ＯＯＤＢのＯＳＩディレクトリシステムへの適用 ～データ操作～

5C－9

山田 広佳　山上 俊之　岩崎 孝夫

（株）東芝　情報処理・機器技術研究所

1. はじめに

筆者らは，ＯＳＩディレクトリシステム（以下ＤＳ）のデータベース部分を商用ＯＯＤＢを用いて構築を行った．ここでとりあげたＤＳはオブジェクト指向を意識した仕様になっており，このことはクラス，そしてその継承という概念を持ち込んだこと等をその例としてあげることができる．

また一方，オブジェクト指向の特長のひとつとして強力なモデリング能力があげられる．この特長により実世界を自然にデータベース上へ写像することが容易になると一般にいわれている．ただ，性能上の観点からは概念とその実装とは別のアプローチで行った方が効率の良いことが多々あるが，ここではこのモデリング能力を重視し極力概念に忠実に実装を行うことにした．

ＤＳにおいて，ＤＳＡ (Directory Service Agent) およびＤＩＢ (Directory Information Base) はその各種情報を管理しそのアクセスを司り，ドライバプロセスはそのうちＤＳにおけるサービスの対象データの管理・アクセスを司っている [1]．以下ではこのドライバプロセスについて述べる．

2. ドライバプロセスの構成

2.1 エントリの基本構成

ドライバプロセスの管理するデータはＤＩＴ (Directory Information Tree) と呼ばれる木構造により管理され，この木においてエントリと呼ばれる各節点に情報が格納される．このエントリは実と別名との本質的に性格の異なる２種類に分類することができる．実エントリは情報の管理を主目的とするものであり，別名エントリは自エントリへのアクセスを透過的に他のエントリへのアクセスに置き換えることにより，別途検索パスを提供する．そこで，ここではこのそれぞれを基本クラスとして別途作成しておくこととした．それ以外のクラスは，これらのメンバおよびメンバ函数を継承することによりメーカプロセスにより定義される [2]．

最初に述べたように，ここではＤＳでの概念に忠実に実装を行なうことにしたので，属性型をそのままインスタンス変数として実現することにした．また，複数の属性値をとるものが存在しうることから，そのインスタンス変数は属性値の集合の形で値をとるものとした．エントリの基本構成例を図1に示す．ここに示すような表現をとることにより，生年月日のように唯一値しか取りえないものも，趣味のように複数値を取りえるものも区別なく扱うことができる．

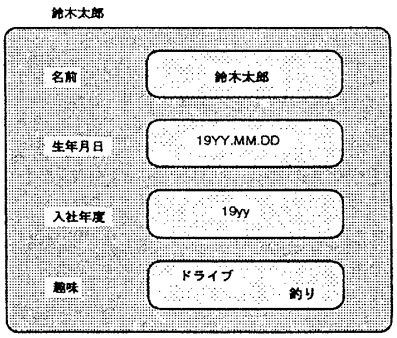

図１ エントリの基本構成

この各クラスにおける基本的なメンバ函数は，ＤＳのサービスの性格から次の２種類を用意すればよいと考えられる．

- 自エントリの各属性に対する参照・更新
- 下位エントリの検索

実エントリにおいては，指定されているエントリが自

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

分であれば参照または更新処理を行い，そうでなければ１レベル下から条件に合うエントリを検索しそのエントリに対し継続処理のメッセージパッシングを行えばよい．また別名エントリの場合には，その指し示すエントリに対し同じメッセージを転送するだけで良いことになる．

2.2 別名エントリの構成

また別名エントリにおいては，その指し示すエントリへのＤＮを属性として格納している．しかし，別名の展開を行なう都度このＤＮから指定のエントリを取り出していては非常に効率が悪い．このため，事前に指定のエントリを獲得しておき，この属性とは別にエントリそのものの情報を別名エントリ内に格納しておくこととした．

別名エントリの構成例を図2に示す．

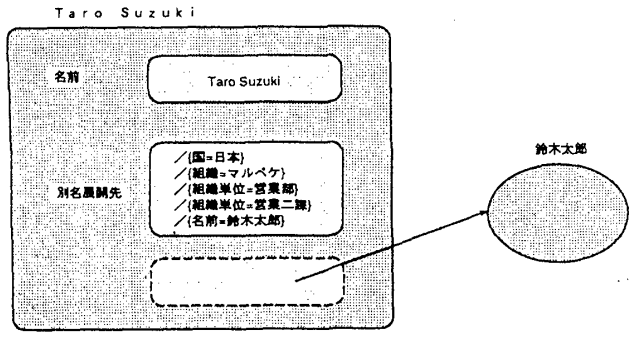

図２ 別名エントリの構成

このような構成をとることにより，その属性に対する更新がなされた場合は同時に指し示すエントリそのものの情報を書き換える必要が生じる．ここでは，各属性型ごとそれをアクセスするメンバ函数が定義されているので，別名展開先を示す属性型に値を格納するメンバ函数においてのみ，その手続き中に指し示すエントリを更新する処理を入れることにより実現している．このことから，その属性値の変更をトリガとして，他の領域を書き換え常に一貫性を保つようになっている．

3. 評価

今回実装したＤＩＢは，従来のファイルベースのもの [3][4] に比べ全般的に処理が遅く，特に最初のエントリ（ルート）の取り出しに時間がかかっている．これは，全オブジェクトが同列に格納されているために，そこからルートを選択するのに時間がかかっていると推測される．このため，ＤＩＴを構成するオブジェクト数の増大に伴って，ルート獲得までの時間がより一層大きくなる．これを解決する手段としては，このルートとそれ以外のものの格納領域を分けておくことが考えられ、そうすればルート獲得のときはその領域だけの検索を行えば良い．

また，ＤＩＴは実エントリと別名エントリが混在して形成されている．今回の実装においてはこの２種類のエントリをそのまま区別なく扱っており，このことが必然的に動的束縛の多用につながり性能劣化の一因となっていると考えられる．この動的束縛を低減させることは可能ではあるが，その場合は逆にモデリングに制約がつくのでオブジェクト指向の特長である表現力を損なう結果となってしまう．性能に対する考察は常に必要であるが，生産性，保守性等を考慮に入れ，ここではその表現力の方を優先することとした．

4. おわりに

オブジェクト指向のモデリング能力は強力であり，実世界に忠実にモデルを作成することが可能となる．しかしその反面，動的束縛等による性能低下の問題が考えられ，そのトレードオフを考慮に入れる必要があるかと思われる．

オブジェクト指向による豊かな表現力を損なわず，性能向上のための技術を検討していきたい．

参考文献

[1] 岩崎他，“ＯＯＤＢのＯＳＩディレクトリシステムへの適用～概要～”，情報処理学会第４７回全国大会，1993

[2] 山上他，“ＯＯＤＢのＯＳＩディレクトリシステムへの適用～スキーマ管理～”，情報処理学会第４７回全国大会，1993

[3] 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

[4] 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