データ知識協調モデルにおけるデータ知識スキーマの変換方式*

7C－3

彭智勇　上林弥彦†

京都大学工学部‡

1　まえがき

データ知識協調モデルは分散したデータベースと知識ベースの統合のために導入された[1]。このモデルは、Smalltalkシステムをデータベースと知識ベースの間に介して、データ・知識を独立なデータ・知識ベースからSmalltalkシステムに動的に移入して(import)協調させる形で実現できる[2]。実現する場合に、移入された異種のデータ知識のスキーマは応用に応じて柔軟に変換できることが要求されている。しかし、サブクラス、ビューおよびバージョンによる従来の変換方式は、データ知識スキーマの変換には不十分である。この問題に対して、我々は代理クラスの概念をオブジェクト指向モデルに導入した。代理クラスは元のクラスの変形として作られる。元のクラスのインスタンスを代理クラスのインスタンスにすると、その属性と操作を代理クラスの定義に従って変更させることになる。また、代理クラスに対しては、許された範囲内での自律性を認めており、これがビューと異なる点といえる。本稿では、代理クラスによるデータ知識スキーマの変換方式について報告する。

2　データ知識協調モデルとその実現法

データベースや知識ベースの応用分野の拡大によって、データベースと知識ベースの協調が非常に重要となっている。従来のデータベースと知識ベースの協調は応用ごとに固定的に行なう。しかし、実際の応用は色々あり、あらかじめ予測できないものもありうる。応用が異なると用いられるデータと知識およびそれらの組合せが異なる。従って、データ知識協調システムは、独立かつ不均一なデータベースや知識ベースから、応用に応じたデータ知識を繋ぎ合わせて使えるようになるべきである。

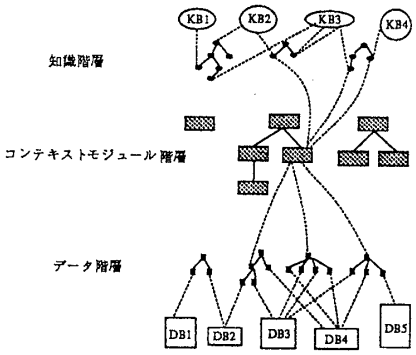

図 1: データ知識協調モデル

この場合に巨大な統合システムを作るのは処理効率や維持管理面から適切ではなく、個々の独立性を持たせたまま動的に統合してゆくのが望ましい。しかし、データ間、知識間およびデータ・知識間に自由にリンクを作れるような手法は自由度が高すぎて、スキーマとしてあまり有効とは言えない。そこで、図1に示すようなデータ知識協調モデルが提案された[1]。

このデータ知識協調モデルでは、データ間の結合はデータ階層の形で扱い、知識間の結合は知識階層の形で扱う。データは同時に利用されるデータどうしがまとめられる傾向にあるのに対し、知識は同じ主題のものがまとめられる傾向であるため、この二つの階層は一般に一致しない。データと知識間の対応を完全に自由にせずにここに一つの階層を設けることにする。これをコンテキストモジュールの階層と呼ぶ。コンテキストモジュールは単なる容器で対応するデータおよび知識はデータベースと知識ベースから供給される。この対応はリンクを中心に実現される。

我々はSmalltalkシステムを中心に実現する方法を提案した。この方法では、Smalltalkシステムは独立なデータベースと知識ベースの間にあり協調環境を実現する。すなわち、データベースと知識ベースのそれぞれに対し、Smalltalkシステムとのリンクを定義する。データと知識はリンクによってSmalltalkシステムにオブジェクトの形で移入される。Smalltalk言語によりデータ階層、知識階層およびデータ知識協調のコンテキストモジュール階層を作る。種々の問題はこの3種の階層の協調により扱うことになる。

3　データの移入とその記述

データは実体の構造と状態を表し、その属性によって分類される。種々のデータベースが利用者の立場から次のように抽象的に定義できる。

[定義1] データベースをDB=$\{T\}$とする。Tは$\langle D, A, Q\rangle$で、属性が同じであるデータの集合を表す。

D:データ識別子の集合{d}であり、dをデータ識別子という。

A: データの属性の集合$\{T_a : a\}$であり、T_aを属性値のデータ型、aを属性の名前という。$T_a : a \in A$, $d \in D$に対して、次のような関数がある。

(1)$read_a(d)$: データdの属性aからデータ型がT_aである値を読み出す。

(2)$write_a(d, v)$: データdの属性aにデータ型がT_aである値vを書き込む。

Q: データ型に関する問い合わせ関数であり、ある条件に満たすデータを問い合わせる。

データの移入は上記の定義に基づいて行なう。データの基本属性値はSmalltalkシステムの中の基本クラスに直接に対応する。例えば、データベース中の整数がSmalltalkシステムに移入されて、Smalltalk言語の整数表現になる。一般のデータ型に対して、一つのデータクラスを作る。データはデータクラスのインスタンスとして移入される。このようなオブジェクトはデータの代理(proxy)として定義され、実際の属性値を持たなくてデータの識別子だけを参照している($o \rightarrow d$)。その属性値を処理するために、属性ごとにデータクラスにインスタンスメソッドとして二つのメソッド$(a, a :)$を定義する。aは 関数$read_a(d)$を呼び出し($o\ a \Rightarrow read_a(d)$), 属性$a$の値を読み出す。$a : v$は関数$write_a(d, v)$ を呼び出し($o\ a : v \Rightarrow write_a(d, v)$), 属性$a$に新しい値$v$を書き込む。データクラスに定義されたクラスメソッドはデータの移入に使う。このよ

*Transformation of Data and Knowledge Schemata for Data-Knowledge Coordination Model

†Zhiyong PENG, Yahiko KAMBAYASHI

‡Faculty of Engineering, Kyoto University

うなメソッドはデータ型に関する問い合わせ関数を引き出し、条件を満たすデータに対して、オブジェクトを一つずつ作る。さらに、データ識別子を対応するオブジェクトのインスタンス変数に保存するようにデータがオブジェクトと結合する。すなわち、データ移入の記述は次のようになる。

[定義 2] データベース DB=$\{T\}$ におけるデータ型 $T = \{\{d_1,...,d_m\},\{a_1,...,a_n\}\}$ に対して,移入されたデータのクラスは $DC = \{\{o_1 \rightarrow d_1,...,o_m \rightarrow d_m\},\{(o_x\ a_1 \Rightarrow read_{a_1}(d_x), o_x\ a_1 : v_{a_1} \Rightarrow write_{a_1}(d_x, v_{a_1})),...,(o_x\ a_n \Rightarrow read_{a_n}(d_x), o_x\ a_n : v_{a_n} \Rightarrow write_{a_n}(d_x, v_{a_n})), x = 1,...,m\}\}$ になる。

4 知識の移入とその記述

知識は実体間の推論関係を表し、推論操作の主題によってまとめられる。種々の知識ベースが利用者の立場から次のように抽象的に定義できる。

[定義 3] 知識ベースを KB=$\{S\}$ とする。S は $\langle F, R, I\rangle$ で、一つの知識主題に関する推論操作を表す。

F: 事実データの集合 $\{f\}$ であり、f が事実データを表す。f に対して、知識ベースの利用プロセス p は次の関数を利用できる。
(1)$read_f(p)$:p は事実データを f から読み出す。
(2)$write_f(p, factData)$:p は事実データ $factData$ を f に書き込む。

R: 推論のためのルールの集合という。

I: 推論のための関数の集合 $\{i\}$ であり、i を推論関数という。

知識の移入は上記の定義に基づいて行なう。つまり、知識主題に対して、Smalltalk システムの中で、知識クラスを作る。知識クラスのインスタンス k を生成すると、一つの知識ベースの利用プロセス p を発火させることになる。k は p を参照して $(k \rightarrow p)$、知識ベースのインタフェースとして利用される。知識ベースの事実データのための関数は知識クラスのメソッドの形で呼び出す $(k\ f \Rightarrow read_f(p), k\ f : factData \Rightarrow write_f(p, factData))$。知識オブジェクトはそれらのメソッドを使って、知識ベースの中の事実データを読み出したり、書き込んだりする。推論関数もメソッドによって呼び出される $(k\ \ m \Rightarrow i(p))$。このようなメソッドは知識ベースの中のルールに基づく推論を起動する。知識ベースからの知識は次のように利用される。すなわち、知識オブジェクトが受けたメッセージは知識クラスに対応するメソッドを呼び出す。そのメソッドは必要な事実データを知識ベースに提供して、知識ベースでは得られた事実データを用い、その中のルールに基づいて推論することになる。最終結果は知識の利用者に戻される。従って、知識移入の記述は次のようになる。

[定義 4] 知識ベース KB=$\{S\}$ における知識主題 $S = \{\{f_1,...,f_q\},\{i_1,...,i_r\}\}$ に対して,移入された知識のクラスは $KC = \{\{k_1 \rightarrow p_1,...,k_p \rightarrow p_p\},\{(k_x\ f_1 \Rightarrow read_{f_1}(p_x), k_x\ f_1 : factData_{f_1} \Rightarrow write_{f_1}(p_x, factData_{f_1})),...,(k_x\ f_q \Rightarrow read_{f_q}(p_x), k_x\ f_q : factData_{f_q} \Rightarrow write_{f_q}(p_x, factData_{f_q}))\},\{k_x\ m_1 \Rightarrow i_1(p_x),...,k_x\ m_r \Rightarrow i_r(p_x)\}, x = 1,...,p\}$ になる。

5 データ知識キーマの変換方式

5.1 従来の方式の問題点

上記のように移入されたデータと知識は,データクラスと知識クラスによって記述されている。これらのデータ知識スキーマを多種多様の応用に応じて変換させる必要がある。しかし、従来の方式は次の理由でデータ知識スキーマの変換には不十分である。

(1) **サブクラス**はスーパークラスから一般の属性と操作を継承した上で自分の特殊な属性と操作を追加することができるが、この継承機構では、継承された属性と操作の削除および変更を直接には行なえず、複雑な変換を必要とする。

(2) ビューは問い合わせにより一定の条件に満たすデータ知識を特定の形でユーザに見せる。ビューに新しい属性を追加するなどのビュー更新は困難な問題である。

(3) **バージョン**方式はデータ知識スキーマをコピーして、応用向きの形に変換させるものである。これはデータ知識の重複といった問題が生じる。

5.2 代理クラスの導入とそれによる変換方式

上記の問題を解決するために、代理クラスの概念をオブジェクト指向モデルに導入する。代理クラス C^p は元のクラス C^o から次のように導出される。

[定義 5] 元のクラス $C^o = \{\{o_1^o,...,o_u^o\},\{(a_1^o, a_1^o :),...,(a_v^o, a_v^o :)\},\{m_1^o,...,m_w^o\}\}$ に対して, その代理クラスは $C^p = \{\{o_1^p \rightarrow o_1^o,...,o_u^p \rightarrow o_u^o\},\{(o_x^p\ a_1^p \Rightarrow o_1^o\ a_1^o, o_x^p\ a_1^p : arg_{a_1} \Rightarrow o_x^o\ a_1^o : arg_{a_1}),...,(o_x^p\ a_v^p \Rightarrow o_x^o\ a_v^o, o_x^p\ a_v^p : arg_{a_v} \Rightarrow o_x^p\ a_v^p : arg_{a_v})\},\{o_x^p\ m_1^p \Rightarrow o_x^o\ m_1^o,...,o_x^p\ m_w^p \Rightarrow o_x^o\ m_w^o\}, x = 1,...,u\}$ になる。

C^p は C^o に依存し、そのインスタンス $o_x^p, x = 1,...,u$ は必ず一つの C^o のインスタンス o_x^o を参照する。これは $o_x^p \rightarrow o_x^o$ で表す。C^o の属性 $a_y(a_y^o, a_y^o :), y = 1,...,v$ に対して、C_p にはその処理メソッド $(a_y^p, a_y^p :)$ が定義される。メソッド a_y^p は o_x^p に作用すると、参照された o_x^o がメソッド a_y^o を引き出して、その属性 a_y の値を戻すことになる。同様な手順で、属性 a_y の値がメソッド $a_y^p : arg_{a_y}$ によって更新される。この二つのメソッドを利用して、o_x^p は o_x^o の属性 a_y を持っているように見える。これらを $(o_x^p\ a_y^p \Rightarrow o_x^o\ a_y^o, o_x^p\ a_y^p : arg_{a_y} \Rightarrow o_x^o\ a_y^o : arg_{a_y})$ で表す。その他、C^o のメソッド $m_z^o, z = 1,...,w$ は C^p のインスタンスが m_z^p の形で利用できる。つまり、メッセージ m_z^p は o_x^p に送って、起動されたメソッド m_z^p はこのメッセージを m_z^o の形で o_x^p に転送して、C^o のメソッド m_z^o が起動されるようになる。この表現が $o_x^p\ m_z^p \Rightarrow o_x^o\ m_z^o$ である。

このようにして、元のクラスのインスタンスは代理クラスのインスタンスとして利用される。すなわち、移入されたデータ知識に対して、その代理が作られる。代理クラスは好きな形で自由に元のクラスの属性と操作を継承、変更、削除することが可能である。そのインスタンスはビューと異なり、元のオブジェクトと同時に働くので、応用に応じて必要となる属性と操作を追加することが容易になる。しかも、これは一般のオブジェクトと同じように参照、再利用される。その他、元のクラスの属性の値と操作の実現は代理クラスに重複していないため、バージョンの問題も解決できる。従って、利用者はデータ知識クラスの代理クラスを作り、利用状況に対応して再定義することによって、データ知識スキーマを応用に応じて柔軟に変換させることができる。

6 あとがき

本稿で提案したデータ知識スキーマの変換方式が異種かつ独立なデータベースと知識ベースからのデータと知識の統合に有効である。今後、有効性に対する検討を行なう予定である。

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のオブジェクト指向方式による実現法, 情報処理学会研究会資料、93-DBS-93 と 93-OS-59(1993).