データベースシステム 101－3
（1995．1．26）

ASN.1データベースシステムにおけるデータ格納方式

齋藤 淳† 原 隆浩* 春本 要* 塚本昌彦** 西尾章治郎*

†奈良先端科学技術大学院大学情報科学研究科 *大阪大学工学部情報システム工学科
**シャープ（株）技術本部情報技術研究所

ネットワーク，マルチメディア，ゲノムデータベースなど多岐のアプリケーションにおいて，データ型定義言語の国際標準である ASN.1(Abstract Syntax Notation One) が利用されるようになってきた．それにともなって，ASN.1 によって記述されるデータを蓄積，再利用したいという要求が高まってきている．本稿では，ASN.1 によって定義されるデータ構造をもつデータを格納することが可能な ASN.1 データベースシステムの設計および実装について論ずる．特に，これまでに商用オブジェクト指向データベースシステムを用いて筆者らが実装してきたシステムにおける検索効率，データの符号化に関する問題点を解決するために，ASN.1 特有の柔軟かつ複雑なデータ構造に適し，効果的な検索を可能にするクラスタリングおよびインデックス手法を提案する．

Design of the Data Storage for an ASN.1 Database System

Jun SAITOH† Takahiro HARA* Kaname HARUMOTO*
Masahiko TSUKAMOTO** Shojiro NISHIO*

†Graduate School of Information Science, Nara Institute of Science and Technology
*Department of Information Systems Engineering, Faculty of Engineering, Osaka Univ.
**Information Technology Research Laboratories, SHARP Corp.

ASN.1 (Abstract Syntax Notation One) is an ISO standard specifying a type definition language (ASN.1 notation) and its encoding rule. Recently, ASN.1 has been used in a wide variety of application areas such as computer networks, multimedia, and genome databases. Accordingly, it is highly required to store and reuse the data specified by ASN.1 in those application areas, and it becomes very important to build the database system for ASN.1 data. In this paper, we discuss the design and implementation issues of an ASN.1 database system. Analyzing several deficiencies of data retrieval and data coding functions in our previously developed system using a commercially available Object-Oriented Database Management System, we propose efficient data clustering and indexing schema which are appropriate for the flexible and complex structure of ASN.1 data.

1 はじめに

抽象構文記法 1 (Abstract Syntax Notation One: ASN.1) は，国際標準化機構 (ISO) によって定められたデータ構造記述言語であり，言語仕様および交換するデータを符号化するための規則から構成される．ASN.1 は，アーキテクチャの異なるマシン間において曖昧さのないデータ交換を可能にすることを目的とし，さまざまな種類のアプリケーション間で交換されるデータの構造を表現できるように強力なデータ構造記述能力を有している．このような特性から，ASN.1 の利用分野は年々拡大し，さまざまなアプリケーションが開発されるようになってきた．それに伴ってアプリケーション間で交換されるデータを蓄積，再利用したいという要求が出てきた．

このような要求にともない，これまでにもさまざまな視点から ASN.1 で記述されるデータ (以下，ASN.1 データと呼ぶ) に対する研究が進められてきた．そのなかには，ASN.1 データの蓄積を目的とした研究もいくつか提案されている．例えば，ODA 文書を格納するデータベースの構築 [8, 9] や管理情報ベース (Management Information Base: MIB) 用データベースの構築 [7]，OSI ディレクトリ用データベースの構築 [12] がこれにあたる．しかし，これらのうち文献 [11] や文献 [12] では，ODA，ディレクトリといった特定のアプリケーション固有のデータベースを構成することでパフォーマンスの向上を図っているため，異なるアプリケーションにおいて用いられている ASN.1 データを一つのデータベースに格納することが不可能であった．

そこで筆者らは，文献 [3, 10] において，アプリケーション開発効率の向上を目指し，ASN.1 でデータ型を記述する言語 ASN.1/PL をデータベース言語とするデータベースシステム ASN.1/DB を提案した．このシステムでは，そのデータ格納部として商用オブジェクト指向データベース管理システム (Object-Oriented Database Management System: OODBMS) を利用していたが，商用 DBMS を利用した ASN.1 データに対するデータベースには，一般に次のような問題点があると考えられる．

1. **検索効率における問題**
   データ検索における高速化に必要なインデックス，クラスタリング機能が，もととなる商用 DBMS で提供されているものをそのまま利用することになるため，ASN.1 データの構造を考慮した適切な手法をとることが不可能であった．さらに，ASN.1/DB としては不必要な商用 DBMS によって提供される機能のオーバーヘッドの影響も大きいと考えられる．
2. **符号化/復号化の問題**
   ASN.1 データは符号列として送受信されるが，データの格納を復号化した文字列によって行なっているため，データの送信においては符号化を必要とする．またデータの格納においては，検索対象とならない可能性のあるデータに対しても復号化する必要があった．

これらの問題は検索効率に直接影響を与えるものであるが，商用 DBMS を利用したシステムではすべてを解決することができない．

そこで本研究では，ASN.1 データを，その利用されるアプリケーションの種類に関わらず，蓄積，再利用ができるデータベースシステムの設計を目的として，ASN.1 で記述されたデータの格納方式について，より一般的な立場から議論する．

以下，2 章では ASN.1/DB に格納するデータ形式について，3 章でクラスタリング，4 章でインデックスについて議論する．5 章では現在筆者らが行なっている実装について触れる．最後に 6 章では，今回の設計・実装に対する考察を行ない，まとめとする．

2 ASN.1/DB におけるデータ形式

本稿において，ASN.1/DB とは，ASN.1 データ (特にネットワークアプリケーションの交換するデータ) を蓄積するための DBMS のことをいう．ASN.1 による記述を型定義言語とすることにより，ユーザは ASN.1 データを直接取り扱うことを可能にする．ASN.1/DB におけるデータ管理は，各 ASN.1 データを一意に識別するために付与する識別子 (OID) によって行ない，検索はこの OID を通して行なう．また，データの格納は物理

記憶に対して直接書き込み，データへのアクセスはページと呼ぶ特定の論理単位で行なう．

一般に，ASN.1 データは次の二つの表現形式で表される．

1. 計算機での内部表現形式
   ASN.1 データに対するさまざまな処理を行なうために，転送構文での表現形式を復号化することによって得られる．
2. 転送構文で表されたデータ
   転送構文での表現形式ネットワークを介して転送される表現形式であり，計算機のアーキテクチャに依存しない符号化規則，例えば BER (Basic Encoding Rules) を用いて計算機での内部表現形式を符号化することによって得られる．

データの格納方式を考える上でデータを転送構文として格納するか，データベースの内部表現で格納するために復号化するのかという問題が生じる．

転送構文で格納する場合とデータベースの内部表現で格納する場合とでは，それぞれ，次のような利点，欠点があげられる．

- 内部表現で格納する場合
  データ検索に対してはデータを提供しやすいが，データ格納時およびデータ送信時にはそれぞれ復号化や符号化処理が必要となる．
- 転送構文で格納する場合
  転送されてくる符号列を格納することで，送信時においてそのままデータを転送することができる．しかし，データ検索時には復号化処理を必要とする．

どちらの方式が効率良くデータを提供できるかという問題は，データ検索や通信の頻度など，ASN.1/DB を利用するアプリケーションに依存すると考えられる．ASN.1/DB は，さまざまなネットワークアプリケーションにおける ASN.1 データに対する効率的な格納を目的としているため，以下では転送構文で格納することを前提とし，この場合にどのようなクラスタリングおよびインデクシングが適しているかを考察する．

3 ASN.1/DB におけるクラスタリング

一般に，ASN.1 で定義されるデータ構造をもつデータは内部にネスト構造を含んだ複雑な構造となっている．このような複雑な構造をもつデータを，全体を一つの格納単位として二次記憶に書き込むと，このデータに含まれるある型に対する検索において，全体を主記憶上に呼び出すことになり，さらにその中から必要なデータを取り出さなければならない．したがって，検索効率を向上させるためには，論理的に一つのデータを分解して二次記憶に書き込む方法が有効であると考えられる [6]．

本章では，ASN.1 で記述されるデータに対して，どのように分解，格納すれば検索効率を向上できるかという問題を ASN.1 データに対する符号化の問題と合わせて考察する．

3.1 符号化規則に基づくクラスタリング

ASN.1 データに対する符号化では単純型 (VisibleString 型や INTEGER 型など) に対する符号化規則とは別に構造型 (SEQUENCE 型や SET 型など) に対する符号化規則が定められている．これは構造型を一つの符号化単位と考える規則であり，構造型に含まれるいくつかの単純型の符号列をまとめて一つの符号列とする方法である．例えば，図 1のモジュール定義によるデータをデータベースに格納する場合，Person 型に含まれる要素は，一つの符号列として符号化される (図 2)．

以下では，単純型に対する符号化と構造型に対する符号化のどちらかを選択した場合の格納方法について説明し，それぞれの方法がもつ利点および問題点を考える．

3.1.1 単純型符号化によるクラスタリング

各単純型に対して符号化を行ないデータを格納する場合，データは各クラス階層の最下層に現れる単純型に応じてページに格納する (図 3)．この場合，実データをもたない型には実データを指す OID をデータとして保持させる．このようにすることで，データの送信時には OID の指す符号列を集めることによって，構造型に対する符号列が得られる．検索においては各構造型に含まれる一

```
PersonalRecords DEFINITIONS ::=
BEGIN
Person ::= SEQUENCE {
        name          VisibleString,
        age           INTEGER,
        belong        BelongTo }
BelongTo ::= CHOICE {
        univ    [0]   University,
        comp    [1]   Company }
University ::= SEQUENCE {
        name          VisibleString,
        faculty       VisibleString,
        department    VisibleString }
Company ::= SEQUENCE {
        name          VisibleString,
        department    VisibleString }
END
```

図 1: モジュール定義の例

つの要素に対して，その要素だけを復号化することができる点において有効であると考えられるが，次のような問題点が挙げられる．

- 深いネスト構造をもつデータの場合，最下層にデータが格納されているため，航行操作に時間がかかる．
- 構造型全体を参照する場合にデータがいくつかのページに分散しているため複数のページへのアクセスが必要となる．

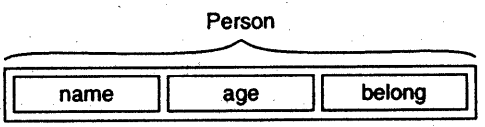

図 2: Person 型に対する符号化

3.1.2 構造型符号化によるクラスタリング

構造型に基づく符号化による格納では，構造型に含まれる要素をすべて同じページに格納する (図 4)．この場合の符号化は構造型に対する符号化を採用するため，一つのネスト構造が一つの符号列にされて格納される．この方法では，構造型で定義される一つのネスト構造を一度にアクセスすることができ，また，送信時はこの符号列をそのまま送ればよい．しかし，次のような問題点がある．

- クラスに含まれるある要素に対する検索では，必要のない要素に対しても復号化を行なう．
- 符号長が長くなるため復号化に時間を要する．
- 型内の一つの要素に対する削除・更新が符号列全体に影響する．

3.1.3 ASN.1/DB における符号化

先に説明した二つの格納方法ではそれぞれに問題点が挙げられた．このため，これらの二つの方法がもつ復号化における欠点を補うために，両者を融合したデータ格納方法が考えられる．この方法では，検索対象となる要素は単純型に対して符号化を行なう．また，検索に必要でない要素は構造型における符号化に埋め込む (図 5)．あらかじめユーザが検索対象になるであろうと考える要素を指定することにより，検索，削除および更新における復号化の冗長性はおこらないと考えられる．

3.2 値に基づくクラスタリング

ASN.1 に特有の型として CHOICE 型がある．これは CHOICE 型として定義される要素のうちどれか一つを選択するというものであり，ASN.1 を用いたアプリケーションには頻繁に現われる型である．要素の選択によりデータ全体のもつ意味が異なる場合，CHOICE 型においてとる値に基づきクラスタリングを行なうことが有効であると考えられる．一般に，値によるクラスタリングは，削除や更新が起こる場合にクラスタリングの再構成を要求するため動的なものとなるため，ユーザからの要求を常に反映することが困難である．しかし，CHOICE 型による選択はそこからさらに型を選択する場合は静的なものであり，ユーザから

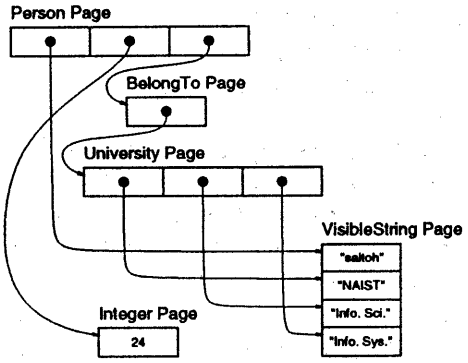

図 3: 単純型符号化によるクラスタリング

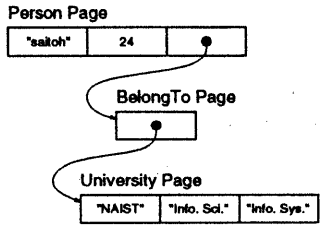

図 4: 構造型符号化によるクラスタリング

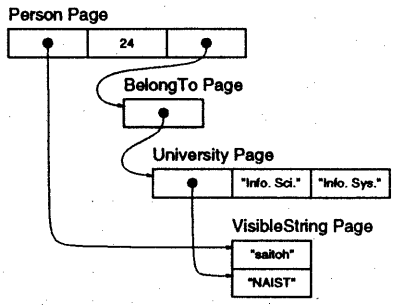

図 5: 符号化を混合させたクラスタリング

の要求が変更することも少ないと考えられるため比較的提供しやすい機能である.

CHOICE 型の値に基づくクラスタリング法では, CHOICE 型による選択をもつデータ構造に対して, CHOICE 型で選択される要素によって格納するページを変更し, 全体を一つのページに格納する. 3 章の例では, CHOICE 型において University 型を選択しているデータと Company 型を選択しているデータを別のページに格納する. この方法での検索における利点は, 全体のデータを一度に主記憶に呼び出せることであり, CHOICE 型による分岐条件のうちの一つを満たしているデータに対するアクセスが一度に行なえる. この方法での問題点としては次のことが挙げられる.

- 画像, 音声といったマルチメディアデータが要素として多数含まれる場合, 一度に全体を呼び出すのは主記憶に負担がかかる.
- 構造型符号化の場合と共通して, 符号化/復号化処理の問題がある.

したがって, 個々の CHOICE 型ごとに, 値に基づくクラスタリングを適用すべきかどうかは異なってくるため, ユーザによって指定することが有効であると考えられる.

4 ASN.1/DB におけるインデクシング

ASN.1 のモジュールがネスト構造になるため, ASN.1/DB のデータベーススキーマもネスト構造を形成する. ここで, 単純型符号化によるクラスタリングを指定していると仮定し, 3 章の例において,

> **所属している University の name が "Osaka-u" である Person の OID をすべて検索する**

という問合せを処理する場合を考える. この場合, インデックスを付加しなければ, すべての Person のデータにおいて, 下位クラスの OID をページから読み込みながらの航行, および name のデータ値の復号化, その値が "Osaka-u" であるかの判定が必要となる.

ODA やディレクトリなど，特定のデータ値に対する検索が行なわれるようなアプリケーションは多いものと考えられるため，ASN.1/DB において検索の効率化のためにインデックスを導入することは有効である．

本章では，ASN.1/DB において導入するインデックス，および，そのインデックスにおいて使用するキーについて説明する．

4.1 ASN.1 データに適したインデクシング

一般に，クラス間のネスト関係を利用したクラス階層インデクシングの代表的な手法として，パスインデックス [1, 2, 6] およびマルチインデックス [2, 6] が知られている．

パスインデックスは，ネスト構造の根から一つの葉までのパスに対して，その葉のデータ値に対してのインデックスを管理する手法であり，キー (データ値) 以外の項目は，そのキーの値をもつパスのパス内に存在するクラスの OID の集合である．また，マルチインデックスは，パスを上位クラスとその下位クラスの組に細かく分割し，その組ごとのインデックスを作成して，パスインデックスと同等の情報を保持する手法である．

それに対して，ODA に代表されるように，ASN.1 を利用しているアプリケーションには数十個にもなるような非常に長いパスをもつネスト構造をとるものが多い．また，ASN.1 特有の CHOICE 型が頻繁に用いられることから，パスの分岐も非常に多くなる．

したがって，ASN.1/DB で用いるインデックスでは，検索における次のような特徴を考慮しなければならない．

1. 検索対象のパスが非常に長い．
2. 検索がなされる複数のパスにおいて，多くの共通部分が存在する．

ここで，パスインデックスを用いた場合，インデックスを付加する複数のパスにおいて，分岐するまでの共通部分のクラスの OID をそれぞれのインデックスにおいて管理しなければならない．この場合，2 の特徴より，かなりの領域の無駄が生じ，更新・削除の手続きが複雑なものとなる．

一方，マルチインデックスを用いた場合，一つのデータの検索において，いくつものインデックスにアクセスしなければならない．これも，1 の特徴から深刻なものとなり，検索効率の面で問題がある．

したがって，ASN.1/DB ではパスインデックス，マルチインデックスの中間的な手法をとることにより，それらを用いた場合の問題点の解決が可能となる．この手法では，パスを上位クラスと下位クラスの組で分割するのではなく，共通部分でまとめて一つのインデックスを作成し，分岐する点でのみインデックスを分割する．例えば 3 章の例において，

Person.belong.univ.name
Person.belong.comp.name

の二つのパスに対してインデックスを付加する場合，この手法では

Person.belong
belong.univ.name
belong.comp.name

の三つのインデックスが作成される．

この手法により，パスインデックスにおける OID データの重複といった問題が解決されるだけでなく，検索においてアクセスするインデックスの数が減少することから，マルチインデックスにおける検索効率の問題も改善される．

4.2 ASN.1/DB におけるインデックスのキー

ASN.1/DB では，インデックスのキーとしてデータそのものではなく，そのデータを符号化した符号列を用いる．符号列をキーにしてインデックスを作成する理由としては，次の二つが挙げられる．

1. 符号列は単なるバイト列であるから，キーの型の考慮が不必要となり，実装が容易になる．
2. 構造型のデータでも容易に符号化できることから，単純型 (ネスト構造の葉) のデータだけでなく，構造型のデータもキーにすることができる．

実際の ASN.1 を用いたアプリケーションにおいても，構造型のデータに対する検索が要求されることが多く，2 は非常に重要なものとなる．ただし，符号列をキーにした場合，同じデータを表す符号列が一意に定まらないという問題が生じる．その理由として，以下の理由が挙げられる．

- 実装によって，整数を表すためのバイト数や，符号列を固定長形式/不定長形式のどちらで表現するのか，などの符号化方法が異なる．
- SET OF の構造型のデータは要素の順序が任意である．

前者は，符号化方法を統一することで解決できる．ただし，符号列が送られてきたときには，符号化しなおす必要がある．一方，後者の解決策としては次の二つが考えられる．

1. 符号化の際に，まず SET OF の各要素をそれぞれ符号化し，それらを辞書順で並べ変えて連結する．
2. SET OF の記述は許すが，その扱いは SEQUENCE OF と同じにし，その要素の順序はアプリケーション側の責任とする．

1 の方法を選択した場合，その作業のオーバーヘッドにより検索効率が低下する．特に，SET OF が入れ子になっている場合には指数オーダで検索効率が低下してしまう．したがって，検索効率を重視するためには，2 の方法がよい．しかし，拡張性を考える上では，1，2 の方法をユーザが指定できることが望ましいと考えられる．

5 ASN.1/DB の実装

本章では，実装中の ASN.1/DB のシステム構成およびデータ格納部について説明する．

5.1 システム構成

システムは図 6のような構成となっている．以下では，各部について簡単に説明する．

[DDL 処理部]

ASN.1 のモジュール定義からデータベーススキーマとなるソースコードを生成する．このとき，

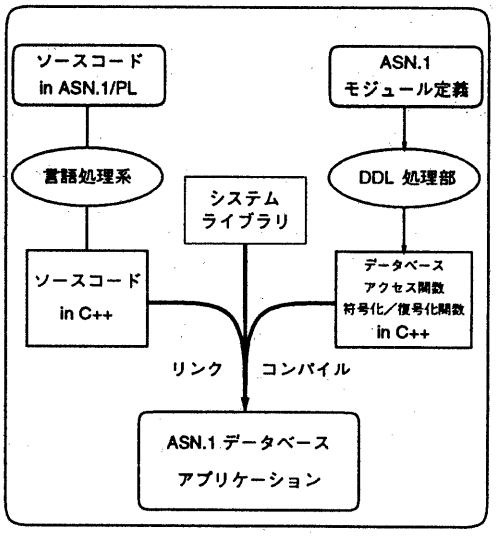

図 6: ASN.1/DB

データベースアクセス関数，符号化/復号化関数等をデータベーススキーマに埋め込む．

[システムライブラリ]

データベースアクセス関数やオブジェクト識別子 (OID) の管理を行なう関数を提供する．また，インデックスの生成やインデックスによる検索に必要な関数もライブラリとして提供する．

[言語処理系]

アプリケーションは ASN.1/PL[4, 5] によって記述される．ASN.1/PL の言語処理系を通してデータベース操作を行なうソースコードが出される．

5.2 クラスタリング機能の利用

3 章で説明したクラスタリング方法は，ユーザがそれぞれの用途に応じたクラスタリング方法を ASN.1 のモジュール定義において指定する．指定のない場合は単純型によるクラスタリングが行なわれるため，構造型に埋め込む要素の定義の直後に次のように記す．

```
-- EMBEDDED
```

"--" につづく文字列は，ASN.1 ではコメントと定義されている．この指定に基づき図5のような格納が行なわれる．CHOICE 型の値に基づくクラスタリングについては，ASN.1 モジュール定義の最後に以下のように記すことで行なわれる．

```
-- CLUSTERING BY BelongTo
```

5.3 インデックス機能の利用

4 章で説明したインデックスも，クラスタリングと同様にユーザが ASN.1 のモジュール定義において指定する．ただし，指定はモジュール定義の最後に次のように記されるものとする．

```
-- INDEX Person.belong.univ.name
```

この場合，Person.belong.univ.name のパスに対して，4 章で提案したインデックスが付加される．また，指定は複数でもよく，構造型をキーとできることから，指定するパスの末端が構造型でもよい．

6 おわりに

本稿では，ASN.1 データを格納するためのデータベースについて論じ，ASN.1 データ特有のデータ構造を考慮したデータ配置およびインデックス手法を提案した．

これまでのシステムの実装は，クラスタリングでは混合型を除く機能，インデクシングでは中間的な手法以外を終え，その動作の確認を行なった．現在，ASN.1/DB および ASN.1/PL を用いて OSI ディレクトリを実装している．その結果として，今回提案したクラスタリングおよびインデックス手法の正当性の検証を含めた性能評価を行なっていく予定である．

謝辞

本研究の一部は，財団法人大川情報通信基金の研究助成によるものである．ここに記して謝意を表す．

参考文献

[1] E. Bertino,C. Guglielmina, "Path-index: An approach to the efficient execution of object-oriented queries", Data & Knowledge Engeering, Vol. 10, No. 1, pp. 1–27, 1993.

[2] E. Bertino,K. Won, "Indexing Techniques for Queries on Nested Objects", IEEE Transactions on Knowledge and Data Engineering, Vol. 1, No. 2, pp. 196–214, 1989.

[3] 春本要, 仲秋朗, 塚本昌彦, 西尾章治郎, 宮原秀夫, 抽象構文記法 1 (ASN.1) に基づくデータベースシステム, 情報処理学会第 97 回データベースシステム研究会報告, pp. 41–50, 1994.

[4] 春本要, 塚本昌彦, 西尾章治郎, ASN.1 データベースプログラミング言語, 情報処理学会第 99 回データベースシステム研究会報告, pp. 249–256, 1994.

[5] K.Harumoto,M.Tsukamoto, and S.Nishio, "A Database System Based on ASN.1: Its System Architecture and Database Programming Language", Proc. of International Symposium on Advanced Database Technologies and Their Integration(ADTI'94), pp. 160–167, 1994.

[6] 加藤和彦, オブジェクト指向データベースシステムの記憶構造, 情報処理学会誌, Vol. 32, No. 5, pp. 532–539, 1991.

[7] 桐葉佳明, 中井正一郎, 有馬啓伊子, 井原慈子, 栗山博, 長谷川聡, 管理情報ベース (MIB) の開発支援環境, 情報処理学会第 86 回データベースシステム研究会報告, pp. 69–77, 1991.

[8] 増永良文, マルチメディア文書の標準データベース格納構造の考察, 情報処理学会第 93 回データベースシステム研究会報告, pp. 83–92, 1993.

[9] 増永良文, ODA に基づくマルチメディア文書データベースの格納方式, 情報処理学会第 47 回全国大会, pp. 4-97–4-98, 1993.

[10] 仲秋朗, 春本要, 齋藤淳, 塚本昌彦, 西尾章治郎, OODBMS を用いた ASN.1 データベースの実現, 情報処理学会第 65 回マルチメディア通信と分散処理研究会報告, pp. 151–156, 1994.

[11] 西山智, 堀内浩規, 横田英俊, 小花貞夫, 鈴木健二, OSI 管理情報ベース (MIB) 用データベースの設計と実装, 情報処理学会第 95 回データベースシステム研究会報告, pp. 59–66, 1993.

[12] 西山智, 小花貞夫, 堀内浩規, 鈴木健二, 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, 情報処理学会論文誌, Vol. 34, No.6, pp. 1486–1496, 1993.