6D－9

ネットワーク型データベースアプリケーションのオブジェクト指向データベースへの移行

山上 俊之　岩崎 孝夫

（株）東芝　情報・通信システム技術研究所

1　はじめに

オープン化の進展にともなって，ネットワーク型データベース(NDB)の既存データ／アプリケーションを，オープンなプラットフォームにどのように移行するかが，情報システム部門にとって大きな問題となっている．

NDB アプリケーションの受け口としては，リレーショナルデータベース(RDB)が考えられるが，部品展開のようにデータナビゲーション性能が要求されるアプリケーションでは性能面で問題がある．このようなアプリケーションの１つの受け口としてオブジェクト指向データベース(OODB)を考える必要もある．

我々は，情報システム部門で稼働している部品表展開を題材に，OODB への試験的な移行作業を行なったので，その概要を報告する．

2　移行要素／手順

システムの移行にあたって，アプリケーション，データベース，ユーザインタフェースの３つを構成要素としてとらえた．NDB 環境と OODB 環境の大まかな構造の違いを図 1に示す．

現行のシステムはバッチ処理が中心であり，ファイル入出力ができれば事足りるが，ターゲット環境(OODB 環境)では，将来の拡張やインタフェースの拡張などを考慮して，ユーザインタフェース部分の拡張性に重きをおいた．ユーザインタフェースに依存する部分は他の構成要素とはできるだけ切り離した構造としている．

OODB を移行先に据える場合には，アプリケーションとデータベースが少なからず密着した構造になるのが特徴である．したがって，伝統的なデータベースアプリケーションにおける以下の２つの側面を統合してとらえる必要がある．

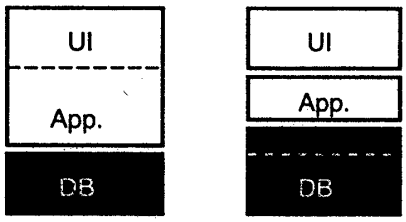

図 1: NDB/OODB 環境での構成要素の相違

静的側面 (従来のデータベース)
既存レコード型の分析，論理データベース変換

動的側面 (従来のアプリケーション)
COBOL プログラムの分析，メソッドの作り込み

次節以降，この２つの移行方法について詳述する．

3　静的側面

NDB の論理設計をどのように変換するかについては，リレーショナルデータベースへのマイグレーションを対象としたものではいくつかの研究がなされている[1]．他方，オブジェクト指向モデルから RDB へと変換する方法についても[2]などでなされているので，その比較と検証から，次のような手順で変換を行なった．

1. 実体を表すレコード型に対してクラスを作る．
2. 多対多関連のレコード型に対してクラスを作る．
3. 汎化特化関係を見出す．

An application of OODB: The Migration of a Legacy NDB Application.
Toshiyuki Yamagami, Takao Iwasaki
Information & Communication Systems Labolatory,
TOSHIBA Corporation

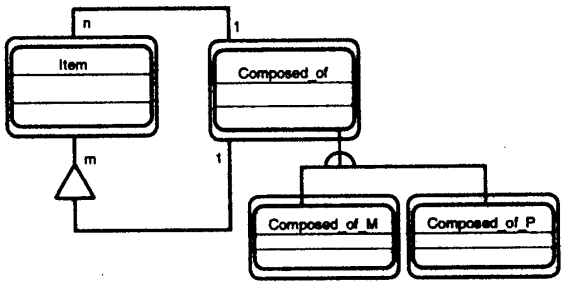

図 2: 変換後のオブジェクト図

我々が移行した，部品表展開アプリの詳細には触れないが，この様な手続きによって変換されたオブジェクト図の基本部分は図 2のようになった．

部品を表すクラス (Item) は，“〜から構成される”という関連を表すクラス (Composed_of) を介して再帰的な関係を構成する．特徴的なものは“〜から構成される”クラスから汎化特化関係で，複数のクラスが導出される点である．これは，部品の種類の違いを多対多関連のレコード型で区別していたためで，現在のオブジェクト指向の考えでは，部品の種類の違いによる汎化特化関係は，部品をあらわすクラス (Item) の方に現れてくるのが自然である．このようにならなかったのは NDB の能力不足による．

現在のオブジェクト指向の考えに即したように変更する事も可能であるが，我々は，依存データの移行を最重視して，図 2をベースに移行を行なう事にした．

4 動的側面

今回の試験的な移行では，既存のサブプログラム毎にメソッドを割り当て，設計時に各メソッドに共通の部分を共有するなどのブレークダウンを行なっていくという手法をとった．

正直なところ，この作業にかなりの時間を要しており，これがオブジェクト指向データベースへの移行に関する一番のネックになると考える．

問題点は，データベースのメソッドとして実装する部分を決める際に，結局はCOBOL プログラムをすべて読む必要があったこと．また，第1図では，ユーザインタフェース，アプリケーション，データベースという3つの構成を示したが，旧資産である COBOL プログラム自体が，ユーザインタフェース部分が密に融合しているものが多く，この部分を切り離すことにも時間をとられることが明らかになった．

システムの移行にかかる工数を考えると，この部分をうまく考えないといけない事が明らかになっている．ひとつの解決方法として，オブジェクト指向的ではないが，あまりサービスメソッドを作り込まずに，手続き的な元々のロジックを尊重するやり方も候補として考える必要がある．

5 結論

データベースの静的な構造を移行することは，さほど難しくはなく，ある意味で直線的に対処できることが明らかになった．しかし，オブジェクト指向データベースに特有な，動的な側面の移行が大きな問題となる事も明らかになった．

COBOL プログラムの資産の中には，構造化されていないものや，ユーザインタフェース部分が密に融合しているものなど，旧資産の質の悪さが移行時に問題となることが多い．COBOL プログラムのリエンジニアリング手法なども，いろいろと発表されており，それとの組合せを考えて行く必要性を感じている [3].

6 おわりに

今後は，COBOL プログラムのリエンジニアリング手法との統合による移行性の向上を主眼として，進めていく予定である．

参考文献

[1] Gillenson,M.L. Physical Design Equivalencies in Database Conversion, *Commun. ACM Vol 33 No 8 (Aug. 1990) pp 120-131.*

[2] J.Rumbaugh et al.,*Object-Oriented Modeling and Design*, Prentice-Hall, Englewood Cliffs, N.J.,1991.

[3] Tan and Ling, Recovery of object-oriented design from existing data-intensive business programs, *Inf. and Soft. Tech. Vol 37 No 2(1995) pp 67-77*