明示的な意味表現を持つデータベースにおける更新処理

3V－4

片山 紀生　高須 淳宏　安達 淳
東京大学大学院　学術情報センター研究開発部

1 はじめに

我々は、明示的な意味表現を持つデータベースとして「意象(いしょう)データベース」を提案している[1]。ここで意味表現とは、個々のデータが持っている意味を表現するための機構のことであり、データベースシステムが明示的にしろ暗黙的にしろ必ず持っている機構である。関係データベースやオブジェクト指向データベースでは、リレーションやオブジェクトといったデータ構造に意味表現としての役割を持たせ、データ構造のどの位置にデータを格納するかによってデータの意味を表現する。これに対して意象データベースでは、意味表現は物理的なデータ構造から分離され、意味ネットワークのようなネットワーク構造として明示的に管理される。

そのような明示的な意味表現を持つデータベースにおいて更新処理を行う場合、特に問題となるのが意味表現の一貫性制御である。意味表現がネットワークという自由度の高い構造を持っているため、意味表現の無矛盾性を維持するために一貫性制御機構が不可欠になるのである。そこで意象データベースでは、演繹データベースで用いられている層状プログラム[2]を拡張した述語論理を用いて一貫性を管理することを提案している[3]。

本稿では、意象データベースにおける更新処理について説明する。意味表現がネットワークという単純な構造を持っていること、また、強力な記述能力を持つ述語論理に基づいていることから、意象データベースにおける更新処理は次のような特徴を持っている。

- 更新処理におけるプリミティブが明確かつ単純であり、それらのセマンティクスを容易に定義できる。
- プリミティブのセマンティクスが明確であるため、どのような場合に一貫性を損なうのか容易に判定することができる。
- 述語論理の枠組で一貫性制御機構を定式化できる。

2 意象データベースの構成

図1に意象データベースの構成を示す。数値、文字列、テキストといった物理データは、抽象データ型によってカプセル化され、それらのデータのことを物象(P-data: Physical Data)と呼ぶ。意味表現はネットワーク構造を持ち、ノードのことを意象(S-object: Semantic Object)、リンクのことを意接(S-link: Semantic Link)と呼ぶ。意

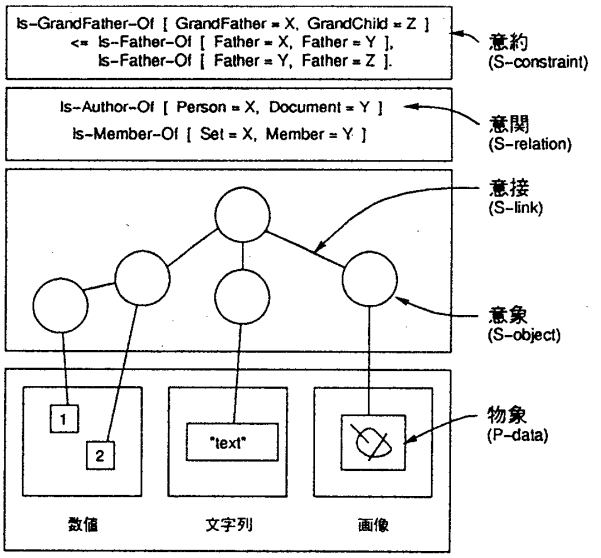

図1: 意象データベースの構成

関(S-relation: Semantic Relation)は同じ意味的関連を表現する意接を集めたものであり、述語を用いて

```
Is-Author-Of [ Person = X, Document = Y ]
Is-Member-Of [ Set = X, Member = Y ]
```

などと表現される。意約(S-constraint: Semantic Constraint)は意味表現に対する制約であり、層状プログラムに基づいて記述する。

図2にプロトタイプシステムの構成を示す。プロトタイプシステムは階層構造を持っており、データベース管理システム、意味モジュール、アプリケーションプログラムの三層に大きく分かれる。データベース管理システムはさらに細かい階層を持っており、下から順に、意象、意接、意関、参照、導出、一貫性、物象、文字列型、ディレクトリ、モジュールの管理を行う。これらは意象データベースとして必要不可欠な構成要素であり、意象データベースの根幹を実現する。これに対して意味モジュールは、データベース管理システムの機能を拡張するためのものであり、利用者がデータ型や意関を定義することによってアプリケーションプログラムの基盤を提供する。例えばデータ型について見てみると、データベース管理システムでは文字列型しか用意されていない。従って、数値型などのデータ型を利用するためには、それらを定義した意味モジュールを用意する必要がある。

3 更新処理のプリミティブ

まず、意象データベースの構成要素で更新処理の対象となり得るのは、意象、意接、意関、部分意関、物象の五

Update Operations for a Database System with an Explicit Semantic Representation

Norio Katayama[1], Atsuhiro Takasu[2], Jun Adachi[2]
[1]University of Tokyo
[2]National Center for Science Information Systems

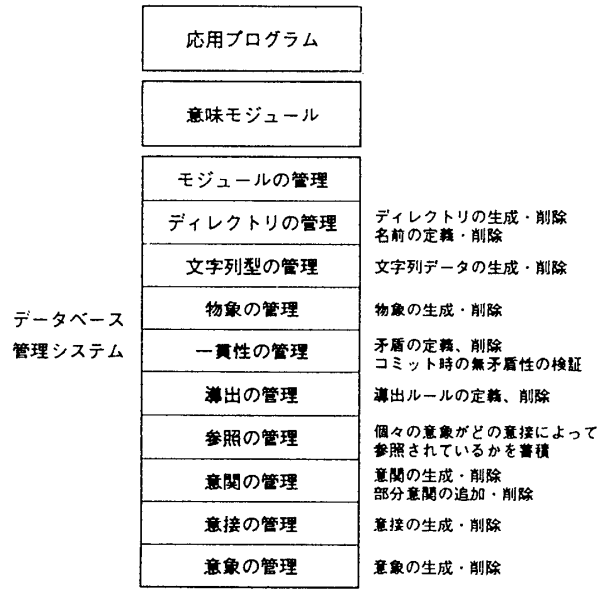

図 2: プロトタイプシステムの構成

つである。部分意関とは、意関を水平分割したものであり、ひとつの導出ルールがひとつの部分意関に相当する。例えば、下の Is-GrandFather-Of 意関の場合、二つの部分意関から構成されている。

```
Is-GrandFather-Of [ GrandFather = X, GrandChild = Z ]
     ← Is-Father-Of [ Father = X, Child = Y ],
       Is-Father-Of [ Father = Y, Child = Z ].
Is-GrandFather-Of [ GrandFather = X, GrandChild = Z ]
     ← Is-Father-Of [ Father = X, Child = Y ],
       Is-Mother-Of [ Mother = Y, Child = Z ].
```

したがって導出ルールの追加は、部分意関の追加として処理される。また、ディレクトリは意関のひとつとして実現されるので、ディレクトリの生成・削除は意関の生成・削除として、ディレクトリにおける名前の定義・削除は意接の生成・削除として処理される。よって、更新処理のプリミティブは以下の十種類になる。

- 意象の生成・削除
- 意接の生成・削除
- 意関の生成・削除
- 部分意関の追加・削除
- 物象の生成・削除

4 一貫性制御

十種類の更新プリミティブのうち一貫性を損なう可能性のあるものは、以下の三種類である。また、制御しなければならない一貫性は、意接による参照、意味表現としての無矛盾性、意接の削除の三種類である。

- 意象の削除: その意象を参照している意接が全て削除されなければならない。
- 意接の生成: 意味表現としての無矛盾性を損なう可能性がある。
- 意接の削除: 意味表現としての無矛盾性を損なう可能性がある。また、導出ルールのために削除できないことがある。

4.1 意接による参照

意象を削除するためには、その意象を参照している意接が全て削除されなければならない。また、逆にどの意接からも参照されていない意象があれば、その意象は削除することができる。この一貫性を維持するためには、個々の意象がどの意接によって参照されているのかという情報を蓄積しなければならない。これが、図 2 における「参照の管理」の層の役割である。

4.2 意味表現としての無矛盾性

意象データベースでは意味表現としての矛盾はルールによって記述する。例えば、Is-Father-Of 意関の Father 項の値が必ず Male 意関にも属していなければならないことを定義するには下のように記述する。

```
← Is-Father-Of [ Father = X, Child = Y ],
  ~Male [ Person = X ].
```

4.3 意接の削除

意接を削除しようとしても導出ルールによって削除できない場合がある。例えば左記の Is-GrandFather-Of に関する導出と下の二つの意接が定義されていたとする。

```
Is-Father-Of [ Father = "為義", Child = "頼朝" ]
Is-Father-Of [ Father = "頼朝", Child = "実朝" ]
```

このとき、以下の意接を削除しようとしても、導出ルールのために削除することができない。

```
Is-GrandFather-Of [ GrandFather = "為義",
                    GrandChild = "実朝" ]
```

この場合、定義されている二つの意接のうち少なくともひとつを削除する必要があるが、どれを削除するかは利用者の判断を仰ぐしかない。そのためデータベース管理システムはこのような場合を検出し、利用者の再考を求める必要がある。

5 おわりに

明示的な意味表現を持つデータベースにおける更新処理について検討し、そのプリミティブおよび一貫性を維持する手法について説明した。今後は、並行処理制御の実現、アプリケーションプログラムの実装などを行ない、明示的な意味表現を持つデータベースの有効性を検証していく考えである。

参考文献

[1] Katayama, N., Takasu, A., and Adachi, J., "A Database with an Explicit Semantic Representation," *Proc. of the 7th Int. Conf. Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, (1994), 323–332.

[2] Minker, J. (Ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann Publishers (1988).

[3] 片山, 高須, 安達, 「データベースにおける意味表現とその述語論理による管理」, 情報処理学会第 48 回 (平成 6 年前期) 全国大会 5F-4.