データ知識協調モデルにおける種々の階層の実現方式*

5F－1

彭智勇　　上林弥彦†

京都大学工学部‡

1　まえがき

データベース応用分野の拡大に伴って、データと知識を共に扱えるような高度のシステムが重要となってきている。しかし、データは同じ属性構造を有するデータ同士がまとめられる傾向にあるのに対し、知識は同じ主題のメソッドがまとめられる傾向があるため、データと知識は別々に保存されることになる。従来のデータベースと知識ベースの統合は応用ごとに固有のものが実現されてきた。しかし、種々の応用がデータや知識を共有するためには、新しい応用が追加されても柔軟に対処できるようなデータベースと知識ベースの統合システムの開発が重要となる。データ知識協調モデル [1] はこのような柔軟性を実現するために有効なモデルである。本稿では、データ知識協調モデルにおけるデータの統合および知識の統合のための種々の階層の実現方式について検討する。データと知識の協調のためのコンテキストモジュール階層の実現は一般のオブジェクト指向方式で行なえる。このため、あらたに代理機構を導入し、これを用いたデータおよび知識の階層の構成を示す。

2　基本事項

データ知識協調モデルに基づくシステムは複数のデータベースと知識ベースのサーバと結合して、クライアント / サーバアーキテクチャを構築することで構成される。データと知識はリンクを通して、データベースと知識ベースから統合システムに移入される。移入されたデータと知識は統合システムの中でデータ知識協調モデルによって統合され、協調して種々の問題を扱う。

本モデルでは、データの統合と知識の統合はそれぞれ各自の階層構造で整理される。これらの二つの階層構造を結び付けるため、コンテキストモジュールと呼ばれる階層がある。各コンテキストモジュールは実際の応用の一つを実現し、必要なデータと知識はデータベースや知識ベースから提供される。応用が異なると、異なるコンテキストモジュールが対応する。また、コンテキストモジュールはオブジェクト指向的な階層を構成することによって、利用状況の変化に対処できる。従って、データ知識協調モデルはこの3種の階層から構成されたオブジェクト指向モデルの一般化といえる。

データ階層: データの統合

知識階層: 知識の統合

コンテキストモジュール階層: データと知識の協調

しかし、データと知識の移入、統合および協調の実現には、従来のビューでは機能的に不充分である。例えば、データと知識のコピーを用いて、実体化ビューを統合システムの中で生成すると、データと知識の重複といった問題が生じる。物理的なコピーを持たずに必要に応じてビューを計算する方法では、計算結果に応用向きの独立な属性やメソッドを追加することができない。また、データと知識の利用権の制御と更新伝達などの機能を充分サポートしていない。我々は**代理機構** (proxy mechanism) という概念を導入し、ビューの持つある種の問題の解決をはかる [2]。この代理機構を用いると、データベースと知識ベースの統合の際に柔軟性および効率が高まることになる。

データと知識はオブジェクト (O) として記述できる。データの記述は属性 (A) を、知識の記述はメソッド (M) を中心にする。クラスは同じ属性構造やメソッドを有するオブジェクト群を集約する。形式的に、クラスは定義1で定義される。ここで、次の記号を用いる。

$\Rightarrow$: 操作の実現の導出

$\uparrow$: 属性値あるいはメソッドの実行の結果の返却

$:=$: 属性値の代入

$\dashv$: メッセージパシング (message passing)

$f_{T\rightarrowtail T'}$: 属性値の型 T から T' への変換関数

$f^{-1}_{T\rightarrowtail T'}$: $f_{T\rightarrowtail T'}$ の逆関数

[定義 1] クラスを $C = \langle O, A, M\rangle$ とする。

(1) $extent(C) = O$ は C のインスタンスの外延であり、$o_i(\in O)$ は C の一つのインスタンスを表す。

(2) $type_a(C) = A$ はオブジェクトの属性の定義の集合であり、$T_j : a_j \in A$ に対して、a_j は属性の名前を、T_j は属性値の型を表す。オブジェクト o_i が属性 a_j の値を実際に持っていれば、この属性値はドット記法を用いて $o_i.a_j$ と表す、それに対する処理は $\{read(o_i, a_j) \Rightarrow\uparrow o_i.a_j, write\ (o_i, a_j, v_j) \Rightarrow o_i.a_j := v_j\}$ として実現される。

(3) $type_m(C) = M$ はオブジェクトのメソッドの定義の集合であり、$m_k \in M$ に対して、m_k はメソッドの名前を表す．メソッド m_k の実行は $o_i \dashv m_k$ の形で表される。

代理クラスは既存のクラスから代理機構によって導出される。この場合には、既存のクラスを導出代理クラスの元のクラスという。代理クラスのインスタンスは元のクラスの対応するインスタンスの代理オブジェクトと呼ぶ。このような基本的な場合の代理クラスとそのインスタンスは次のように定義できる。

[定義 2] クラス $C^s = \langle O^s, A^s_\alpha, M^s_\alpha\rangle$ を元のクラスとする。その代理クラスの定義は $C^p = \langle O^p, A^p_\alpha \cup A^p_\beta, M^p_\alpha \cup M^p_\beta\rangle$ になる。

(1)$extent(C^p) = O^p$ は C^p のインスタンスの外延であり、$O^p = \{o^p_i | o^p_i \rightarrow o^s_i, o^s_i \in extent(C^s)\}$ のように生成される。$o^p_i \rightarrow o^s_i$ は o^p_i が o^s_i の代理オブジェクトとして、o^s_i の識別子を持っていることを表す。

(2)$type_a(C^p) = A^p_\alpha \cup A^p_\beta$ は代理オブジェクトの属性の定義の集合であり、次の二つの種類に分けられている。

A^p_α : 元のオブジェクトから継承された属性の定義の集合である。$T^p_j : a^p_j \in A^p_\alpha$ に対する処理は $\{read(o^p_i, a^p_j) \Rightarrow\uparrow f_{T^s_j\rightarrowtail T^p_j}(read(o^s_i, a^s_j)), write(o^p_i, a^p_j, v^p_j) \Rightarrow write(o^s_i, a^s_j, f^{-1}_{T^s_j\rightarrowtail T^p_j}(v^p_j)), T^s_j : a^s_j \in type_a(C^s)\}$ のように切替えされ、元のオブジェクトに対して実行される。

A^p_β : 代理オブジェクトに追加された応用向きの独立な属性の定義の集合である。$T^p_{j'} : a^p_{j'} \in A^p_\beta$ に対する処理は $\{read(o^p_i, a^p_{j'}) \Rightarrow\uparrow o^p_i.a^p_{j'}, write(o^p_i, a^p_{j'}, v^p_{j'}) \Rightarrow$

---

*Realization of Various Hierarchies for Data-Knowledge Coordination Model Utilizing Proxy Mechanism

†Zhiyong PENG, Yahiko KAMBAYASHI

‡Faculty of Engineering, Kyoto University

$o_i^p.a_{j'}^p := v_{j'}^p\}$ として元のオブジェクトと関係なく、代理オブジェクトの中で実行される。

(3)$type_m(C^p) = M_\alpha^p \cup M_\beta^p$ は代理オブジェクトのメソッドの定義であり、次の二つの種類に分けられている。

M_α^p ：元のオブジェクトから継承されたメソッドの定義の集合である。$m_k^p \in M_\alpha^p$ に対する実行は $\{o_i^p \dashv m_k^p \Rightarrow\uparrow o_i^s \dashv m_k^s, m_k^s \in type_m(C^s)\}$ のように切替えされ、元のオブジェクトに頼んで行なわれる。

M_β^p：代理オブジェクトに追加された応用向きの独立なメソッドの定義の集合であり、$m_{k'}^p \in M_\beta^p$ に対する実行は $o_i^p \dashv m_{k'}^p$ として元のオブジェクトと関係なく、代理オブジェクトの中で行なわれる。

3 データ階層

データの統合は代理機構によって合併階層と結合階層を構成することによって実現できる。

(1) 合併階層

性質が同じデータは複数のデータベースに分散され、別々の元のクラスのインスタンスとして統合システムの中に存在することが可能である。それらの統合は代理機構の合併操作を用いて、一つの代理クラスにまとめることによって実現される。統合のための代理クラスのインスタンスは別々の切替え操作を用いて、別々の元のクラスのインスタンスの代理オブジェクトとして利用される。合併階層の定義は次のようになる。

[**定義 3**] 複数の性質が同じであるデータの元のクラス $C_1^s = \langle O_1^s = \{o_{i_1}^s\}, A_1^s = \{T_{j_1}^s : a_{j_1}^s\}\rangle$,..., $C_n^s = \langle O_n^s = \{o_{i_n}^s\}, A_n^s = \{T_{j_n}^s : a_{j_n}^s\}\rangle$ に対して、それらの共通部分の属性の定義の集合を $A^s = A_1^s \cap, ..., \cap A_n^s$ とする。合併階層はそれらの元のクラスの上で、合併操作 $C^p = \mathbf{Union}(C_1^s, ..., C_n^s)$ を用いて以下のような統合のための代理クラスを作ることによって構成された階層である。

$extent(C^p) = \{o_{i_1}^p | o_{i_1}^p \rightarrow o_{i_1}^s, o_{i_1}^s \in extent(C_1^s)\} \cup ... \cup \{o_{i_n}^p | o_{i_n}^p \rightarrow o_{i_n}^s, o_{i_n}^s \in extent(C_n^s)\}, type_a(C^p) = \{T_j^p : a_j^p | read(o_{i_1}^p, a_j^p) \Rightarrow\uparrow f_{T_j^s \mapsto T_j^p}(read(o_{i_1}^s, a_j^s)), write(o_{i_1}^p, a_j^p, v_j^p) \Rightarrow write(o_{i_1}^s, a_j^s, f^{-1}_{T_j^s \mapsto T_j^p}(v_j^p)), ..., read(o_{i_n}^p, a_j^p) \Rightarrow\uparrow f_{T_j^s \mapsto T_j^p}(read(o_{i_n}^s, a_j^s)), write(o_{i_n}^p, a_j^p, v_j^p) \Rightarrow write(o_{i_n}^s, a_j^s, f^{-1}_{T_j^s \mapsto T_j^p}(v_j^p)), T_j^s : a_j^s \in A^s\}$.

(2) 結合階層

データは要素として複数のデータベースで蓄えられている場合には、これらのデータ要素の統合は代理機構の結合操作を用いて、統合のための代理クラスを作ることによって実現される。そのクラスのインスタンスは結合条件 cp を満たすデータ要素を組み合わせて、データ要素の代理オブジェクトとして利用される。代理オブジェクトは代理クラスに定義された切替え操作を用いて、すべてのデータ要素の属性を持っているように見える。結合階層の定義は次のようである。

[**定義 4**] 複数のデータ要素の元のクラスを $C_1^s = \langle O_1^s = \{o_{i_1}^s\}, A_1^s = \{T_{j_1}^s : a_{j_1}^s\}\rangle$,..., $C_n^s = \langle O_n^s = \{o_{i_n}^s\}, A_n^s = \{T_{j_n}^s : a_{j_n}^s\}\rangle$ とする。結合階層はそれらの元のクラスの上で、結合操作 $\mathbf{Join}(C_1^s, ..., C_n^s, cp)$ を用いて、以下のような統合のための代理クラスを作ることによって構成された階層である。

$extent(C^p) = \{o_i^p | o_i^p \rightarrow o_{i_1}^s \times ... \times o_{i_n}^s, cp(o_{i_1}^s, ..., o_{i_n}^s) = true, o_{i_1}^s \in extent(C_1^s), ..., o_{i_n}^s \in extent(C_n^s)\}, type_a(C^p) = \{T_{j_1}^p : a_{j_1}^p | read(o_i^p, a_{j_1}^p) \Rightarrow\uparrow f_{T_{j_1}^s \mapsto T_{j_1}^p}(read(o_{i_1}^s, a_{j_1}^s)), write(o_i^p, a_{j_1}^p, v_{j_1}^p) \Rightarrow write(o_{i_1}^s, a_{j_1}^s, f^{-1}_{T_{j_1}^s \mapsto T_{j_1}^p}(v_{j_1}^p)), T_{j_1}^s : a_{j_1}^s \in type_a(C_1^s)\} \cup ... \cup \{T_{j_n}^p : a_{j_n}^p | read(o_i^p, a_{j_n}^p) \Rightarrow\uparrow f_{T_{j_n}^s \mapsto T_{j_n}^p}(read(o_{i_n}^s, a_{j_n}^s)), write(o_i^p, a_{j_n}^p, v_{j_n}^p) \Rightarrow write(o_{i_n}^s, a_{j_n}^s, f^{-1}_{T_{j_n}^s \mapsto T_{j_n}^p}(v_{j_n}^p)), T_{j_n}^s : a_{j_n}^s \in type_a(C_n^s)\}$.

4 知識階層

知識の統合は代理機構によって汎化階層と抽象化階層を構成することによって実現できる。

(1) 汎化階層

知識ベースから移入された一般の知識に対して適用できるメソッドは応用向きの特殊な知識にも適用できる。すなわち、特殊な知識は一般の知識のメソッドの一部を受け続ぐと共に、それ独自のメソッドも持つことになる。これらの統合は代理機構の拡張操作を用いて、一般の知識の元のクラスの代理クラスを作り、さらにそれ独自の特殊な知識のメソッドを追加することによって実現される。知識の汎化階層の定義は次のようになる。

[**定義 5**] 一般の知識の元のクラスを $C^s = \langle O^s, M_\alpha^s\rangle$ とする。M_β^p は応用向きの特殊な知識のメソッドの定義の集合である。汎化階層は一般の知識の元のクラスから拡張操作 $C^p = \mathbf{Extend}(C^s, M_\beta^p)$ を用いて、特殊な知識のメソッドが追加された以下のような代理クラスを作ることによって構成された階層である。

$extent(C^p) = \{o_i^p | o_i^p \rightarrow o_i^s, o_i^s \in extent(C^s)\}, type_m(C^p) = \{m_k^p | o_i^p \dashv m_k^p \Rightarrow\uparrow o_i^s \dashv m_k^s, m_k^s \in type_m(C^s)\} \cup \{m_{k'}^p | o_i^p \dashv m_{k'}^p, m_{k'}^p \in M_\beta^p\}$

(2) 抽象化階層

知識ベースからの抽象的な知識は応用向きの具体的な知識より更に広い範囲に適用できる。具体的な知識のメソッドの実行に失敗したら、抽象的な知識の対応するメソッドを実行させることによって、具体的な知識が適用できないメソッドには抽象的な知識のメソッドを応用する。そのために、具体的な知識のメソッドから抽象的な知識の対応するメソッドを計算する関数 f_a^m が必要である。具体的な知識と抽象的な知識の統合は、代理機構の拡張操作を用いて、抽象的な知識の元のクラスの代理クラスを作り、さらに具体的な知識のメソッドを関数 f_a^m に基づいて追加することによって実現される。知識の抽象化階層の定義は次のようになる。

[**定義 6**] 抽象的な知識の元のクラスを $C^s = \langle O^s, M_\alpha^s\rangle$ とする。M_β^p は応用向きの具体的な知識のメソッドの定義の集合であり、また、関数 f_a^m に対して、$\forall m_{k'}^p (m_{k'}^p \in M_\beta^p) \rightarrow \exists m_{k'}^s (m_{k'}^s = f_a^m(m_{k'}^p) \wedge m_{k'}^s \in M_\alpha^s)$ が成立する。抽象化階層は一般の知識の元のクラスから拡張操作 $C^p = \mathbf{Extend}(C^s, M_\beta^p \rightarrow f_a^m(M_\beta^p))$ を用いて、具体的な知識のメソッドが追加された以下のような代理クラスを作ることによって構成された階層である。

$extent(C^p) = \{o_i^p | o_i^p \rightarrow o_i^s, o_i^s \in extent(C^s)\}, type_m(C^p) = \{m_k^p | o_i^p \dashv m_k^p \Rightarrow\uparrow o_i^s \dashv m_k^s, m_k^s \in type_m(C^s)\} \cup \{m_{k'}^p | \mathbf{ifFailure}(o_i^p \dashv m_{k'}^p) \Rightarrow\uparrow o_i^s \dashv f_a^m(m_{k'}^p), m_{k'}^p \in M_\beta^p\}$

5 あとがき

本稿では、従来のビューよりも機能の高い代理機構を用いて、データ知識協調モデルにおける種々の階層の実現方式について述べた。我々は Smalltalk を用いて、この代理機構の実現を行なっている。

参考文献

[1] Qiming Chen,Yahiko Kambayashi, Coordination of Data and Knowledge Base Systems under Distributed Environment, IFIP, DS-5 Semantics of Interoperable Database Systems. Lorne, Victoria, Australia, November 16-20 1992.

[2] 彭智勇, 上林弥彦: データ知識協調モデルにおける代理クラスとその応用, 電子情報通信学会技術報告, DE93-46, pp. 1-8 (1993-11).