拡張可能 DBMS 構築技法に基づく高速 OSI ディレクトリ用 DBMS の設計と評価

西山 智† 小花貞夫†
堀内浩規† 鈴木健二†

これまで，OSI ディレクトリの適用分野としてメッセージ通信処理システム (MHS) など OSI 通信システムのネームサーバや電子電話帳が検討されてきた．近年，OSI ディレクトリを用いてユニバーサルパーソナル通信 (UPT) 等のサービスで必要なデータベース機能を提供することが考えられているが，そのためには高速な OSI ディレクトリのサービスを提供できるシステムが必要となる．筆者らは OSI ディレクトリのデータモデルと操作を直接提供する専用 DBMS を開発し OSI ディレクトリの高速化を図った．本論文ではこの DBMS の設計と評価について論じる．ここでは，開発と DBMS 内部の最適化の容易性を考慮して，ツールキット方式の拡張可能 DBMS の構築技法を用いて DBMS 内部を階層的にモジュール化している．従来の OSI ディレクトリの実装で高速化の妨げとなっていた 1) 名前解読処理，2)エントリ格納方式，3)フィルタ処理に対して，それぞれ 1) ディレクトリ情報木 (DIT) をエントリから分離し，木構造のアクセス手法を用いて格納すること，2)高速な ASN.1 符号化規則を格納形式とするエントリ単位の直接クラスタリング，3)識別名インデックスと属性インデックスによるフィルタ処理，を実現することによって高速化を図った．UNIX ワークステーション上に実装した DBMS の性能評価を通じて設計の妥当性を実証した．

Design of DBMS for High Performance OSI Directory Based on Extensible DBMS Technique and Its Evaluation

Satoshi Nishiyama,† Sadao Obana,† Hiroki Horiuchi† and Kenji Suzuki†

OSI Directory has been standardized to provide information on telecommunication users and OSI communication systems such as MHS. Recently, it is proposed that OSI Directory can also be used as a name server for other advanced telecommunication services such as Universal Personal Telecommunication (UPT). Since these are real-time switching services, it is required that the response time of the name server should be as small as possible. We consider that a key to high performance OSI Directory used for such name server is a dedicated DBMS which directly supports the OSI Directory data model and operations as its data model and operations. In this paper, the design of the DBMS is discussed. The software is divided into modules hierarchically, based on toolkit approach, one of the techniques for extensible DBMS, from the view points of easiness of design and further customization. To reduce the response time, the DBMS uses 1) distinguished name (DN) index, stored using 'tree structure' access method, for name resolution, 2) direct clustering based on Light Weight Encording Rules of Abstract Syntax Notation One (ASN. 1), as its clustring rule, and 3) attribute indexes and DN index for filtering entries without accessing the stored entries. The evaluation result of the implementation on a UNIX workstation shows that the DBMS can execute a READ operation in about 12 millisecond and it has enough performance for UPT name server.

1. はじめに

ネットワークの高度化に伴い，ネットワークの効率的な利用や運用を支援するための通信支援用データベースが重要になってきた．代表的な通信支援データベースとして通信相手の接続番号（アドレス）や通信能力等に関する情報を提供する開放型システム間相互接続 (OSI) ディレクトリ[1]が標準化されている．これまでに OSI ディレクトリはメッセージ通信処理システム (MHS) などの通信システム用のネームサーバや電子電話帳として適用されてきたが，近年 UPT（ユニバーサルパーソナル通信）等の高度な交換通信サービスを実現するためのネームサーバとしての適用も提案されている[2]．UPT は通信の利用者がどの網に移

† 国際電信電話株式会社研究所
KDD R&D Laboratories

動しても，その利用者に割り当てたパーソナル通信番号を指定することによってその利用者との通信を可能とするサービスであるが，サービス実現上必要なパーソナル通信番号から物理的なアドレスを得るための UPT ネームサーバとして OSI ディレクトリを適用できる．

OSI ディレクトリの従来の適用分野であった電子電話帳を通信利用者が利用する場合，秒単位の応答速度でも十分実用になった．また MHS で利用する場合についても，MHS は蓄積交換型の通信サービスであるためディレクトリに対する高速な応答を要求していない．これに対し，UPT 等は即時交換型のサービスであり接続遅延がサービス品質に大きく影響するため，数十ミリ秒単位の高速なディレクトリサービスを必要とする．これまでにいくつかの OSI ディレクトリの実装例が報告されている[3)~5)]が，これらは汎用的なディレクトリ機能の実証を主目的としたもので，高速性の実現を主目的としたものではない．特にリレーショナル型やオブジェクト指向型等の汎用あるいは独自のデータベース管理システム（DBMS）を用いた従来の実装[3),4)]では，OSI ディレクトリのデータモデル，操作と DBMS のデータモデル，操作とのマッピングが必要となり，高速化に限界があった．

筆者らはデータモデル間・操作間のマッピングをなくし OSI ディレクトリの高速な実現を可能とするために，OSI ディレクトリのデータモデルと操作をそのままデータモデルと操作とする専用 DBMS を開発した．本論文では，この DBMS の設計と評価について論じる．まず，OSI ディレクトリのデータベースとしての特徴について 2 章で述べる．3 章では高速化の観点からこれまでの実装例の問題点を明確化する．4 章では専用 DBMS の設計について，特にツールキット方式の拡張可能 DBMS の構築技法[6)]の適用と従来の問題点への対処を重点に論じる．最後に 5 章で，この設計に基づき実装した DBMS に対して 3 章の問題点に対する対処の観点から評価を行い，設計の妥当性を検証する．

2. データベースとしての OSI ディレクトリの特徴

ここでは文献 1) の標準で規定される OSI ディレクトリのデータベースとしての特徴を述べる．

2.1 データモデル

ディレクトリは，通信の対象となる「もの」（オブジェクト）に関する情報のデータベースで，これらの情報に関する各種のアクセス（読み出し，探索，変更など）機能を，利用者（例えば人や計算機プロセスなど）に提供する．ディレクトリ情報を提供する OSI の応用プロセスはディレクトリサービスエージェント（DSA）と呼ばれる．実際には，ディレクトリは複数の DSA から構成されることもあり，この場合これらの DSA が協調動作して利用者にディレクトリ情報を提供する．また，ディレクトリが提供する情報の集合はディレクトリ情報ベース（DIB）と呼ばれる．DIB は，各オブジェクトの名前管理（明確に識別できる名前を与えること）のために，また，複数の DSA にまたがって DIB を分散管理できるようにするために，ディレクトリ情報木（DIT）と呼ばれる木構造で論理的に表現される（**図 1**）．DIT 木構造における木の節はエントリを，また木の枝はエントリ間の従属関係を表す．一つの節から出る木の枝にはそれぞれ他の枝と異なる名前（相対識別名：RDN）が付与される．木構造の根（ルート）から特定のエントリに至る相対識別の並びは識別名（DN）と呼ばれ，そのエントリを一意に識別する．

オブジェクトに関する情報を格納するエントリをオブジェクトエントリと呼ぶ．このほかにオブジェクトに別名を与える別名エントリもあるが，ここでは簡略化のためにオブジェクトとエントリを同義語として扱う．エントリは属性の並びからなり，各属性は任意の数の属性値を持つことができる．属性値の型は抽象構文法記法 1（ASN. 1)[7)] により規定される．また，類似の性質を持つエントリを分類するために，国，組織，人間といったオブジェクトクラスが定義される．各エントリはいずれかのオブジェクトクラスに属しており，属するオブジェクトクラスにより格納可能な属性の種類が規定される．

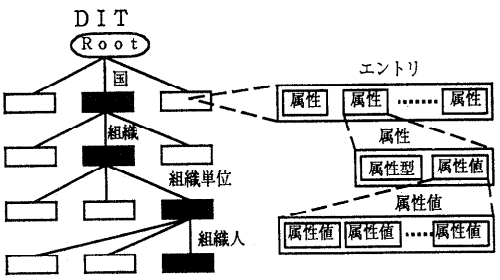


図 1 ディレクトリ情報木（DIT）の構造
Fig. 1 Structure of directory information tree (DIT).

2.2 データ操作

DIT に対して，既知の識別名からエントリの情報を読み出す Read 操作，特定の条件に合致するエントリを検索し情報を読み出す Search 操作，エントリを追加する AddEntry 操作等，**表1**に示す9種類の操作が定義されている．Search 操作では，検索範囲の部分木とフィルタと呼ばれる検索条件を指定できる．フィルタは各属性値に対する条件（値と比較条件）を任意に組み合わせたものである．

また上記のディレクトリ操作に対し DIB は以下のような性質を持つものとして規定される．

- 一時的なデータの不整合を許容している．
- すべての操作がエントリの単位で操作される．
- 更新操作は操作単位でのアトミック性を要求している．
- 検索系の操作に比較して更新操作の頻度は少ない．

3. 従来の OSI ディレクトリ実装における問題点の明確化

これまでにいくつか OSI ディレクトリの実装例が報告されている（**表2**）．これらは OSI ディレクトリの機能を汎用的に実現することを主目的として実装が行われたもので高速化の実現を主目的としたものではない．これらの実装例を高速化の観点から分析すると以下の問題点を挙げることができる．これらの問題点は，主にデータ管理機能として使用する汎用あるいは独自の DBMS やファイルシステムが提供するデータモデル，操作が DIB のモデル，操作と異なることに起因している．

（1） 名前解読処理方法

(1-1) DIT がエントリから分離されていない

文献 4), 5) は DIT とエントリの格納を分離していないため，DIT を1段名前解解読を行うたびにエントリ内容あるいは UNIX のディレクトリ内容を読む必要がある．

(1-2) DIT に適したデータ構造が提供されていない

文献 3) は，DIT をリレーショナル型 DBMS（RDB）の表形式にマッピングしており DIT を1段（あるいは複数段）名前解読を行うごとに RDB 操作を実行する．

（2） エントリの格納方法

(2-1) モデルが複数の属性値を扱えない

OSI ディレクトリはエントリに任意の数の属性値を持つことを許しているので，使用するデータ管理機能のデータモデルも複数の属性値を扱えるものが望ましい．RDB のモデル上の制約から，文献 3) では多数の属性値を含むエントリは複数の

表1 ディレクトリ操作
Table 1 Directory operations.

読み出し	Read	エントリの情報の読み出し
	Compare	エントリの情報と指定値の比較
	Abandon	すでに発行した操作の中止
検索	List	下位のエントリ一覧の表示
	Search	DIT 部分木に対する条件（フィルタ）検索
更新	AddEntry	エントリの追加
	RemoveEntry	エントリの削除
	ModifyEntry	エントリ内容の変更
	ModifyRDN	エントリの相対識別名の変更

表2 OSI ディレクトリの実装例
Table 2 Existing implementations of OSI Directory.

	文献 3)	文献 4)	文献 5)
データ管理機能	リレーショナル型 DBMS	独自オブジェクト指向 DBMS (UNIX ファイルシステムを格納に使用)	UNIX ファイルシステム
名前解読処理	DIT 専用の表にエントリ間の上下関係を格納し，複数の RDN を一度に解析	各エントリ内に下位エントリの RDN のリストを保持．各エントリ内容を読んで処理	DIT を UNIX の木構造管理にマッピング．UNIX のディレクトリ情報を読んで実現．
エントリ格納	クラスごとの表に格納．属性値が溢れた場合のみ属性ごとの表に格納．独自符号化を使用．	UNIX ファイルに直接クラスタリング．ASN. 1 基本符号化を使用．	UNIX ファイルに直接クラスタリング．独立符号化を使用．
フィルタ処理	検索範囲のエントリ内容を読んで処理	検索範囲のエントリ内容を読んで処理	検索範囲のエントリ内容を読んで処理

表にまたがって格納され，格納・検索操作が複雑となる．

(2-2) エントリ単位の直接クラスタリングが実現できない

OSI ディレクトリではエントリ単位の操作のみが定義されるため，エントリ単位で連続した領域に格納されている格納方法（直接クラスタリング）が望ましい．文献 3) は使用する RDB の制約上，直接クラスタリングを実現していない．

(2-3) エントリごとにファイルを割り当てている

文献 4), 5) はエントリ格納の管理を簡単にするためエントリごとに UNIX ファイルに格納している．このためエントリを読むたびに UNIX のファイルオープン・クローズ処理が必要となる．

(3) フィルタ処理方法

(3-1) フィルタ処理用のデータ構造がない

文献 3), 4), 5) のいずれもフィルタ処理を高速化するためのデータ構造を持っておらず，探索範囲のすべてのエントリの内容を読み込んでフィルタ条件に合致するエントリを探索する方法をとる．したがって，操作に要する時間は結果の件数ではなく探索範囲に比例する．

4. OSI ディレクトリ用 DBMS の設計

本章では OSI ディレクトリ用 DBMS の設計について述べる．ここでは特に外部仕様，拡張可能 DBMS 構築技法の採用，従来の実装例からの OSI ディレクトリ高速化に対する問題点への対処を重点として述べる．**表 3** に以下の設計の概要をまとめる．

4.1 外部仕様：データモデルと操作

OSI ディレクトリの高速化を図るためには DIB のデータモデル，操作と使用するデータ管理機能のデータモデル，操作とのマッピングをなくすことが必要である．ここでは 2.1 節，2.2 節で述べた DIB のデータモデル，操作を DBMS のデータモデル，データ操作系とし，データモデル間，操作間のマッピングを排除する[8),9)]．DBMS のデータ記述・定義系としての言語は特に設けず，データ記述・定義に相当するディレクトリスキーマと呼ばれる情報（例えばオブジェクトクラスや属性の型定義）を DBMS の中にデータの形で埋め込んで実現する．このデータ記述・定義系に関する仕様の妥当性については 6.2 節で考察する．

4.2 拡張可能 DBMS 構築技法の採用

一般にデータモデル，操作あるいは内部の実現方法を拡張できる DBMS を拡張可能 DBMS と呼ぶ．拡張可能 DBMS には，大きく拡張部分を解釈実行できる汎用的な DBMS を提供する方式[10)]と DBMS のソフトウェアを部品化しアプリケーションに応じた DBMS を生成するツールキット方式[6),11)]の 2 種類がある．ツールキット方式では，DBMS ソフトウェアを機能的，階層的にモジュール化することでモジュー

表 3 OSI ディレクトリ用 DBMS の機能概要
Table 3 Design summary of DBMS for OSI Directory.

外部仕様 (4.1 節)	データモデル	DIB モデル
	データ記述・定義系	なし．DBMS 中にデータとして埋め込み
	データ操作系	ディレクトリ操作
	外部インタフェース	ソケット通信
	通信符合化形式	ASN.1 LWER (注)
拡張可能性 (4.2 節)	実現方式	階層的モジュール化
	モジュール構成	9 レイヤ，11 モジュール
名前解読処理 (4.3.1 節)	実現方式	木構造アクセス手法を持つ識別名インデックスによる
エントリ格納 (4.3.2 節)	格納方式	エントリ単位の直接クラスタリング
	格納符号化形式	ASN.1 LWER (注)
	格納の効率化	フラグメンテーションにより固定長化
	アクセス手法	B-木
フィルタ処理 (4.3.3 節)	実現方式	識別名インデックスと属性インデックスを使用
	属性インデックスの作成	ディレクトリのクラス共通で属性型ごとに作成．値を正規化．B-木に格納
多重処理 (4.4.1 節)	実現方式	1 プロセス多重処理
	多重処理方式	オブジェクト単位で多重処理
排他制御 (4.4.2 節)	制御方式	施錠（更新ロック）
	制御単位	ページ単位
障害回復 (4.4.3 節)	対象障害	アプリケーション障害，システム障害
	実現方式	ページ単位，後イメージ
バッファリング (4.4.4 節)	実現方式	優先度付き LRU

(注) LWER：OSI の抽象構文記法 1 (ASN.1) のライトウェイト符号化規則

ル単位で DBMS の機能を拡張可能とするため，新しいデータ型や操作の追加のみならず，データモデル自身の変更や内部で使用するアクセス手法等のアルゴリズム変更も可能である．

OSI ディレクトリ用 DBMS の設計においては以下の理由によりツールキット方式の拡張可能 DBMS の構築技法を採用し，階層的にモジュール化されたソフトウェア構成とする．

- 設計の容易性

  一般に DBMS は複雑なソフトウェアであるが，ツールキット方式の拡張可能 DBMS の構築技法に従ってソフトウェアをモジュール化することで，個個のモジュールの機能が単純化され設計や実装が容易となる．

- アプリケーションに応じた高速化

  OSI ディレクトリでは，格納されるデータの内容（例えば属性の数や型，各属性の値の数）およびそれに対する操作の性質（例えば Read 操作が大半を占める等）は，OSI ディレクトリを使用するアプリケーションに大きく依存する．アクセス手法や論理ブロックなどの機能単位で DBMS のモジュールを変更することで，格納されるデータ内容や操作の性質に応じてデータの格納方式を最適化でき，アプリケーションに応じた DBMS の高速化が図れる．

- 他の類似データベースへの適用性

  OSI に基づくネットワーク管理のためのデータベースである OSI 管理情報ベース (MIB) は，OSI ディレクトリと類似した木構造，オブジェクト指向のデータモデルを持つ．OSI ディレクトリ用 DBMS のディレクトリ操作を実現しているモジュール等一部のモジュールを変更することで，MIB 専用の DBMS を作成することが可能となる．

ここでは DBMS の階層として，ファイル管理システムに相当する下位7レイヤ，インデックスを用いたフィルタ処理のためのインデックスレイヤと，ディレクトリ操作を実現するディレクトリサービスレイヤの

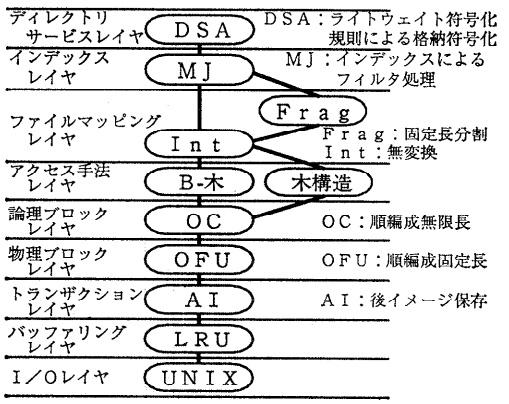


図 2 OSI ディレクトリ用 DBMS のモジュール構成
Fig. 2 Module structure of DBMS for OSI Directory.

表 4 各レイヤとモジュールの機能
Table 4 Function of each layer and module.

レ イ ヤ	機　　能	モジュール	モジュールの機能
ディレクトリサービス	ディレクトリ操作（Read 操作，Search 操作等）の実現	DSA	ASN. 1 LWER 符号化を使用
インデックス	フィルタ処理の実現	MJ	識別名インデックスと属性インデックスの検索結果によりフィルタ処理を実現
ファイルマッピング	ファイル構造の変換	Frag	固定長に分割
		Int	無　変　換
アクセス手法	論理ブロックを用いて高速なアクセス手段を持つファイルを提供	木 構 造	識別名インデックスのためを DIT に適したアクセス手法
		B-木	B-木を実現
論理ブロック	論理的なブロックを提供	OC	複数の物理ブロックを用いて無限長のソートされた論理ブロックを提供
物理ブロック	物理的なブロックへの格納方法を規定	OFU	固定長のブロックにソートして格納
トランザクション	トランザクションのアトミック性を提供	AI	コミットまで主記憶上で後イメージを保持
バッファリング	バッファリングにより I/O を減少	LRU	優先度付き LRU アルゴリズム
I/O	システム依存の I/O 操作を共通化	UNIX	UNIX のシステムコール

9 レイヤ構造とする（**図 2, 表 4**）．レイヤごとにインタフェースを統一しそのレイヤのモジュールはすべて同一のインタフェースを持たせることで，モジュールの追加，変更を容易とする．

4.3 従来の実装における問題点の対処

3 章で明確化した問題点に対する本 DBMS の対処について名前解読処理方法，エントリ格納方法，フィルタ処理方法の順に論じる．

4.3.1 名前解読処理方法

(1-1) DIT をエントリから分離し DIT 表現として**図 3** に示すように識別名に対してインデックスを生成する．名前解読処理では，この識別名インデックスを用いて，エントリ内容を読むことなしに識別名から対象となるエントリを特定する．

(1-2) DIT の表現に適したデータ構造として，DIT の木構造に対応した木構造アクセス手法を設ける．木構造アクセス手法を用いて識別名インデックスを格納することで名前解読処理の高速化を図る．この木構造アクセス手法を**図 4** に示す．各エンリに対応するノードは，下位レイヤ（論理ブロックレイヤ）が提供する論理的に長さが無限の論理ブロックにより実現する．論理ブロックにはキーとデータ内容からなる複数のレコードがキーの値に従って順編成で格納される．キーとして下位エントリの相対識別名を用いる．また，データ内容をさらにポインタとフラグに分け，ポインタには通常下位エントリのノードへの論理ポインタが格納される．その際，格納領域の効率化のために，その下位エントリが葉である場合は，下位エントリに対応するノードを設けずにポインタに下位エントリの実体を指すオブジェクト識別子を格納する（例えば図 4 の節 Z，A2）．フラグは，ポインタに格納される論理ポインタとオブジェクト識別子を区別する．

また，特殊なキーの値（ここでは 0）を用いてそのエントリ自身の情報を表す．この場合ポインタにはエントリの実体を指すオブジェクト識別子を格納する．

4.3.2 エントリ格納方法

(2-1) DIB のデータモデルをそのまま DBMS のデータモデルとすることで，データモデル上で複数の属性値を許容する．

(2-2) エントリ単位の直接クラスタリングを用いてエントリを格納する．直接クラスタリングを行う場合，格納の符号化形式が高速化を図るために重要となる．OSI ディレクトリでは属性の型として ASN.1 の任意の型が定義できることから，ASN.1 による符号化をクラスタリングの格納符号化形式とする．さらに符号化の高速化を図るために ASN.1 の基本符号化規則（BER）ではなく，ライトウェイト符号化規則（LWER）[12] を用いる．インデックスレイヤ以下のレイヤではデータの格納形式に依存しない構造とするために，ディレクトリサービスレイヤでクラスタリングを行う．

(2-3) DBMS の管理するファイル領域にエントリを格納することで無駄なファイルオープン・クローズ処理を除く．直接クラスタリングされたエントリの大きさは一定ではない．ファ

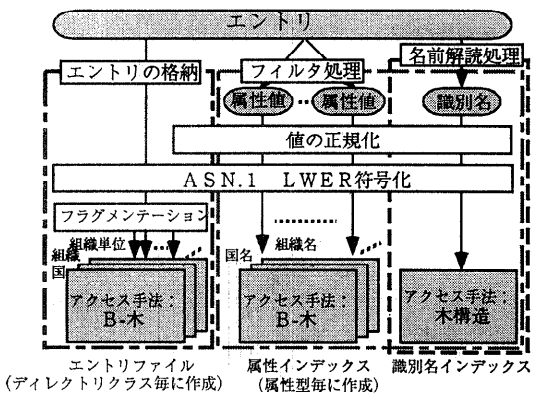


図 3 エントリとインデックスの格納
Fig. 3 Storing methods for entries and indexes.

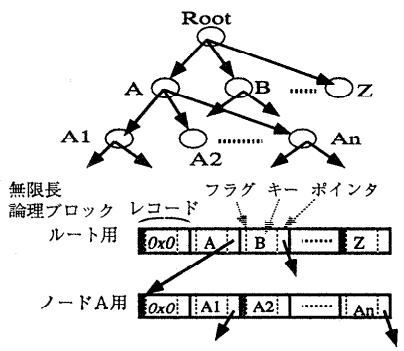


図 4 木構造のアクセス手法
Fig. 4 Access method of tree structure.

イルの格納効率を向上するためクラスタリングしたエントリをファイルマッピングレイヤで一定の大きさに分割（フラグメンテーション）する．分割したエントリをさらに各エントリに対して内部で付与するオブジェクト識別子 (OID) をキーとして B-木を用いて格納する．

以上のエントリの格納方法の概略を図3に示す．

4.3.3 フィルタ処理方式

(3-1) フィルタ条件に指定可能な属性に対してインデックスファイル（属性インデックス）を作成し，DIT を表現する識別名インデックスと属性インデックスによりエントリ内容を読むことなしにフィルタ検索処理を可能とする．

以下に属性インデックスの作成方法を示す．

- 属性インデックスの作成

  検索条件として指定可能な属性型ごとに属性インデックスを作成する．OSI ディレクトリでは複数のオブジェクトクラスが共通の属性型を含む場合がある．また Search 操作での検索範囲は DIT 部分木で指定され，その範囲内に異なるディレクトリクラスのエントリが含まれうる．したがって属性インデックスはクラスごとではなく，すべてのクラスに共通に作成する．インデックスレイヤが特定の符号化形式に依存しないように，ディレクトリサービスレイヤが属性インデックスを作成しインデックスレイヤに通知する．属性インデックスはB-木をアクセス手法として用いる．

- 属性値の格納形式

  格納形式として ASN.1 の LWER 符号化を用いる．OSI ディレクトリでは各属性に対して意味を考慮した比較規則（例えば大文字小文字を無視とか，電話番号のハイフンを無視する等）を定めている．検索時に単なるバイト列の比較が行えるように，属性インデックスに対しては符号化結果が一意となるようディレクトリサービスレイヤが正規化を行う．例えば，電話番号に対しては含まれるハイフンを除去する処理を行う．

識別名インデックスと属性インデックスを用いたフィルタ処理例を以下に示す．**図5**の DIB に対して {国：JP，組織：B} 以下の全部木を検索範囲としてフィルタ（通称＝a）を持つ Search 操作を例に用いる．識別名インデックスから該当エントリのオブジェクト識別子の集合 OID＝{3, 6, 7} が得られ，また通称の属性インデックスから OID＝{4, 6} が得られる．これらの集合の積をとることで検索結果のエントリのオブジェクト識別子の集合（この場合 OID＝{6}）が得られる．

4.4 多重処理，排他制御，障害回復，バッファリング

以下では，DBMS のその他の機能のうち OSI ディレクトリ用 DBMS として一般の DBMS と異なる実現方法を採用した多重処理，排他制御，障害回復，バッファリングの方法について述べる．

4.4.1 多重処理方法

一般に多重処理の実現方法として利用者ごとに生成される複数のプロセスで DBMS を構成し並行処理する方法と，1プロセスで多重処理する方法の2種類が考えられる．前者では利用者ごとにプロセスが生成され多数の利用者がアクセスするデータベースでは資源面で不利であるため，後者の1プロセス内で多重処理する方法を用いることとする．具体的にはディレクトリサービスレイヤがエントリ単位の処理を行うたびに複数のディレクトリ利用者の要求を切り替えて多重処理を行う．

4.4.2 排他制御方法

2.2節で述べたようにOSIディレクトリは一時的なデータの一貫性のない状態を許容している．このため高速化の観点から排他制御を簡略化し，更新ロックのみによる排他制御を行う．ロックの単位はページ単位としアクセス手法レイヤで実施する．4.3.3 節で述べたように各々の属性インデックスをすべてのオブジェクトクラスに共通に作成する．B-木に対して更新時にファイル全体にロックを行う通常のロック手法を用いた場合，その属性型を持つエントリすべてに対する更新操作がロック待ちを引き起こす．ここではロック範囲を最小限とすることでこの問題に対処するため，B-木については更新で影響が及ぶ部分木のみをロッ

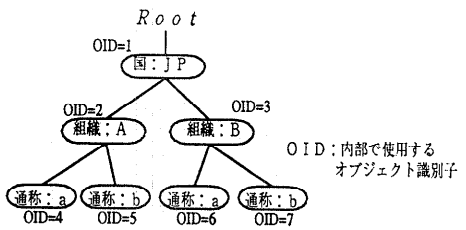


図 5 DIT の例
Fig. 5 Example of DIT.

クする手法[13]を用いることとした.

ロック待ちは操作単位とし，ディレクトリサービスレイヤが多重処理の一環として行う．デッドロックが発生した場合は，後発の操作を中止してデッドロック解除を行う.

4.4.3 障害回復処理方法

障害回復の対象をアプリケーション（ディレクトリ利用者エージェント：DUA）の障害ならびにシステム（DBMS）の障害とし，トランザクションレイヤでページ単位の障害回復処理を実現する．OSI ディレクトリではトランザクション中で一度に更新されるエントリは少ないため，障害回復方法として主記憶上に変更後のページのイメージをコミットまで保持する方法を用いる.

4.4.4 バッファリング方法

バッファリングはバッファリングレイヤでページ単位で実施する．バッファリングアルゴリズムとして，優先度「高」のページが一定割合以内の場合，優先度の「低」のページから掃き出しの犠牲者が選択されるという高低2段階の優先度付き LRU を使用する．アクセス手法レイヤがこの優先度の指定を行うこととし，アクセス傾向がランダムに近い B-木の葉ノードに対応するページは優先度が「低」で，他のページはすべて優先度が「高」としてバッファリングさせる.

5. 性能評価

4章で述べた設計に基づき，C 言語と ASN. 1 の記述から C 言語の LWER の符号化／復号関数を生成する LWER コンパイラ[14]を用いて Sun ワークステーション上に実装を行い性能評価を行った.

ここでは，3章で挙げた1)名前解読処理，2)エントリ格納，3)フィルタ処理の3項目に対する処理性能の観点から，検索系の操作（Read 操作，Search 操作）の応答速度を測定した．**表5**に評価の測定条件を示す.

5.1 名前解読処理

名前解読処理の性能を見るために2分木，RDN の深さ14段，エントリ件数 $2^{14}-1$（=16,383件）件の DIT に対して指定する識別名の RDN 数を変化させて Read 操作の応答時間を測定した結果を**図6**に示す．ここではバッファのヒット率の変化を排除するために同じ操作を繰り返し実行した．グラフの傾きから RDN1 段当たりに必要な名前解読処理時間が約 0.6 ミリ秒であることがわか

表5 測定条件
Table 5 Condition for measurement.

測定環境	Sun SPARC 670 MP	
測定内容	利用者プロセスでの操作を DBMS に送信してから結果を受信するまでの応答時間	
格納内容	国	属性：クラス，国名の2件9バイト
	組織	属性：クラス，組織名の2件23バイト
	組織単位	属性：組織単位名等5件約60バイト
	組織人	属性：通称等6件約70バイト
操作内容	Read	指定エントリの全属性の読み出し
	Search	指定エントリ以下の全部分木を範囲とし一致条件のフィルタ1件．結果は1件.

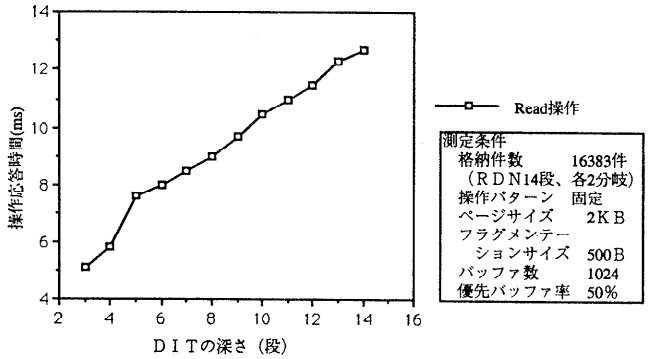


図6 DIT の深さと操作応答時間
Fig. 6 Response time: Number of RDNs.

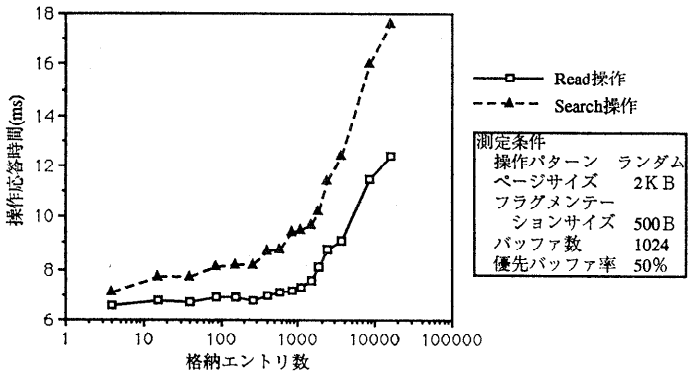


図7 エントリ格納件数と操作応答時間
Fig. 7 Response time: Number of entries.

る.

5.2 エントリ格納

エントリ格納件数の変化に伴うエントリ検索性能を見るために，RDN の深さ4段（国，組織，組織単位，組織人），2段目以下が各段で均等に分岐する DIT に対して，各段での分岐数を変化させることでエントリ格納件数を変化させ，ランダムに選んだ組織人クラスの葉エントリに対する Read 操作，葉の1段上の組織単位エントリに対して表5で示す Search 操作を実行させた．これらの操作の応答時間を**図7**に示す．

5.3 フィルタ処理

フィルタ処理性能を見るために検索範囲を変化させて Search 操作の応答速度を測定した．アクセス手法の評価で用いた2分木，RDN の深さ 14 段，エントリ件数 $2^{14}-1$ (=16,383 件)の DIT を用いて，1段の部分木の下にフラットに分岐数が増えていくために検索範囲が広がる場合と，2分木で部分木の深さが増えていくことにより検索範囲が広がる場合を測定した結果を**図8**に示す．ここでも，バッファのヒット率の変化を排除するために同じ操作を繰り返し実行した．グラフの傾きから，検索範囲が1件広がることによる遅延は，1段にフラットに多数のエントリが存在する場合は 0.15 ミリ秒程度，また2分木の場合は 0.8 ミリ秒程度であることが分かる．

6. 考　　察

6.1 性能評価結果について

（1） 総合的な処理性能について

評価の結果，約16,000件格納した状態で Read 操作が約 12 ミリ秒，Search 操作が 17 ミリ秒という応答時間が得られた．UPT を含む高度な通信サービスであるインテリジェントネットワーク（IN）の規格の1つである AIN[15] では UPT での番号変換など交換サービス制御を行う SCP（サービス制御ポイント）の応答時間として平均150ミリ秒，最悪 180 ミリ秒と定めている．今回実現した専用 DBMS の応答時間は SCP の応答時間の1割程度であるため，SCP がネームサーバとして用いても，SCP の応答時間の遅延にあまり影響しないと考えられ，本 DBMS の UPT 等のネームサーバとししての適用性が実証できた．

（2） 名前解読処理性能

評価結果から RDN 1 段当たりの名前解読処理時間は 0.6 ミリ秒程度である．現実的には DIT の深さは多くとも 10 段程度と考えられ，最大でも名前解読処理時間は6ミリ秒程度に抑えられる．高速な名前解読処理の実現ができたと考えられる．

（3） エントリ格納性能

実際の操作応答時間から名前解読処理の時間を除くと，エントリの読み出し速度については読み出しから符号化まで1万件以上格納した状態で10ミリ秒以下と十分高速化ができた．格納件数を変化させた場合の応答速度については，格納件数を M とすると B-木の特性である $O(\log M)$ よりも悪い傾向を示している．評価内容がランダムなエントリに対する検索という LRU アルゴリズムに最も悪い条件であり，エントリ格納件数の増加にともなってバッファのヒット率が低下しているためと考えられる．バッファ数を増加してヒット率を100%としたところ，応答時間が $O(\log M)$ 近くまで改善された．したがって，バッファリングアルゴリズムを改良することでさらに処理性能の向上が期待できる．

（4） フィルタ処理性能

識別名インデックスの格納に用いた木構造アクセス手法の構造から，検索範囲1件当たりの遅

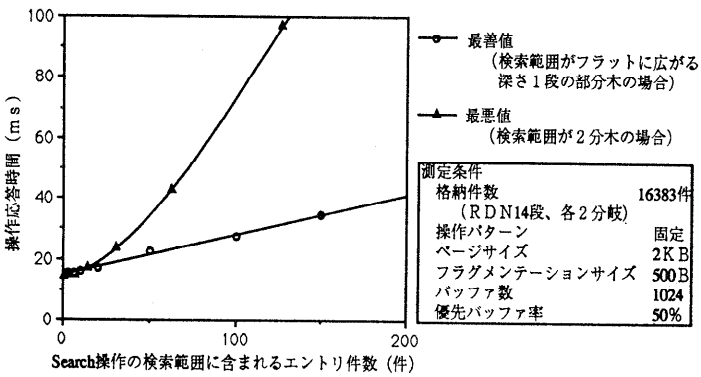


図8 フィルタ検索範囲と操作応答時間
Fig. 8 Response time: Search scope.

延は1段にフラットに多数のエントリが存在する場合が最良の条件，2分木の場合が最悪の条件であると考えられる．評価結果から，フィルタ処理の遅延は検索範囲に比例して増加するものの，検索範囲1件当たりの応答時間の増加は最良の条件で0.15ミリ秒，最悪値で0.8ミリ秒程度と，検索範囲のすべてのエントリの内容を実際に読む場合と比較して十分小さく，高速なフィルタ処理が実現できた．

6.2 実現した OSI ディレクトリ用 DBMS の汎用性について

OSI ディレクトリでは，オブジェクトクラスや属性の定義を総称してディレクトリスキーマと呼ぶ．OSI デイレクトリ用 DBMS ではディレクトリスキーマが追加，変更された場合，ディレクトリサービス中に存在するディレクトリスキーマに関するデータを変更し，さらに ASN.1 で記述されたオブジェクトクラスや属性型の構造定義を ASN.1 の LWER コンパイラにかけて再コンパイルすることで追加，変更に対処できる．したがって，従来の OSI ディレクトリの実装例が実現していた静的なディレクトリスキーマへの対応機能を本 DBMS も具備している．一般に OSI ディレクトリに必要なオブジェクトクラスや属性型の種類は使用されるアプリケーションに依存するので，ディレクトリスキーマは動的に変更されることは少なく，OSIディレクトリ用 DBMS は静的なスキーマ変更機能を具備することで十分であると考えられる．

7. むすび

本論文では，高速な OSI ディレクトリを実現するために DIB のデータモデル，操作をデータモデル，操作とする専用 DBMS の設計と評価について論じた．

モジュール化によって設計，実装が容易になることとアプリケーションに応じた DBMS 内部の最適化が容易になることを考慮し，ツールキット方式による拡張可能 DBMS の構築技法を用いて DBMS の内部構造を9レイヤに階層的にモジュール化した．従来の OSI ディレクトリの実装例での高速化の問題点に対して以下の対処を行った．まず名前解読処理方法については，DIT をエントリから分離して格納した（識別名インデックス）．識別名インデックスを格納するために DIT の表現に適した木構造アクセス手法を実現した．エントリ格納方法については，DBMS の管理するファイル領域にエントリ単位に直接クラスタリングを行った．フィルタ処理方法については，フィルタに指定可能な属性型ごとに属性インデックスを設けて，エントリ内容を直接読むことなしに識別名インデックスと属性インデックスからフィルタ処理を実現した．

さらに DBMS としての多重処理方式，排他制御方式，障害回復方式，バッファリング処理などにも OSI ディレクトリの性質を考慮した実現を行った．

評価の結果，1万件程度のエントリが格納された場合で Read 操作が 12 ミリ秒程度で実行でき，UPT などの高速性を必要とするアプリケーションのネームサーバとしても適用できることを示した．

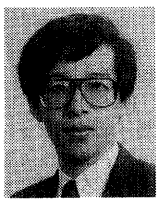
謝辞 日頃御指導頂く国際電信電話(株)研究所 浦野義頼 所長，眞家健次 次長ならびに御討論頂いた同研究所通信網支援ソフトウェアグループ浅見徹リーダに感謝します．また本論文の作成に当たり有益な御助言，御指導を賜った文部省学術情報センター 小野欽司教授（前国際電信電話(株)研究所 所長）に感謝します．

参考文献

1) ISO/IEC 9594 1-8: Information Processing Systems—Open Systems Interconnection—The Directory (1988).
2) 小花，堀内，加藤，鈴木：ユニバーサルパーソナル通信（UPT）への OSI ディレクトリの適用と評価，電子情報通信学会論文誌，Vol. J 74-B-1, pp. 959-970 (1991).
3) 小花，西山，鈴木：リレーショナルアプローチによる OSI ディレクトリの DIB（ディレクトリ情報ベース）の実装と評価，情報処理学会論文誌，Vol. 32, No. 11, pp. 1488-1497 (1991).
4) 中川路，勝山，宮内，玉田，水野：OSI ディレクトリシステムにおける DIB（ディレクトリ情報ベース）のオブジェクト指向による実現，情報処理学会論文誌，Vol. 32, No. 3, pp. 304-313 (1991).
5) Robbins, C. J., Kille, S. E. and Turland, A.: The ISO Developement Environment: User's Manual Vol. 5: QUIPU (1989).
6) Batory, D. S. et al.: GENESIS: A Reconfigurable Database Management Systems, University of Texas at Austin, Tech. Report TR-B 6-7 (1986).
7) ISO 8824, 8825: Information Processing Systems—Open Systems Interconnection—Specification of Abstract Syntax Notation One (ASN.1)/Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) (1987).

8) 西山，小花，堀内，杉山，鈴木：拡張可能 DB アプローチに基づく通信支援用 DBMS の構築に関する提案，第 44 回情報処理学会全国大会論文集，3L-4 (1992).
9) 西山，小花，堀内，鈴木：高速 OSI ディレクトリ用 DBMS の設計，第 45 回情報処理学会全国大会論文集，3R-9 (1992).
10) Stonebraker, M. and Rowe, L.: The Design of POSTGRES, *Proceedings of 1986 ACM SIGMOD Conference on Mangement of Data* (1986)
11) Carey, M. J. et al.: The Architecture of the EXODUS Extensible DBMS, *Proceedings of Object-Oriented Database Workshop*, pp. 52-65 (1986).
12) ISO/IEC JTC 1/SC 21 N 6131: Working Draft for Light Weight Encoding Rules (1991).
13) Bayer, R. and Schkolnik, M.: Concurrency of Operations on B-Trees, *Acta Informatica* (1977).
14) 堀内，小花，鈴木：OSI 応用層プロトコル用 ASN. 1 ライトウェイト符号化規則のための符号化／復号処理系の実装と評価，情報処理学会マルチメディア通信と分散処理研究会資料，DPS-55-2, pp. 9-16 (1992).
15) Bellcore: Advanced Intelligent Network (AIN) Release 1: Service Logic Program Framework Generic Requirments, Issue 1 (1991).

(平成 4 年 11 月 9 日受付)
(平成 5 年 4 月 8 日採録)

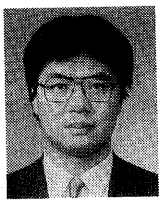


西山　智（正会員）

昭和 36 年生．昭和 59 年東京大学工学部電気工学科卒業．同年国際電信電話(株)入社．平成 3 年米国テキサス大学オースチン校計算機科学科修士課程修了．現在，同社研究所通信網支援グループ担当主査．この間，データベース，ネットワークアーキテクチャ，国際ビデオテックス通信，分散処理技術およびプロダクションシステムの研究に従事．

小花　貞夫（正会員）

昭和 28 年生．昭和 51 年慶応義塾大学工学部電気工学科卒業．昭和 53 年同大学院修士課程修了．同年国際電信電話(株)入社．現在，同社研究所交換グループ主任研究員．工学博士．この間，パケット交換方式，ネットワークアーキテクチャ，OSI プロトコルの実装，データベース，国際ビデオテックス通信，分散処理技術，インテリジェントネットワークの研究に従事．電子情報通信学会会員．



堀内　浩規（正会員）

昭和 35 年生．昭和 58 年名古屋大学工学部電気工学科卒業．昭和 60 年同大学院情報工学専攻修士課程修了．同年国際電信電話(株)入社．現在，同社研究所通信網支援グループ主査．この間，ネットワークアーキテクチャ，OSI プロトコルの実装方式，通信プロトコルの形式記述技法，ネットワーク管理の研究に従事．平成 4 年度電子情報通信学会学術奨励賞受賞．電子情報通信学会会員．

鈴木　健二（正会員）

昭和 20 年生．昭和 44 年早稲田大学理工学部電気通信学科卒業．昭和 44 年から 45 年までオランダのフィリップス国際工科大学に招待留学．昭和 51 年早稲田大学大学院博士課程修了．工学博士．同年国際電信電話(株)入社．現在，同社研究所 OSI 通信グループリーダ．この間，磁気記録，パケット交換方式，ネットワークアーキテクチャ，分散処理の研究に従事．昭和 62 年度前島賞，平成 4 年度電子情報通信学会業績賞受賞．平成 5 年度より電気通信大学大学院情報システム学研究科客員教授．電子情報通信学会，IEEE 各会員．