2G−7 永続プログラミング言語INADAによる永続オブジェクト集合の操作

有次 正義　寺本 圭一　天野 浩文　牧之内 顕文
(九州大学 工学部 情報工学科)

1. はじめに

永続プログラミング言語INADAは，機能が高く拡張の容易なオブジェクト指向データベース管理システムを記述するための永続オブジェクト指向プログラミング言語である．INADAはオブジェクト指向プログラミング言語C++をベースに，オブジェクトの永続化，永続オブジェクトのマルチタイプ，集合体オブジェクトの検索と一括操作の3つの機能をもつものである．これらの機能の多くがクラス定義の形でモジュール化されているため，応用に適した設計の採用や新たな実装技術の導入による言語処理系のカスタマイズがプログラマによって容易に行なえる点に特色がある．本稿ではこれらの機能を概説し，それらを用いることによって永続オブジェクト集合の持つタイプを変更し，仮想的な集合を定義・操作することが可能であることを示す．これによってオブジェクト指向パラダイムの枠内で関係データベースのビューを実現する一つの手法を提案する．

2. 永続オブジェクト

INADAでは，仮想記憶空間中に2次記憶上のある領域と1対1に対応する空間(Persistent Heap:PH)を考え，その空間上に，いわゆるヒープ領域と同じような記憶領域割当を行なうことによってデータの永続性を実現している．永続性を持たせたいオブジェクトはPH領域に割り当てれば良い．具体的にはINADAを使って図1のように記述すれば良い．

```
persistent <<クラス名>>* <<オブジェクト名>>;
<<オブジェクト名>>
        = new(<<集合体オブジェクト名>>)
                <<コンストラクタ>>;
<<オブジェクト名>> -> <<メンバ名>>;
```

図 1: 永続オブジェクトの生成・操作

3. マルチタイプオブジェクト

永続オブジェクトが複数の応用プログラムによって共有される場合，モデル化されている実世界の実体は同一であっても，それぞれのプログラムから見たそのオブジェクトの役割が異なる場合もある．

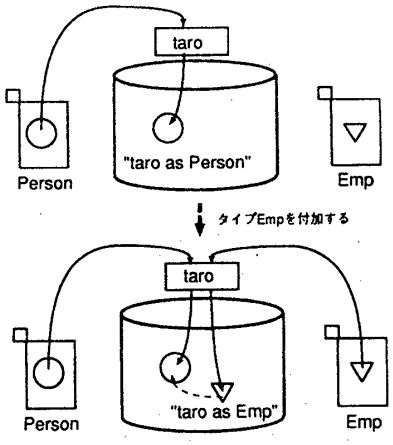

図 2: マルチタイプオブジェクトの模式図

これに対する一つの解決策は，例えばすべての見方に対応できるようなクラス(タイプ)の定義を最初に行ない，以後は一貫してそれに基づくオブジェクトを共有することである．

しかし，これではクラス(タイプ)の設計が非常に困難であるだけでなく，モデル化した実体自身の変化に対応してクラスを変更することも不可能である．永続オブジェクトはそれを作成したプログラムが使用されなくなった後でも保存・使用される可能性があることを考えると，この方法では問題がある．そこでINADAでは，新たに必要となったクラス(タイプ)を必要となった時点で付加する機能を提供し，マルチタイプのオブジェクトを許す．ここでいうクラスはC++のクラスと全く同じものであり，INADAではオブジェクトの同一性を保ちながら複数のクラス(タイプ)を持つことが可能となる．

例えば，クラスPersonの永続オブジェクトtaroを生成し，それにEmpというタイプを付加することができる．永続オブジェクトtaroを操作するときには，クラスPersonとEmpを切替えて，操作する時に適切な役割でアクセスすることが可能である(図2参照)．

さらに，INADAでは永続オブジェクトの存在とそのタイプとは独立である．つまり，「タイプのない」オブジェクトも存在し得る．これによって，まだ性質のはっきりしないオブジェクト同士をそのOIDにより区別することが可能となる．しかも，タイプを持たないオブジェクトの性質は，それが明らかになった時点で記述できる．さらに，その時点にすでに存在する他のオブジェクトからのそのオブジェクトへの参照もまた有効である．

Manipulation of Persistent Object Sets in Persistent Programming Language INADA
Masayoshi Aritsugi, Keiichi Teramoto, Hirofumi Amano, and Akifumi Makinouchi
Department of Computer Science and Communication Engineering,
Kyushu University

4. 集合体オブジェクト

大量の永続オブジェクトを再利用するプログラムは，そのうちの必要なオブジェクトだけを選択的に操作しなければならないことがある．INADAでは，集合を操作するためのクラスを導入する．集合的な性質を持つものをINADAでは総称して集合体と呼ぶ．集合体を応用プログラムの側で作成することを可能とするために，データベースに対する問い合わせとのインターフェイスを考慮した集合体標準インターフェイスを定義する．

集合体標準インターフェイスは，集合体オブジェクトの各要素に対し順次操作を可能とするために次のように定義する．

i) `iterator* OpenScan(iterator* i)`
探索子(iterator)iによって指定された点から集合を操作するための探索子を返す．

ii) `void CloseScan(iterator* i)`
探索子iを消去する．

iii) `Type* GetElement(iterator* i)`
探索子iの指す要素へのポインタを返す．

iv) `iterator* Next(iterator* i)`
探索子iの指す要素の次の要素への探索子を返す．

これらを用いて，Type型の要素の集合体オブジェクトdomain1に対する拡張for文(図3)

```
for all Type* x in domain1
such that <<条件式>>
   do <<文>>
```

図 3: 拡張for文の例

は図4のようなC++プログラムで処理される．

```
iterator* i=domain1;
while(i){
   if( <<条件式>> )
      <<文>>;
   i=domain1->Next(i);
}
```

図 4: 拡張for文のC++での処理

5. ビュー

INADAにおけるオブジェクトのコンテナは前章で説明したC++におけるクラスにより定義される集合体(クラスのインスタンス)であり，コンテナとクラスを分けて考える．このため，OODBに問い合わせを行なって得られるオブジェクトの集合はクラスではなく集合体となる．そこで，OODBにおいて，関係データベースにおける組をインスタンス，関係スキーマをクラス定義，関係をあるクラスの要素の集合と対応づける．

INADAの持つ機能を次のようにして用いることによってビューを実現する．ここでは簡単のために以下の例をもとに考える．

[例 1] クラスEmployee(Name,Age,Salary)のインスタンスを要素とする集合体クラスEmpsetのインスタンスempsetがある．この時，Salaryを隠蔽したビューを作成する．

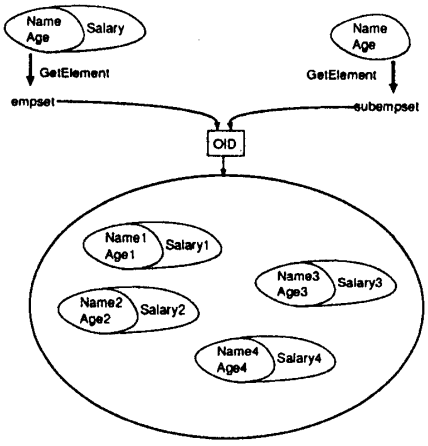

図 5: 射影によるビューの模式図

これはINADAの機能を使って次のように考えればよい．クラスEmployeeのインスタンスempsetにSubEmployee(Name,Age)クラスのインスタンスを要素とする集合のクラスSubEmpsetのタイプ(クラス)を付加する(図5参照)．図5で，オブジェクトempsetに対してメソッドGetElementを起動すると，タイプEmployeeの要素が得られ，同じオブジェクトに対してSubEmpsetのタイプとしてメソッドGetElementを起動すると，タイプSubEmployeeの要素を得ることができる．ここで特に言及すべき点は，実際に存在するオブジェクトを参照する(内部データを持たない)仮想的なタイプを付加することによってビューを提供していることであり，ここではビューを実現するためのオブジェクトを別に作る必要がないことである．これにより，ビューからの操作による変更などもすべて有効に働くことになる．

6. まとめ

本稿では永続プログラミング言語INADAの提供する機能を概説し，それらを用いて仮想的な集合を生成・操作することによってビューを実現できることを示した．

今後は，マルチタイプオブジェクトを実現するより効率の良い機構，それを用いたビューの定義・表現の検討や，インデックス検索を可能とするものなどのさまざまな集合に対する処理についての考察などを行なう予定である．

参考文献

[AA92a] 有次, 天野, 牧之内:“永続プログラミング言語INADAのマルチタイプオブジェクト,” 情報処理学会アドバンストデータベースシステムシンポジウム,pp.93–100, 平4.12.

[AA92b] M.Aritsugi, H.Amano, A.Makinouchi:“Roles, Views, and Virtual Set Attributes in an Object-Oriented Persistent Programming Language,”Tech. Rep. CSCE-92-C14, Dec. 1992.

[AA92c] 天野, 有次, 白, 寺本, 牧之内:“永続プログラミング言語INADAのデータ管理,” 信学技法 DE92-33,pp.15–22, 平4.11.