ＯＯＤＢのＯＳＩディレクトリシステムへの適用 ～概要～

5C－8

岩崎 孝夫　山田 広佳　山上 俊之

（株）東芝　情報処理・機器技術研究所

1. はじめに

オブジェクト指向データベース（ＯＯＤＢ）が注目されるようになってから数年が経過した．現在は，実用化の段階に入ってきたということができる．しかし，エンジニアリング分野やマルチメディアを扱った分野への適用例が多い．最近では在庫管理システムへの適用という非エンジニアリング分野での応用例も見受けられるが，まだ経験が少なくその効果も未知数である．そこで，非エンジニアリング分野のアプリケーションとして，ＯＳＩのディレクトリシステムのディレクトリ情報ベース[1]を選び，Ｃ＋＋をデータベース言語とする市販ＯＯＤＢＭＳを利用して試作した．今回はその実装方式について発表する．

本稿ではディレクトリシステムおよび今回の実装方式の概要について述べる．

2. ディレクトリシステム

ディレクトリシステム（ＤＳ）は通信に関わる情報案内のために世界規模で共同して構築される一つの分散データベースとして考えることができる．ディレクトリはＯＳＩ応用層のサービス要素の１つで，ＩＳＯ／ＣＣＩＴＴで標準化が進められている．図１にＤＳの機能構成を示す．図中のＤＳＡ（Directory System Agent）がディレクトリサービスを通じて，実際のデータベース機能をサービスする．このデータベース部分はディレクトリ情報ベース（ＤＩＢ:Directory Information Base）と呼ばれ，ＤＳの核となる．ＤＳＡの中の実現方法については規定されていないが，ＤＩＢのモデルは標準化されている．このモデルは複雑な構造を持ち，オブジェクト指向に近い概念を用いているため，ＯＯＤＢと相性が良いと思われる．

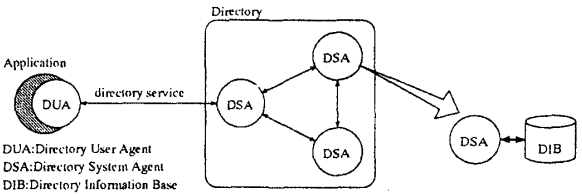

図１ ＤＳの機能構成

Implementation of OSI Directory System using OODB
Takao Iwasaki, Hiroyoshi Yamada, Toshiyuki Yamagami
Information Systems Engineering Labolatory,
TOSHIBA Corporation

3. DIB

3.1 ディレクトリ情報

実世界の対象物の情報を集めた管理単位をエントリと呼び，ＤＩＢではこのエントリをツリー構造に沿って管理する．このツリーをディレクトリ情報ツリー（ＤＩＴ:Directroy Information Tree）と呼ぶ．ＤＩＴはroot（／）と呼ばれるノードから始まり，各エントリはこのツリーの１ノードとして結びつけられる．各エントリを区別するために識別名（ＤＮ:Distinguished Name）を用いる．ＤＮはＤＩＴのツリー構造に沿って記述したものであり，相対識別名（ＲＤＮ:Relative Distinguished Name）と呼ばれる情報の並びとして構成される．その様子を図2に示す．

エントリは実世界の対象物を表す．図2の例の鈴木太郎さんの場合，｛名前=鈴木太郎｝，｛生年月日=19YY.MM.DD｝，｛入社年度=19yy｝といった｛型=値｝で表現することのできる属性を持っている．

3.2 エントリ

前節で述べたようにエントリは対象物を表すが，必ずしもエントリと対象物が一対一に対応するとは限らない．○×株式会社から株式会社マルペケに社名変更した場合，○×とマルペケは同じ会社（対象物）を

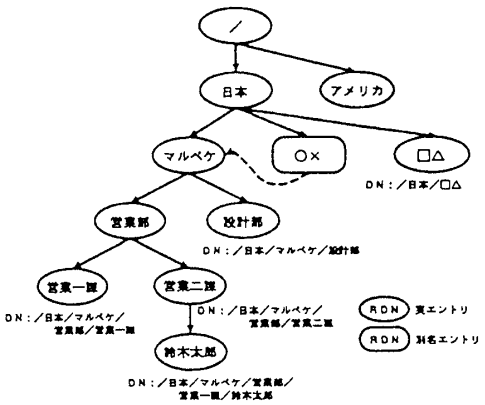

図2 ＤＩＴの概要

指す．図2のようにエントリ○×がマルペケを指すようにしておけば（図中の破線），旧社名で検索要求があっても困ることはないし，○×の鈴木太郎さんとマルペケの鈴木太郎さんを同一人物として扱うことができる．この時，マルペケを実エントリ，○×を別名エントリと呼び区別する．

3.3 ディレクトリスキーマ

ディレクトリ情報の様々な定義をディレクトリスキーマと呼ぶ．この定義にはオブジェクトクラスの定義，ＤＩＴ構造定義，などが含まれる．オブジェクトクラスはエントリの持つべき属性などを定義するもので，オブジェクト指向で言うクラスと同様の概念である．ＤＩＴ構造定義は，ＤＩＴを構成する際に親子関係になりうるエントリのオブジェクトクラスを定義する．

4. 構成

今回は既にあるモジュールを活かし[2][3]，ＤＩＢの部分を独立したプロセスにより管理する方式とした．この方式ではＤＩＢを管理するプロセスは2つの機能に分けることができる．一つはディレクトリ情報のアクセスを行なう機能で，もう一つはディレクトリスキーマを管理する機能である．筆者らはこれらの役割から別々のプロセスとして実現した．前者をドライバプロセスと呼び，後者をメーカプロセスと呼ぶ．この様子を図3に示す．

ドライバプロセスは常駐しており，ＤＳＡからの要求があるごとにディレクトリ情報のデータベースをアクセスし，要求に応える．一方，メーカプロセスは管理者がディレクトリスキーマの保守を必要とする時，スキーマ管理ツールを通して起動され，ディレクトリスキーマのデータベースをアクセスし，要求に応える．Ｃ＋＋を利用しているので，もしスキーマが変更になると，データベースをアクセスするプログラムの再コンパイルが必要となる．したがって，プロセスを2つに分け，アクセスするデータベースを適切に分割することにより，ディレクトリスキーマの変更に対しても柔軟に対応することが可能となる．

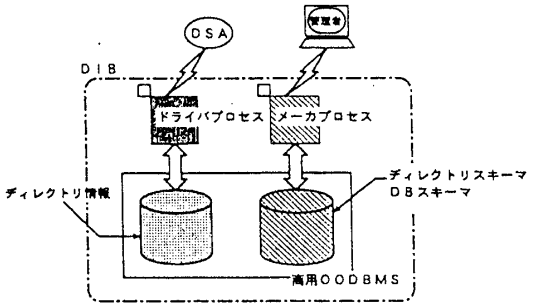

図3 ＤＩＢの構成

5. おわりに

今回は非エンジニアリング分野のアプリケーションとして，ＯＳＩディレクトリシステムのディレクトリ情報ベースを選んだ．これは既に様々なＤＢＭＳを用いたり，ＤＢＭＳを利用せずに実現した例が報告されている．しかし，これらはディレクトリシステムの一実現方法としての報告であった．筆者らはＯＯＤＢの非エンジニアリング分野での効果を探るという観点から今回の試作を行なった．この試作では，ディレクトリ情報システムのモデルをそのままオブジェクトとして表現し，スキーマの変更などにも柔軟に対応できるよう実現した．

参考文献

[1] ISO/IEC 9594-1～8/CCITT X.500 シリーズ

[2] 増尾他，“ディレクトリシステムのＤＩＢ実装方式 (1) ＤＩＢファイルのしくみ”，情報処理学会第４６回全国大会,1993

[3] 池ノ谷他，“ディレクトリシステムのＤＩＢ実装方式 (2) ＤＩＢファイルの操作”，情報処理学会第４６回全国大会,1993