

映像データベースにおける 検索加工表示システムのWWW上での実装

春日 史朗

有澤 博

横浜国立大学 工学部 電子情報工学科
〒240 横浜市保土ヶ谷区常盤台 79-5

本研究室では、柔軟で高いモデリング能力を持つ関数型データモデル AIS を提案してきた。しかし、その実装においては、マルチメディアデータベースを構築する際に必要となるメディアの拡張手法が確立していなかった。

そこで本研究では、メディアの拡張性に優れたデータベースシステムの構築手法を提案し、クライアント-サーバモデルを用いた実装を行なった。さらに、その拡張性を利用し、World Wide Web 上における映像データベースの検索加工表示システムを構築し、その有効性を示した。

Implementation of Retrieving, Processing and Visualizing System for Video Image Database in World Wide Web

Shiroh Kasuga

Hiroshi Arisawa

Division of Electrical and Computer Engineering
Faculty of Engineering

Yokohama National University 79-5, Tokiwadai, Hodogaya-ku, Yokohama-shi, Japan

We have presented the functional data model "AIS" which are flexible and has expressive power in data modeling. But, as for implementation of AIS model data base, we had not established a technique of media extension to construct a multimedia database.

For that reason, in this paper, we are proposing a system architecture of a database system which are superior in media extension, and we implement it following to a client-server model. Additionally, making use of it's augmentability, we are approving an efficiency of this system to implement a retrieving, processing and visualizing system for video image database on World Wide Web.

1 はじめに

データベースで扱う情報は、従来の文字、数値などに加えて、画像、映像、音声などさまざまな形態を持つマルチメディア情報へとその範囲を広げている。

このようなマルチメディアデータベースでは、利用される目的に従って、さまざまな独自のメディアが追加される。

現在、リレーショナルデータベースなどの旧来のデータベースに対して画像、音声などの新しい

メディアを追加したシステムが存在するが、メディアに依存した処理を行なうために、データベースシステムに対して大規模な変更が必要である。

このような問題に対し、オブジェクト指向データモデルは一つの解を与えるものである。しかし、このデータモデルでは、オブジェクトというデータ単位の内部にデータ構造を持たせているために、データの中立性が失われ、データベース利用者にとってのデータの見え方(ビュー)が固定される、という問題点があげられる。

そこで我々は、データの中立性を実現するデータモデルである、AIS モデル [1][2] を提案してきた。

本研究では、データモデルとしてこの AIS モデルを採用し、さまざまな形態のメディアが混在するデータベースを構築する際に、容易にメディアや、その固有のデータ操作を追加することが可能なデータベースシステムの構築手法を提案する。また、提案する構築手法をクライアント-サーバモデルを用いて実装し、その拡張性を示すために World Wide Web 上で検索を行なうことができる映像データベースを実際に構築した。

2 AIS モデル

ここでは、実装したデータベースシステムで採用したデータモデルである AIS モデルについてその概要を述べる。

2.1 AIS モデル

AIS モデルは関数型データモデルの一種であり、現実世界を表現するのに、主体(entity)と二項関係(association)という概念を用いている。(図 1)

主体は現実世界の「もの」を表す。一つの主体は内部構造を持たない「点」として表され、他の主体とは内部的な識別子(entity ID)で明確に区別される。また、文字、数字、文字、ビットマップなどの値データも主体であり、値主体(value entity)と呼ばれる。全ての主体は ER モデルのように主体型(entity type)によってまとめられ、主体型名によって主体の表す意味を区別する。

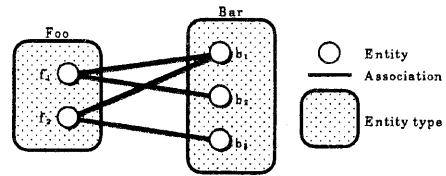


図 1: 主体(entity)と二項関係(association)

二項関係は主体と主体の間の意味的なつながりを表す。この二項関係には 1 対 1 と 1 対 n があり (n は任意数)、1 対 n の場合はつながり先の主体間に順序付けがなされている。この順序付けは、動画のフレームを順序付ける場合等に利用できる。

二項関係を用い、任意の主体型から特定のパスを通してある主体型への関数を定義することが可能となる。この関数をデータベース関数(database function)と呼ぶ。

データベースへの検索は、ある主体型に属する主体集合と、その主体型に定義されたデータベース関数を提示することにより、データベース関数を適用した結果である主体集合を得ることである。

図 1 における例として、主体 f_1, f_2 に対して、主体型 Foo から主体型 Bar へのつながりを表すデータベース関数 $FooBar$ を適応すると、 $FooBar(f_1) = \{b_1, b_2\}$, $FooBar(f_2) = \{b_2, b_3\}$ が得られる。また、データベース関数 $FooBar$ の逆関数 $BarFoo$ が同時に定義され、 $BarFoo(b_1) = \{f_1, f_2\}$, $BarFoo(b_2) = \{f_1, f_2\}$, $BarFoo(b_3) = \{f_2\}$ が得られる。

2.2 オブジェクト式

データベースアプリケーションにとっては、AIS モデルで表現された単純な情報より、内部に階層構造を持つ情報の方が取り扱いに優れている。

この要求に応えるため、データベースアプリケーションに対する検索結果として、アプリケーションが独自に定義した形式の、内部に階層構造を持つオブジェクトを出力する方式を採用している。

図 2 は階層構造をもつオブジェクトの例である。オブジェクトは sun を表し、その惑星(planet)として mercury, venus, earth, mars を含んでいる。さ

らに earth と mars は衛星 (satellite) として moon, phobos, deimos を含み, それぞれが名前 (name) と画像 (image) を含んでいる。

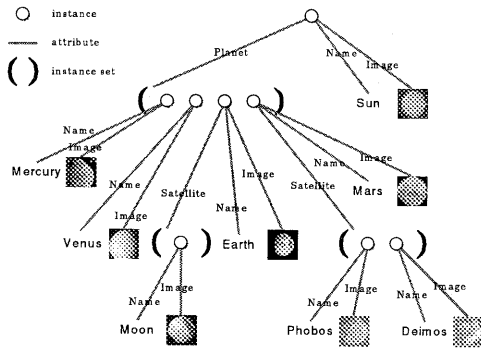


図 2: 階層構造をもつ情報の例

図 2において, 白丸で示されているものは, 値を持たない事物の存在を表す. それに対して, 値を持つ点の場合には値自身が埋め込まれている. また, “(” と “)” により値の列が表現される.

“値” と “値の列” を結ぶ線分を, 属性 (attribute) と呼ぶ. 属性には属性を識別するために名前が付けられる. これを属性名 (attribute name) と呼ぶ. 属性はある主体に対して下層の主体 (又は主体列) が属性名で示される意味において所属することを表す.

図 2の階層構造における特定の階層のことをランク (rank) と呼ぶ. ここで主体に対する操作はランクを単位として行なうとし, 主体を個々に指定する操作は行なわないことにする. これにより, データベースシステムでは多くの処理で, 列中の全てのオブジェクトで同時並列処理を行なうことが可能となる. そのために, OE 中で列を構成する全てのオブジェクトの構造は等しいものとする. 従って, この共通な構造が定義され, これを基本形 (cardinality) と呼ぶ. 図 2において mercury と venus では, 属性 satellite が存在しないように見えるが, 空の主体列が存在していると考えられる.

ところで, このような階層構造をもった情報を, 伝統的に使用されてきた記述法と似た形式で得るために, 図 2を式 (expression) で表したものが, 図

3に示すオブジェクト式 (Object Expression:OE) である.

```
[ NAME: "sun",
  IMAGE: [ ]
]
PLANET {
  [ NAME: "mercury",
    IMAGE: [ ]
  ],
  [ NAME: "venus",
    IMAGE: [ ]
  ],
  [ NAME: "earth",
    IMAGE: [ ]
  ],
  SATELLITE: [ NAME: "moon",
    IMAGE: [ ]
  ],
  [ NAME: "mars",
    IMAGE: [ ]
  ],
  SATELLITE: [
    [ NAME: "phobos",
      IMAGE: [ ]
    ],
    [ NAME: "deimos",
      IMAGE: [ ]
    ]
  ]
}
```

図 3: オブジェクト式の例

3 OE の計算機上での実現

ここで, 計算機の記憶装置上においてデータベースシステムが扱うデータの構造である OE を改めて定義する.

3.1 AIS モデルによるデータベースシステム

AIS モデルを用いて構築したデータベースシステムは, データベースに保存されている情報と, データベースアプリケーションに渡される情報の構造が大きく異なるため, AIS モデルにより表現された平坦な情報から, データベースアプリケーションが要求する, 内部に階層構造を持つ情報に変換する機構が必要である.

本論文ではこのような処理を行なう機構をデータベース検索エンジン (database retrieving engine) と呼ぶ. (図 4)

データベース検索エンジンにとって, OE は検索処理の処理対象であり, 検索結果の表現でもある. 従って, OE の効率的な実装は AIS モデルによるデータベースにおいてはキーとなる技術である.

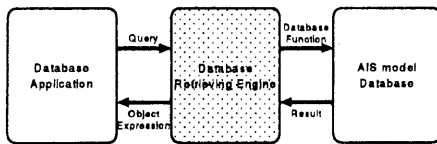


図 4: データベースシステムの構成

3.2 OE の内部表現

OE は図 2 で示される階層構造をそのまま計算機の主記憶上に構築することで、計算機の扱う情報として表現することができる。(図 2 において主体間の線分をポインタとみなす)

しかし、そのような構造では同じ型の主体間(図 3 における“mercury”と“venus”という文字列や水星と金星の画像)の属性情報を統括的に管理する構造がないため、型に関する情報を主体毎に持つことが必要になる。また、主体間に張られるポインタが膨大な量になるため、構造の記憶装置間の移動(主記憶⇄補助記憶, 主記憶⇄主記憶)でポインタを付け換える操作が必要となる。

そこで、OE を表現する際に、それらの問題を解決する構造である階層リスト構造と、その要素である OE 区分リストを提案し、その構造について述べる。

3.3 OE 区分リスト

先に述べたように、AIS モデルでは、個々の主体に対して操作を行なうことはなく、操作単位は必ずランクである。従って、主体を個々に検索する機構は必要でない。

そこで、同じ型の主体を可能な限り統括して管理するため、OE 中に存在する多数のオブジェクトについて、同一ランク中の主体をまとめてリストを構成する。(図 5)

この時、別々のオブジェクトの主体であったことを表すために、リストには区分(partition)が設けられる。

この構造を OE 区分リスト (OE Partitioned List:OEPL) と呼ぶ。

図 5 において、OEPL は区分を“(”と“)”で囲

`string ( ) ( "moon" ) ( "phobos" "deimos" ) ( "io" "europa" "ganymede" "callisto" )`

図 5: OE 区分リストの例

んで表現している。一つの区分には 0 以上の任意個数の主体を含む。

OEPL を構成する主体は全て共通の型であり、一つの OEPL は一つの型を持つ。

3.4 階層リスト構造

OE の持つ情報とは、単純化すると図 6 に示した 1 対 n の対応関係の集合である。

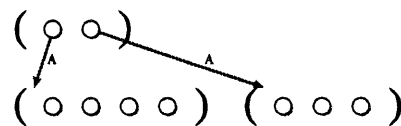


図 6: OE の表現する情報

この対応関係は直接的なポインタを用いることなく、図 7 に示すように計算機の記憶装置上に構成することが可能である。

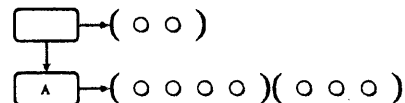


図 7: HAL による表現

このような構造を用いることにより、図 3 に示されるような OE は OEPL の階層構造により表現することが可能である。(図 9)

ここで、このような構造を階層リスト (Hierarchical Associated List:HAL) 構造と呼ぶ。

OEPL は HAL 構造により、上下層に位置する別の OEPL と対応関係を持つ。通常、ある任意の階層の OEPL の区分数は、上位の OEPL の主体数と等しく、下位のそれぞれの区分と上位の主体が対応関係となる。図 8 では上位の OEPL の主体と、下位の OEPL の区分の対応関係が矢印で示されている。

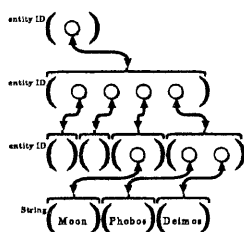


図 8: 上下の階層との対応関係

HAL 構造には以下のような特徴がある。

- 階層構造中の各階層間にポインタによる指示がない
- リストには個々の要素に対する特別な検索構造を持たない
- リスト毎に記憶領域の圧縮が可能である
- ポインタがないことにより他のメモリ空間への転送が容易である

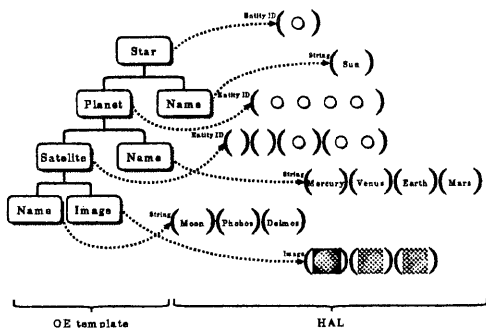


図 9: OE テンプレートと HAL 構造により表現された OE

3.5 OE テンプレート

HAL 構造を用いて OE の主体を表現した場合、OEPL の上下関係を決めるために木構造が用いられる。

この構造は OE の基本形に相当するもので、OE テンプレート (OE template) と呼ばれる。(図 9)

OE テンプレートは同時に、データベースアプリケーションがどのように OE を生成させるかを

定める OE の設計図であり、データベースアプリケーションがデータベースに対して出す検索要求 (query) には、OE テンプレートを用いることができる。

3.6 OE テンプレートの拡張

データベースの検索処理においては、単純なデータベース関数による主体の付加だけでなく、既に付加されている主体の間で演算をする必要な場合がある。

データベースアプリケーションのそのような処理要求は、OE テンプレートに演算子を埋め込むことで、データベース検索エンジンに処理させることができる。

例として、 $(\sin x + 1.00) \times 2.00$ という式は図 10 のような階層構造として表現される。

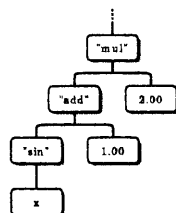


図 10: 式の展開

データベース検索エンジンは、木構造に展開された式を、末端から処理し、最終的に式全体を解決することができる。

3.7 解決した OE

以上から OE とは、基本形である OE テンプレートと主体の列である OEPL の総称である。

ここで、OE テンプレートに必要な主体が全て OEPL として含まれており、かつ式を含まない OE を“解決した OE(resolved OE)”と呼び、それ以外の OE を“未解決の OE(unresolved OE)”と呼ぶ。

データベース検索エンジンの行なう処理とは、OE テンプレートのみで構成された未解決の OE に対して OEPL を付加し、解決された OE にすることである。

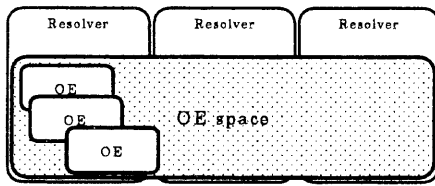


図 11: データベース検索エンジンの構築手法

4 データベース検索エンジンの構築手法

ここでは、本研究で提案するデータベース検索エンジンの構築手法を示す。

4.1 構築手法

実装したシステムでは、データベース検索エンジンとして図 11に示す構築手法を用いている。

モデルを構成するのは以下の3つの要素である。

OE：先に定義した OE

OE 空間 (OE space)：解決した OE、未解決の OE を問わずシステムに存在する全ての OE が存在する空間

リゾルバ (resolver)：OE に対する基本的な操作機能

4.1.1 リゾルバ

システム中にリゾルバは任意数存在することができ、全てのリゾルバが一つの OE 空間を共有する。

個々のリゾルバは、それぞれ特徴付けられた機能を持っており、OE 空間に存在する OE 全てを独自に調査することが可能であり、調査の結果、処理することが可能な部分を発見した場合は、OE に対してそれぞれの特徴的な処理を加える。

一つの OE に対し、複数のリゾルバが、様々な処理を加えてゆくことにより初期状態で OE テンプレートのみであった OE は、最終的に解決した OE となる。

リゾルバの持つ機能の種類としては下記のものが挙げられる。

- データベースに問い合わせを行なうことにより OE に OEPL を付加する
- OE に埋め込まれた演算子进行处理する
- 外部の問い合わせを OE テンプレートに変換し OE 空間に投入する
- 解決した OE を視覚化する

4.1.2 モデルの有効性

この構築手法には下記の利点が挙げられる。

- リゾルバの追加により他の部分に影響を与えずに機能を拡張することができる
- リゾルバはある時点で複数存在する処理対象から最も処理に適したものを独自に判断することが可能である
- まったく同じ処理を行なうリゾルバを同時に存在させることにより処理速度性能を向上させることが可能である

5 データベース検索エンジンの実装の一手法

本研究では、先に提案した構築手法の有効性を示すために、この構築手法に従ったデータベース検索エンジンの実装を行なった。

ここでは、先に示した構築手法に従った実装の一手法の詳細を示す。

5.1 システム構成

この構築手法により実装する上での問題点は、OE 空間を複数のプロセスで共有することである。

今回の実装では、サーバクライアントモデルを用いることで、OE 空間の共有を実現している。(図 12)

システムの中核となるのは OE サーバ (OE server) であり、クライアントとして複数のリゾルバが存在している。

5.2 OE サーバ

OE サーバはその内部に OE 空間を持つプロセスである。

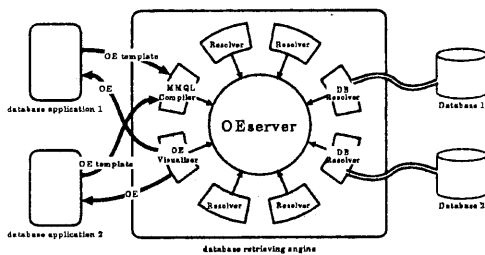


図 12: データベース検索エンジンのシステム構成

OE サーバはシステム中にただ一つ存在しており、クライアントプロセスからの要求に従って OE を公開する。

5.3 リゾルバ

構築手法におけるリゾルバにあたる機構は、実装されたシステムにおいてはクライアントとして存在する。以降このクライアントを示す場合もリゾルバと呼ぶことにする。

リゾルバは OE に対して基本的な処理を行なうプロセス群である。

OE 空間の共有はプロセス間通信により実現されている。リゾルバが OE に対して処理する場合は、プロセス間通信を用いて、OE サーバから OE を受け取り、OE の変更後、OE サーバに OE を書き戻すことで行なわれる。

構築手法で示したリゾルバの基本的な機能のそれぞれに対して、下記のリゾルバを用意した。

データベースリゾルバ HAL 構造において、上層の OEPL が存在する時、AIS モデルデータベースから情報を検索し、その下層に OEPL を構成して付加する。

メディア依存処理リゾルバ OE に埋め込まれている演算子を解釈し、メディア毎の依存処理を行なう。メディアの拡張は新しいメディアに対応したメディア依存処理リゾルバを付加することにより行なう。

検索言語コンパイラ 人が直接データベースにアクセスする場合に、OE テンプレートを構築する

ための言語である MMQL(Multimedia Query Language) が用意されている [1][2]。検索言語コンパイラは MMQL から OE テンプレートを生成する。

OE 可視化ソフトウェア 人が直接データベースにアクセスする場合に、OE の可視化処理を行なう。メディアによって図 3 のような可視化方法から、HTML 化、MPEG 映像化などの可視化法がある。

リゾルバは同時に複数動作しているため、OE サーバには同時に 2 つ以上のリゾルバからの同時アクセスが起こる。そのため、OE サーバは OE のロック機構を持ち、一つの OE に 2 重書き込みが起こるのを防いでいる。

リゾルバは他のリゾルバの動作中でも、起動、停止が可能であるため、動作中のデータベースシステムに対して、リゾルバの付加、排除が可能である。

このような構成により、メディア依存の処理をリゾルバとして追加できるため、データベースシステムに対してメディアを容易に追加することが可能である。

5.4 OE 空間の共有の効率化

OE サーバとリゾルバ間では、OE がやり取りされる。

ここで、多数の OEPL が付随する OE は、非常に大きな構造であるので、プロセス間の転送には高いコストがかかる。

しかし、あるリゾルバが OE に対して処理を行なう場合に、OE 中の全ての OEPL が必要となることは稀である。ほとんどの場合は、ある特定のランクの主体だけが必要であり、OE 全体を OE サーバより受け取る必要がない。

このシステムでは、主体はランク毎に OEPL としてリスト化して存在するため、プロセス間の転送に優れ、また、必要な OEPL のみを転送することで、転送コストを下げる事が可能である。

