上で共通に動作するデータベースツールがないため、このサーバを開発した。

ＯＺでは、実行時の資源管理にはｉｄを使用する。ディクショナリサーバは、資源とｉｄを結び付けるデータベースとして機能する。

データベースの検索においては、検索項目の論理積の結果を用いる。これは、ＯＺにおいてはあるドメインが持つオブジェクトをさがしたり、あるオブジェクトの上位／下位の関係にあるオブジェクトをさがすときなどは、論理積条件で項目を絞る場合が大半であるためである。

ディクショナリサーバが支援する機能を表２に示す。

表2　ディクショナリサーバがサポートする機能

adddib	新規レコードとして登録する
moddib	指定レコードの項目を変更する
qrydib	指定レコードを検索する
deldib	指定レコードを削除する

２．３　カタログとライブラリ

ファイルサーバ、ディクショナリサーバは、どちらも情報を記憶するための容器であり、これらのサーバをいかに使うかが問題となる。

ＯＺにおいては、ユーザをまたがるオブジェクトは全てｉｄで表され、このｉｄをカタログに登録することにより、他ユーザがこのｉｄをもつオブジェクトの参照が可能になる。また、オブジェクトの実行部分（タイプ）がもつオブジェクトの名称をｉｄ化しタイプに付けたｉｄをキーとしてタイプをライブラリに登録することにより、他ユーザがこのタイプをロードすることができるようになる。

２．４　これらの機能の関係

これらの機能の包括関係は、カタログ／ライブラリ＞

OZ: Object Oriented Open Distributed System Architecture -- Servers for Object Store
Michiharu TSUKAMOTO[1], Shogo NAKAGOME[2], Nobuaki TANAKA[3], Nobuo YOSHIE[4], Takasi KONDO[5]
1)Electrotechnical Laboratory　2)ABC Corporation,Ltd.　3)MATSUSHITA Electric Industrial Co.,Ltd.
4)SUMITOMO Electric Industries, Ltd.　5)SHARP Corporation

ディクショナリ＞ｉｓａｍ＞ｓａｍの順になっている。カタログ／ライブラリはディクショナリをサーバとして用いる。ディクショナリサーバでは直接ｉｓａｍを使い二次記憶にアクセスしている。

３．実装

次に、各サーバの実装の方法について報告する。

３．１　ファイルサーバ

ファイルサーバは、大きく２つの部分に分けられており、ｓａｍ機能部はｕｎｉｘファイルシステムと同様であるが、ｉｓａｍ機能部はこれがｕｎｉｘにないため新たに実現した。

ｉｓａｍ機能部は次の様に実現した。ｕｎｉｘでは、磁気ディスクのシリンダ、トラック、セクタ、レコード番号等の物理的アドレスを使用して、アクセスする事が難しくまた、相対アドレッシングも相対バイトアドレスのみであり、可変長レコードを扱うのが難しい。このため、ｖｓａｍのｃｉ(Control Interval)の概念をとりいれて、ｕｎｉｘファイルを固定長の箱（ブロック）の集合としてとらえ、その箱（ブロック）の中にインデックスレコード、データレコードを格納するという実現方法をとった。

ｉｓａｍのインデックスブロックは階層構造になっており、上位のインデックスブロックは、下位のインデックスブロックまたはデータブロック中のレコードの最終キーの値を保持している。

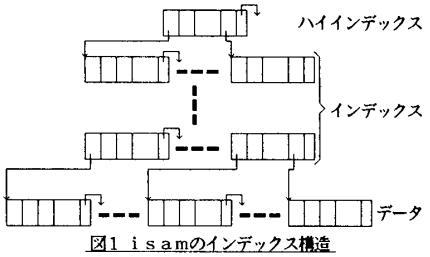

図1 ｉｓａｍのインデックス構造

検索する時は、上位のインデックスレコードを順次検索していき、検索キーよりもインデックスレコードの値が大きいか等しくなった時のインデックスレコードが持つ、下位ブロックへのポインタで指されているブロックに求めるキーのレコードがある。もしも、下位ブロックがインデックスブロックの場合には、ふたたびそのインデックスレコードを順次検索する。

下位ブロックがデータブロックの場合には、物理レコードごとにキーを比較していき、求めるレコードを決定する。

３．２　ディクショナリサーバ

ディクショナリサーバは、項目と属性値を項目別に登録する部分（図２のｉｓａｍ部）と、属性値をキー値として登録し検索する部分（図２のｉｓａｍ部以外）よりなる。

登録する属性値は、登録単位ごとに項目をまとめ索引キーを付けたｉｓａｍレコードを作成しｉｓａｍファイルに保存する。

ｉｓａｍレコードは、固定部、ディレクトリ部、データ部からなる可変長レコードである。固定部は、索引キー、データ部開始ロケーション、データ部サイズよりなり、ディレクトリ部、データ部には、属性値が定義されている項目のみを持つことにより、ファイルスペースの効率化をはかっている。このｉｓａｍレコード形式は図書情報の交換において広く用いられているＭＡＲＣ形式をベースにした。

一方、属性値をキーとする部分は、メモリ上に展開しハッシュ手法によって検索の効率を上げている。

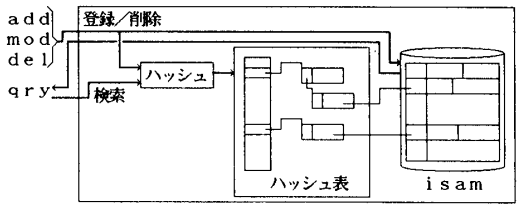

図2 ディクショナリの構造

新規レコードを登録するさいは、属性値をｉｓａｍレコードのデータ部に追込み、ディレクトリ部に各項目のｉｄと属性値の長さをセットし出力する。また、検索のためのキーは項目ごとに、項目ｉｄと属性値をハッシュし登録する。

検索における該当レコードの決定は次の手順で行う。各検索キーに対して、それをハッシュし、ハッシュ表より索引キー集合（ａ）を求める。全ての検索キーからもとめた（ａ）の論理積をとり、その結果の索引キー集合（ｂ）を作成する。（ｂ）に含まれる索引キーで、ｉｓａｍファイルからｉｓａｍレコード（ｃ）を読む。（ｃ）より選択指定項目を抽出し検索結果とする。

４．むすび

以上述べたファイルサーバとディクショナリサーバを用いて、カタログとライブラリを実現する作業を進めている。これについては別の機会に述べたい。

参考文献

[1]M.Tsukamoto et al.,"The Architecture of OZ: Object-Oriented Open Distributeed System", Proc. of 2nd ISIIS,p153-p166(1988)
[2]塚本他,"ＯＺ：対象指向開放型分散ｼｽﾃﾑｱｰｷﾃｸﾁｬ",情報処理学会第36回全国大会講演論文集,p633-p642(1988)