スクリプト言語を用いた移動計算機向けの永続オブジェクトシステム

6D－5

Wisut Sae-Tung　大森 匡　星 守
電気通信大学 大学院 情報システム学研究科*

1 はじめに

今日のインターネット上では 多様なマルチメディア情報を提供する情報ベースが 自然発生的に 点在しており、これら情報ベースから利用者が移動計算機を通して自分の望む情報を自由に検索・加工したい、という要求が高まっている。一方で、各情報ベースは 自然発生的なものであり、統一的な検索インターフェースや利用者の意図に沿ったデータのスキーマ、検索メソッド、SQLのような問い合わせ言語を提供していないことが多い。そこで、利用者からの自由な検索を補助するための仲介者（mediator）機構が注目されている。その必要機能は次の二点である：

1：一般には、ユーザは 情報ベース X が提供するスキーマやメソッドを事前に知らない。そのため、これらを X から 教えてもらう必要がある。

2：次に、上の情報を元に、ユーザの意図に沿った自由な検索を効率良く情報ベース X で実行させる。

本稿では 永続オブジェクト機能を有したスクリプト言語 Persistent Perl を提案し、それを用いて 移動計算機向けの永続オブジェクト管理システムと 上の Mediator 機能を実現する。

2 永続オブジェクトの実現

図1に 本研究で 想定する情報ベース・ネットワークを示した。図中、ユーザは 移動計算機 を通して 未知の情報ベース X から自分の意図に沿った自由な情報検索を行ないたいとする。このとき、本研究では前節の機能1を 次の手法で実現する：

「各情報ベース X は Persistent Perl を用いて 自分が提供する永続オブジェクトの定義をスクリプトとして記述しておく。ユーザからの問い合わせに応じてクラス定義を 電子メイルや HTML などで X から ユーザへ通知する。」

この方式では、各情報ベース X は そのデータ管理方式や記憶構造、検索メソッドは X 固有の管理方式のままで良い。しかし、自分が提供する永続オブジェクトのクラス ClassX の構造や 検索手続きを Persistent Perl でディクショナリ情報として記述しておく。特に X が提供している情報検索の手続きは スクリプトで指定されたメソッドを通して実行できるとする。

*Persistent Object Manager using a Script Language for Mobile Computers
S.T.Wisut, T.Ohmori, M.Hoshi (U.Electro-Comm.)

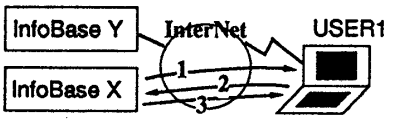

図 1: 情報ベース・ネットワーク

例1：情報ベース X が 永続オブジェクト StudentClass の定義とその１つのインスタンス・オブジェクト例を 電子メールでユーザに転送する場合を考えよう。以下が X から 送られてくる Persistent Perl で書かれたスクリプトである。

```
########## メールの内容 ##########
#class definition
persistent class StudentClass
body
  StudentCode string,
  Name string,
  BirthDate string,
  Sex char,
  Relation refto PersonClass,
method
StudentClass($StudentCode, $Name,
             $BirthDate, $Sex, $Relation)
begin
  $self->{StudentCode} = $StudentCode;
  $self->{Name} = $Name;
  $self->{BirthDate} = $BirthDate;
  $self->{Sex} = $Sex;
  $self->{Relation} = $Relation;
end
getStudent()
begin
  print "My name is $self->{Name}\n";
  print "My Professor is
    $self->{Relation}->{Name}\n";
  print "My Professor's Department is
    $self->{Relation}->{Department}->{Name}\n";
end
endclass
#sample of data object
IS40030!#!Wisut Sae-tung!#!22/01/1966!#!M
!#!PersonClass_2!#!
#end mail
```

Persistent Perl では 各永続オブジェクトには その生成時に [クラス番号_インスタンス番号] 形式の識別子 (*OID*) が与えられ、プログラム中からは連想配列 $*OID*として扱われる。その要素オブジェクトは $*OID* →{ 属性名 } で参照される。上の例では、StudentClass は 属性 Relation だけが 他のオブジェクト PersonClass への *OID* であり、他は原子値である。

移動計算機上のユーザは、 この Persistent Perl を用いて 自分が有する永続オブジェクトのスキーマ、メソッド定義と インスタンス・データベース記憶領域とを管理している。インスタンス・オブジェクトについては、現在は 永続オブジェクトを クラスごとに Unix ファイルとして永続化しており、*OID*が参照される度に *OID*から Perl の連想配列への Swizzling の検査、および（必要なら）ファイルからのロードを実行している。オブジェクトの永続性は 現在 shadowing で実現している。

3 Mediator の実現方式

次に、移動計算機上のユーザから 情報ベースへ 任意の検索を実行させる方法を述べる。先述したように、ユーザ 1 が行ないたい検索手続きやスキーマ化は X に事前に用意されていない。そこで 本研究では ユーザが （X から送られてきたクラス定義を元に） Persistent Perl によって任意の検索スクリプトを記述し、これを電子メイルなどテキスト形式で サーバ X へ送りつけ、X がそのスクリプトを実行する、という方法を採用する。即ち：

・各ユーザに依存した問い合わせ最適化やスキーマ管理機能は移動計算機側の 個人データベースとして 管理し、

・各情報ベースは 自前の検索方法以外には Persistent Perl の処理系だけを用意し、これによってユーザから与えられた任意の検索スクリプトを実行する。

この方針に基づくと、次の手順 1 − 3 で Mediator 機能が実現できる：

手順 1 ： X は ユーザ 1 の要求に応じて 永続オブジェクト ClassX の定義（スキーマとメソッドとインスタンス値）を （電子メイルや HTML などの）テキスト形式で携帯計算機へ 転送する。

手順 2 ：ユーザ 1 は、転送されてきた ClassX の定義を自分の永続オブジェクト管理システムへ登録する。次に、自分の検索意図に応じて ClassX に対し 適切なスキーマ追加や検索を行なうスクリプトを Persistent Perl で記述する。次に、このスクリプトを 電子メイルや HTML などテキスト形式で X へ 送る。

手順 3 ： X は、送られてきたスクリプト・プログラムを実行し、その結果をユーザに送り返す。

例 2 ：既に例 1 で上の手順 1 は実行済みとして、手順 2 からを説明しよう。まずユーザは、例 1 で X から 送られてきた StudentClass の定義を自分の移動計算機内の永続オブジェクトシステムに登録する。次に、StudentClass の定義と自分のクラス定義 (DepartmentClass, PersonClass) を見て、StudentClass への適切な検索メソッドを Persistent Perl で作成する：

```
########## メールの内容 ##########
#class StudenClass
#additional method
sub queryStudentByProf{
  my $self = $_[0];
  my $ProfessorName = $_[1];
  my $result;
  do{
    $_ = $self->{Relation}->{Name};
    if (/^(.*)$ProfessorName(.*)$/){
      $result .=  $self->{StudentCode}.'#!#'.
        $self->{Name}.'#!#'.
        $self->{BirthDate}.'#!#'.$self->{Sex}.'#!#'.
        $self->{Relation}->{PersonCode}.'#!#'."\n";
      }
    $self = &getNextObject(StudentClass);
  }while ($self);
  $result;
}
require 'persistent.pl';        #
&dbOpen('IsInformation');       #
require 'StudentClass.dic';     #
 $firstObj = &getFirstObject('StudentClass');
 $result = &queryStudentByProf($firstObj,'Omori');
open (DAT,'>tmp.txt'); print DAT $result;
close DAT;
```

このスクリプトは StudentClass から 選択条件の元で navigation 操作を行ない、属性 Relation の値を *OID* ではなく PersonClass のキー値へスキーマ変換を要求している。このスクリプトを X へ電子メイルで送ると、X がこれを実行し、その結果 (tmp.txt) を ユーザへ送り返す：

```
    #### 検索結果ファイル tmp.txt の内容 ####
IS40030#!#Wisut Sae-tung#!#22/01/1966#!#M#!#IS001
#!#IS40032#!#Zhang Ji#!#xx/xx/xxxx#!#M#!#IS001#!#
```

最後に、ユーザは X から送られてきた結果を適切な形で自分のデータベースに組み合わせ、移動計算機側でのアプリケーションを実行する。下のスクリプトは送られてきた StudentClass オブジェクトの検索結果をユーザ側で登録・表示する例である：

```
&BeginTransaction;
  < StudentClass の Relation の値を ユーザ側の>
  < PersonClass の OID へ変換して DB へ登録する >;
  $firstoid = &getFirstObject('StudentClass');
  &getStudent($firstoid);
&EndTransaction;

 # 実行結果
My name is Wisut Sae-tung
My Professor is Omori Tadashi
My Professor's Department is System Design
```

4 まとめ

本稿では 永続オブジェクト機構つきのスクリプト言語による Mediator 機構を提案した。提案方式では、ユーザが情報ベースから 集めてくる永続オブジェクト（クラス定義や検索メソッド）は 全て個人データベース（従来の View にあたる）として移動計算機に管理されており、ユーザと共に 情報ベース・ネットワーク内を移動する。そして、ユーザ固有の検索メソッドとして 必要に応じて 移動計算機から 情報ベースへ送られ、そこで実行される。この方式は、特に マルチメディアデータをユーザ固有の情報フィルタプログラムを使って 情報ベース側で フィルタリングする際に有効と考えている。

現在 複合オブジェクトやテキスト検索の範囲での試作を進めている。画像データの扱いや分散情報ベースの渡り歩きが今後の課題である。

参考文献：The TSIMMIS project. 第 100 回情処 DBS 研資料 '94 年 10 月.