D_011

言語透過的なネイティブオブジェクトデータベースの研究
Native-Object Database Management System with Language Transparency

高野 光弘† 今泉 貴史‡
TAKANO Mitsuhiro IMAIZUMI Takashi

1. はじめに

ソフトウエア開発の大規模化にともない，オブジェクト指向プログラミング言語が広く普及してきた．また，さまざまな対象領域の問題を解決するために，オブジェクト指向プログラミング言語の種類も増えている．プログラミング言語の種類により，得意とする対象領域が異なるため，ある言語の実行環境でインスタンス化されたオブジェクトの振る舞いを別の言語の実行環境で利用することができれば，それぞれの言語の特徴を生かしたオブジェクトの利用ができる．しかし，ある言語で記述されたオブジェクトの振る舞いはその言語専用のものとなるため，他の言語では解釈できない．ある言語で記述したオブジェクトの振る舞いを別の言語で利用するにはオブジェクトの振る舞いの記述を移植する作業が必要となる．また，インスタンス化されたオブジェクトは状態をもっているため，振る舞いの記述を移植するだけではオブジェクトを別の実行環境でインスタンス化することはできない．そのため，オブジェクト指向プログラミング言語はオブジェクト指向という共通の枠組みを持っているにも関わらず，ある言語の実行環境でインスタンス化されたオブジェクトを別の言語の実行環境で扱うことができない．

これを実現するための手法として，オブジェクトを異なる実行環境に移動させる方法や，異なる実行環境から利用できるインタフェースを提供する方法などが提案されている．しかしこれらの方法では，実行環境を構成する言語が異なる場合に移動が困難であったり，オブジェクトが常にインスタンス化された実行環境に存在しなければならないといった問題点がある．

本研究ではこの問題を解決するために，言語透過的なネイティブオブジェクトデータベースを提案する．ある言語の実行環境でインスタンス化されているオブジェクトについて，同じ状態と振る舞いを持つオブジェクトを別の実行環境で復元することを目指す．これにより，さまざまな言語の実行環境間でオブジェクトの移動を行うことができ，実行環境同士が状態を持ったオブジェクトを送受信しあうことが可能となる．そして，振る舞いを持つオブジェクトを送受信することで，オブジェクトを配信するサービスや分散環境の開発を支援する．異なる言語の実行環境をまたいでオブジェクトの移動を行うためには，それぞれの言語によるオブジェクトの振る舞いの記述を用意する必要がある．そのために，各プログラミング言語とは独立した中間言語によりオブジェクトの振る舞いを記述し，それをデータベースに格納する．そして，実際に利用したい実行環境に合わせたプログラミング言語の記述へと変換する．また，インスタンス化されたオブジェクトを移動させるためにオブジェクトの状態をデータベースに格納して永続化する．さらに，オブジェクトの振る舞いと状態をデータベースに格納することを生かし，オブジェクトの統一的な管理とプログラム資産やデータ資産の集約を考える．

†千葉大学大学院 自然科学研究科
‡千葉大学 総合メディア基盤センター

2. 言語透過的なオブジェクトの利用

言語透過的なオブジェクトの利用とは，ある言語により構成された実行環境に存在するオブジェクトを別の言語により構成された実行環境から利用することである．ここでは言語透過的にオブジェクトを利用するための既存の手法について説明する．

2.1 整列化と非整列化によるオブジェクトの移動

整列化とはインスタンス化したオブジェクトから状態を取り出すための技術であり，非整列化とは整列化により取り出した情報を元にオブジェクトを復元するための技術である．整列化により出力するデータの構造と非整列化に入力するデータの構造を一致させ，このデータの構造をオブジェクトの属性の構造と対応付けることで，移動元の実行環境を構成する言語と移動先の実行環境を構成する言語が異なっていてもオブジェクトの状態を移動させることができる．ただし，整列化と非整列化はオブジェクトの状態を移動させることはできるが，オブジェクトの振る舞いを移動させることはできない．そのため，移動元と移動先の実行環境を構成する言語が異なる場合，移動元の言語で記述されたオブジェクトの振る舞いを移動先の言語で記述し直すなどの移植作業が必要となる．移植作業には移動元の言語と移動先の言語を習得した技術者を要する．また，移植作業を行うことで，ひとつの言語によるオブジェクトの振る舞いの記述がふたつの言語によるオブジェクトの振る舞いの記述に分かれることになる．そのため，オブジェクトの振る舞いを変更するためにふたつのオブジェクトの振る舞いの記述を変更する必要が生じる．

2.2 分散オブジェクト

分散オブジェクトとは，ある実行環境に存在するオブジェクトへのインターフェースを提供することで，他の実行環境からもオブジェクトを言語透過的に利用することを可能にする技術である．つまり，オブジェクトに対し，共通の呼び出し規約を定めることで，オブジェクトの実行環境をまたいだ利用を可能としている．つまり，呼び出し規約に準拠することで，環境の違いを意識せずにさまざまな実行環境にあるオブジェクトを利用することができる．実行環境をまたいだオブジェクトの利用により，実行環境同士が作業を分担し効率的に処理を行うこともできる．しかし，言語をまたいでオブジェクトを利用するための手段として分散オブジェクトを用いる場合，利用したいオブジェクトが存在する実行環境に対しメソッドを呼び出すため，それぞれの実行環境が独立していてはオブジェクトを利用することができない．オブ

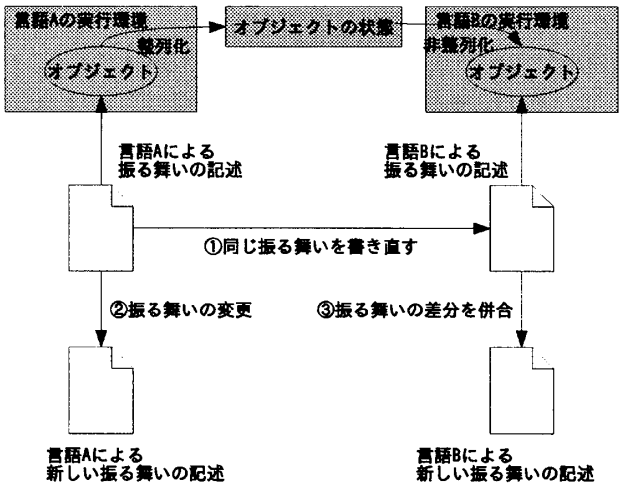

図 1: 整列化と非整列化を利用したオブジェクトの移動

ジェクトを移動させる方法を用いる場合は，実行環境をまたいでオブジェクトを移動させることができれば，それぞれの実行環境同士が独立していてもオブジェクトを利用することができる．

2.3 オブジェクトデータベース

オブジェクトの永続化に主眼をおき，オブジェクトを言語透過的に利用する方法としてオブジェクトデータベースを利用した方法がある．オブジェクトデータベースではオブジェクトの状態に加え，オブジェクトの振る舞いをデータベースに格納することで永続化を実現している．オブジェクトデータベースでは条件にあわせてオブジェクトの関連や継承を扱うことができる．また，オブジェクトデータベースを利用したアプリケーションプログラムでは格納したオブジェクトをクエリー言語で参照できるほか，参照したオブジェクトに対してメソッドの呼び出しを発行し，実行結果を受け取ることもできる．つまり，オブジェクトデータベースは既存のリレーショナルデータベースに比べ，オブジェクト指向プログラミング言語の表現力を生かしたデータの永続化が可能となる．オブジェクトデータベースでは異なる言語の実行環境から同じオブジェクトを利用するために，それぞれの言語に対応したインターフェイスを提供している．しかし，このインターフェースは，本質的にはオブジェクトデータベースに対してメソッドの呼び出しを発行する仕組みであり，オブジェクト自身はオブジェクトデータベースの内部に存在している．そのため，メソッドの処理はオブジェクトデータベースの内部で行われ，多くのオブジェクトを管理するオブジェクトデータベースでは過負荷の原因となる．また，オブジェクトデータベースに格納するオブジェクトの振る舞いはオブジェクトデータベースに固有の言語によって記述する．そのため，アプリケーションプログラムの実行環境を構成しているオブジェクト指向プログラミング言語を習得した技術者であっても，オブジェクトの振る舞いの変更が困難となる場合がある．また，分散オブジェクトと同様に実行環境がオブジェクトデータベースと独立した状態ではオブジェクトを利用することができない．

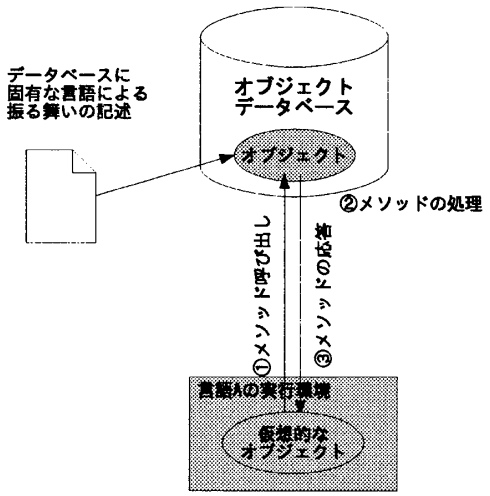

図 2: 既存のオブジェクトデータベースを用いたオブジェクトの利用

3. 提案手法

本研究では実行環境が独立した状態で言語透過的にオブジェクトを利用し，本来ならば実行環境を構成する言語ごとに必要となるオブジェクトの振る舞いの記述をひとつに統一する手法を提案する．オブジェクトを利用する実行環境にネイティブオブジェクトをインスタンス化し，オブジェクトの状態と振る舞いを統一的に管理するために，言語透過的なネイティブオブジェクトデータベースを用いる．ここでネイティブオブジェクトとは，他の実行環境に依存せずに利用できるオブジェクトのことである．

ネイティブオブジェクトをインスタンス化するためには，オブジェクトの状態とオブジェクトの振る舞いの記述が必要となる．既存のオブジェクトデータベースではオブジェクトの状態のみをデータベースに格納しているため，ネイティブオブジェクトをインスタンス化できない．言語透過的なネイティブオブジェクトデータベースではオブジェクトの状態に加え，オブジェクトの振る舞いの記述もデータベースに格納する．

3.1 オブジェクトの振る舞いの記述

ネイティブオブジェクトの振る舞いの記述は，移動先の実行環境を構成する言語ごとに必要となる．これを実行環境を構成する言語ごとに用意すると，オブジェクトの振る舞いを変更するたびにそれぞれの記述を変更しなければならないため，管理作業が煩雑になる．これを解決するために，さまざまな言語と相互に変換が可能な中間言語を用いる．言語透過的なネイティブオブジェクトデータベースでは，この中間言語を用いてオブジェクトの振る舞いを記述し，データベースに格納する．データベースは中間言語によるオブジェクトの振る舞いの記述を解釈し，既存のオブジェクトデータベースと同様なオブジェクトの振る舞いの呼び出しを行う．

3.2 オブジェクトの状態

言語透過的なネイティブオブジェクトデータベースでは，オブジェクトの振る舞いに含まれる情報をオブジェクトの状態の格納に使う．

既存のオブジェクトデータベースでは，オブジェクトの状態を保存できる構造のテーブルなどを用意するためにデータベースのスキーマが必要となる．オブジェクトデータベースのスキーマを記述するためには，オブジェクトの属性の名前や型についての情報が必要となる．この情報はオブジェクトの振る舞いの記述に含まれているため，言語透過的なネイティブオブジェクトデータベースではスキーマの記述は不要である．

オブジェクトの状態の問い合わせや更新については，既存のオブジェクトデータベースと同様にトランザクション処理を行う．

3.3 ネイティブオブジェクトのインスタンス化

構成している言語が異なる実行環境をまたいでネイティブオブジェクトをインスタンス化する場合には，それぞれの実行環境を構成している言語によるオブジェクトの振る舞いの記述が必要となる．

言語透過的なネイティブオブジェクトデータベースは格納した中間言語によるオブジェクトの振る舞いの記述を解釈し，実行環境を構成している言語の振る舞いの記述へと変換する．その後，生成したオブジェクトの振る舞いの記述と格納しているオブジェクトの状態から実行環境にネイティブなオブジェクトをインスタンス化する．なお，オブジェクトの振る舞いの記述には状態の初期値についての情報が含まれる．これを利用することで，実行環境とデータベースのいずれの環境でも初期状態のオブジェクトをインスタンス化できる．

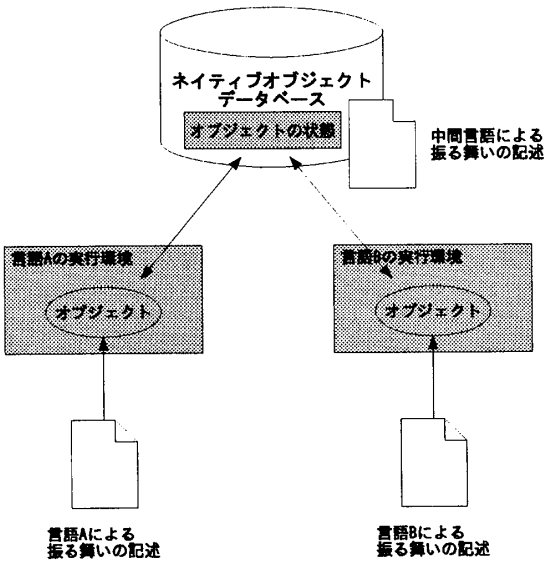

図 3: ネイティブオブジェクトデータベースを用いたオブジェクトの移動

実行環境がオブジェクトの機能を利用するときにネイティブオブジェクトをインスタンス化する必要がなければ，既存のオブジェクトデータベースと同様にデータベースに対してメソッドの呼び出しを行う．呼び出しを受け取った言語透過的なネイティブオブジェクトデータベースはデータベースに格納しているオブジェクトの状態とオブジェクトの振る舞いの記述を解釈し，メソッドの呼び出しの処理を行い，必要があればデータベースに格納しているオブジェクトの状態を更新する．

4. 考察

本節では，提案した言語透過的なネイティブオブジェクトデータベースについて考察する．

4.1 オブジェクトの振る舞いの記述

さまざまな言語と相互に変換が可能な中間言語によるオブジェクトの振る舞いをデータベースに格納しているため，データベースからさまざまな言語の振る舞いの記述を復元することが可能となる．また，オブジェクトの振る舞いを記述するためにデータベースに固有な言語を用いる必要はなく，開発者が習熟した言語を用いたオブジェクトの振る舞いの変更が可能となる．開発者は言語透過的なオブジェクトデータベースに対し，習熟した言語によるオブジェクトの振る舞いの記述を要求し，その振る舞いの記述を変更する．そして，その振る舞いの記述を中間言語による記述に逆変換し，データベースに格納することでオブジェクトの振る舞いを変更する．さらに，データベースに格納したオブジェクトの振る舞いの記述からさまざまな言語によるオブジェクトの振る舞いの記述に変換することが可能であるため，同じオブジェクトの振る舞いを利用するサーバ/クライアント型のプログラムの作成などにおいて，それぞれの実行環境を構成する言語が異なる開発が容易となる．

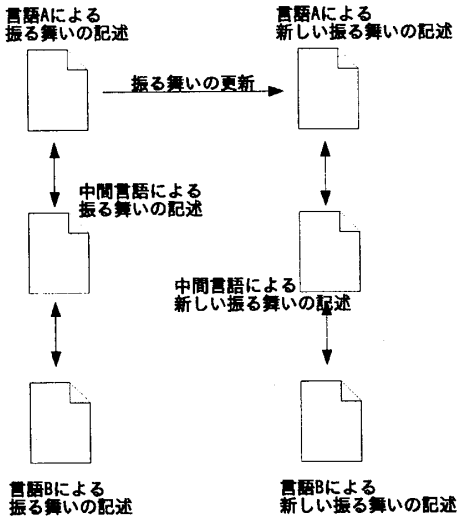

図 4: 実行環境を構成する言語と中間言語の相互変換

4.2 ネイティブオブジェクトの復元

既存のオブジェクトデータベースではデータベースと実行環境が独立した環境ではオブジェクトが機能できない．提案する手法では，実行環境にネイティブオブジェ

クトを移動することができれば，その後はデータベースと実行環境が独立した環境でもオブジェクトが機能できる．これにより，オブジェクトを相互に送受信するアプリケーションや，オブジェクトを配布するオブジェクトサーバなどへの応用が可能となる．

ネイティブオブジェクトを復元する必要がなければ，データベースのオブジェクトにメソッドの呼び出しを行うことで，分散オブジェクトとして利用することが可能である．

4.3 ネイティブオブジェクトの更新

ネイティブオブジェクトはオブジェクトの振る舞いを他の実行環境に依存せずに処理することが可能であるため，実行環境のオブジェクトを更新する目的に使うことが可能である．オブジェクトを更新するときにデータベースと情報のやり取りを行えば，更新後はデータベースと実行環境がが独立している環境でもオブジェクトは機能できる．分散したプログラムの配置を行っている場合でも，データベースにあるオブジェクトの振る舞いを配布し，各分散環境の更新を行うことが可能となる．また，データベースが格納しているオブジェクトの新しい状態として，実行環境に存在するオブジェクトの状態を与え，状態の更新をすることも可能である．

4.4 ネイティブオブジェクトの移動

ネイティブオブジェクトは独立した環境で機能するため，ネイティブオブジェクトを利用し，エージェントを構成できる．既存のオブジェクトデータベースを利用した場合，本質的にはオブジェクトデータベースに対してのメソッドの呼び出しで機能し，オブジェクトを利用する実行環境がデータベースと独立した環境では利用できない．そのため，データベースとオブジェクトを切り離し，実行環境に配置することはできない．本手法によるネイティブオブジェクトを利用した場合，オブジェクトを利用する実行環境とデータベースが独立した環境で利用できるため，オブジェクトはデータベースとは関係なく，実行環境を移動することができる．

さらに，ネイティブオブジェクトはデータベースや実行環境の過負荷を解消する目的に利用することが可能である．資源が限られた実行環境にあるオブジェクトでコストの高い処理の呼び出しを行う必要があるときは，資源が十分な実行環境にオブジェクトを移動させてから処理の呼び出しを行い，再び資源が限られた実行環境にオブジェクトを戻すことで負荷の軽減を図ることが可能となる．

また，資源が十分な実行環境ではネイティブオブジェクトを復元するときに用いたオブジェクトの振る舞いの記述を利用し，新たな言語透過的なネイティブオブジェクトデータベースとして機能でき，レプリケーションすることも可能である．

4.5 オブジェクトの振る舞いの管理

さまざまな言語と相互に変換可能な中間言語によるオブジェクトの振る舞いの記述をデータベースに格納する．これを管理することでプログラム資産を統一的に管理することが可能となり，一度ある言語で記述したオブジェクトの振る舞いは二度と記述する必要がない．また，履歴を管理することで，プログラム資産の統一的なバージョン管理が可能となる．履歴の管理にはオブジェクトの振る舞いの記述を解釈することにより得られる構文木を用いた管理を行うことができる．

プログラム資産を統一的に管理することでさまざまな面で開発を支持することができる．たとえば，アプリケーションプログラムを開発するときに提案したデータベースに格納したオブジェクトを利用し，各環境で常に同じバージョンのオブジェクトの振る舞いを利用することができる．これにより開発環境ごとの差異を埋め，開発の効率化を図ることが可能となる．オブジェクトの振る舞いをデータベースが解釈できるため，テスト作業をデータベースに格納し，自動化することなども可能である．また，提案したデータベースを利用している運用環境を更新する場合はオブジェクトのバージョンを指定するだけで作業を完了することが可能となる．

5. まとめ

言語透過的なネイティブオブジェクトデータベースにより，ネイティブオブジェクトを移動させることが可能となる．また，オブジェクトの振る舞いをさまざまな言語と相互に変換できる中間言語で表現することにより，さまざまな言語のプログラム資産をまとめることが可能となる．言語透過的なネイティブオブジェクトにより分散環境の充実やデータベースに依存しないオブジェクトの利用を図ることができた．

今後はオブジェクトデータベースの詳細な機能について調査し，永続化したいオブジェクトの属性が他のオブジェクトへの参照を含んでいる場合の処理方法などを検討する．また，ある言語で記述したオブジェクトの振る舞いを中間言語による記述に変換し，さらに他の言語のオブジェクトの振る舞いの記述として復元する方法についても調べる．中間言語としてダイアグラムなどを利用することについても検討する．そして，オブジェクトデータベースに中間言語による記述を言語透過的なオブジェクトとして格納し，統一的に管理することによる利点を考える．

解消すべき問題点としては，言語透過的なオブジェクトの記述を行う中間的な言語の記述能力が低い場合に他の言語の能力を十分に生かせないことや，オブジェクトの振る舞いが新しくなったときに，すでにデータベースに永続化されているオブジェクトの状態を新しい振る舞いに対応させなければならないこと，オブジェクトの振る舞い以外の成果物や開発途中で中間言語に変換できない段階のオブジェクトの振る舞いを管理することが難しいことなどがあげられる．

参考文献

[1] ECMA-355： Common Language Infrastructure (CLI), http://www.ecma-international.org/publications/standards/Ecma-335.htm

[2] InterSystems: Cache, http://www.intersystems.co.jp/cache/

[3] db4objects,Inc.： db4objects, http://www.db4o.com/

[4] 高野 光弘：言語をまたがるオブジェクトマイグレーションの実現法，2004

[5] 東海林 健志, 久住 貴史, 西 宏昭, 宮川 治, 大山 実: オブジェクトの永続化と RDB 設計からの開放，FIT2005