データ型に基づくマルチメディアデータ参照機構の実現とその有効性

原田正則[†] 宝珍輝尚[††]
中田充[†††] 都司達夫[††]

本論文では，マルチメディアデータの一部分を独立したデータとして参照可能とするマルチメディアデータ参照機構の実現とその有効性について述べる．本機構は，流動的データ型と呼ぶ抽象データ型の枠組みの下で実現する．流動的データ型では，マルチメディアデータの一部分を表すデータ（指示エントリ）を用い，指示エントリを生成する手続き（導出手続き）と指示エントリをもとにマルチメディアデータの一部分を導出する関数（対応関数）を利用する．これらはユーザによる定義が可能である．本機構の実現においては，流動的データ型の定義において，対応するデータ型の作成と流動的データ型に関する定義情報の管理を行わなければならない．さらに，データフォーマット変換も考慮しなければならない．本論文では，この機構をオブジェクト指向ソフトウェア開発方法論 OMT により設計を行い，商用のデータベース管理システムを用いて実現する．なお，プログラミング言語として C++を採用する．さらに，本機構をいくつかの事例に適用し，小規模のプログラムによりマルチメディアデータの一部分の参照が可能になること，マルチメディアデータの一部分の参照においてオーバヘッドがほとんどないこと，ならびに，導出手続きと対応関数の区別がプログラム作成のガイドラインになっておりプログラミングを容易にしていることを明らかにする．

Implementation of a Multimedia Data Reference Mechanism Based on a Data Type and Its Effectiveness

Masanori Harada,[†] Teruhisa Hochin,[††] Mitsuru Nakata[†††] and Tatsuo Tsuji[††]

This paper describes a multimedia data reference mechanism that makes it possible to manipulate a part of multimedia data as a separate data. This mechanism is implemented by using the abstract data type called a liquid data type. In the liquid data type, users make use of deriving procedures and mapping functions. A deriving procedure creates pointing entries. A mapping function derives a part of multimedia data through a pointing entry by using it as data representing a part of multimedia data. These procedures and functions are defined by users. In implementing this mechanism, it has to be considered that defining a new liquid data type causes the system to create a corresponding data type, and to manage defined information on the liquid data type. Moreover, data format conversion between original multimedia data and a virtual segment has to be considered. A prototype system is designed by using the object-oriented software development methodology OMT. It is implemented on a commercial database management system in C++. Applying it to several applications clarifies several advantages. First, users can manipulate a part of multimedia data by coding small scale program. Second, referring a part of multimedia data has little performance overhead. Third and last is that separation of a deriving procedure and a mapping function results in a programming guideline and can make programming easy.

† 福井大学大学院工学研究科情報工学専攻
Department of Information Science, Graduate School of Engineering, Fukui University
†† 福井大学工学部情報工学科
Department of Information Science, Faculty of Engineering, Fukui University
††† 福井大学大学院工学研究科システム設計工学専攻
Department of System Design Engineering, Graduate School of Engineering, Fukui University

1. はじめに

近年，テキスト，イメージ，グラフィックス，ビデオなどのマルチメディアデータが計算機で手軽に扱えるようになってきている．ここで，マルチメディアデータの一部分を取り出して操作したいという要求がある．たとえば，ハイパメディアシステムにおいて，自動車の写真データの部分にアンカーを付け，その部分をマ

ウスでクリックすることによりタイヤに関する詳細情報を表示するといった場面や，ユーザに応じてレベルを変えたマニュアルを呈示するといった場面で必要とされている．しかし，現状では，応用ソフトウェアがそれぞれでマルチメディアデータの一部分を取り出す機構を作成しており，すべての応用分野において汎用的に利用できる機構としては提供されていない．これには，2つの大きな問題がある．1つは，応用ソフトウェアの開発コストや維持管理コストが高いことである．すなわち，必要とされる機能を作り込まなければならず，拡張機能などをすべて応用ソフトウェア側で実装する必要があるため，開発や維持管理に要する工数が大きくなる．汎用的に利用できる上記のような機構があれば，応用ソフトウェアのこれらのコストを軽減できる．2つ目は，汎用性が考慮されていないということである．すなわち，個々の応用ソフトウェアで必要な機能のみを作成しており，データ型や排他制御といった汎用的な機構としては提供されていない．これらの問題は，データを管理する汎用システムであるデータベース管理システム（DBMS）にとっては非常に大きな問題であり，DBMSにおいて，マルチメディアデータの一部分を取り出し，独立したデータとして参照する汎用的な機構が必要とされている．

このような機構として，流動的データ型[1)~3)]や埋め込みオブジェクト[4)]が提案されている．流動的データ型では，マルチメディアデータの一部分を表す一般に不連続な領域から連続なバイト列（仮想セグメント）を作成する．ユーザは仮想セグメントを通常のデータとして操作することができる．これにより，たとえば，顔の画像中の目の部分のみ取り出して扱うことが可能となる．仮想セグメントを実現するために，指示エントリと呼ばれるユーザ定義の構造体を導入する．指示エントリはもととなるマルチメディアデータと仮想セグメント間の対応を保持する．埋め込みオブジェクトでも同様の機構がオブジェクト指向の枠組みの中で提案されている[4)]．これらは，DBMSでサポートすべき汎用の枠組みと考えられる．しかし，これらの枠組みの実現についての報告はなく，有効性は確認されていない．

そこで，本論文では，画像データを含むマルチメディアデータの一部分の参照を少ない開発負荷で実現可能とすることを目的として，値指向のデータベースにも適用できる流動的データ型の考え方に基づいたマルチメディアデータ参照機構（以降，MMRMとする）の実現法を提案し，その有効性を定量的に評価する．MMRMの実現においては，流動的データ型の定義において，対応するデータ型の作成と流動的データ型に関する定義情報の管理を行わなければならず，データフォーマット変換を考慮しなければならない．MMRMのプロトタイプを商用のオブジェクト指向データベース管理システムUniSQL[5)]を用いてC++により実装する．そして，実現した機構を実際の事例に適用し，小規模のプログラムによりマルチメディアデータの一部分の参照が可能になること，マルチメディアデータの一部分の参照においてMMRM使用のオーバヘッドがほとんどないこと，ならびに，導出手続きと対応関数の区別がプログラム作成のガイドラインとなっておりプログラミングを容易にしていることを明らかにする．

以下，2章では，流動的データ型の概要について述べる．3章では，流動的データ型によるMMRMの実現について述べる．ここでは，主に，実現したクラスについて説明する．4章では，例を用いて，MMRMの使用法について説明する．5章では，MMRMの応用例を示す．6章では，MMRMをプログラム記述量，ならびに，実行時性能の観点から評価する．最後に7章で，まとめを行い，今後の課題を示す．

2. 流動的データ型

流動的データ型は，バイト列で表現されているマルチメディアデータの不連続領域の操作を可能とする抽象データ型である．マルチメディアデータの不連続領域のバイト列を仮想セグメントと呼び，バイト列全体を実データと呼ぶ．仮想セグメントから実データの一部分への写像はユーザ定義の指示エントリと呼ばれるデータ構造体を用いて表現される．指示エントリと実データの関係を**図1**に示す．図1に示した茶碗のイメージデータが実データであり，四角形で囲まれた領域が仮想セグメントである．指示エントリの作成は，ユーザ定義の手続きである導出手続きにより行う．すなわち，ユーザは，実データから指示エントリを作成する手続きを作成し，導出手続きとして登録してお

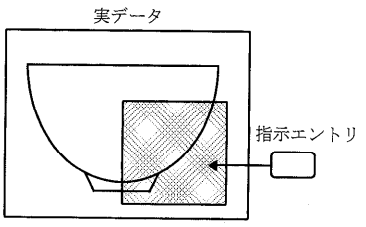


図1 実データと指示エントリの関係
Fig. 1 Relationship between real data and pointing entry.

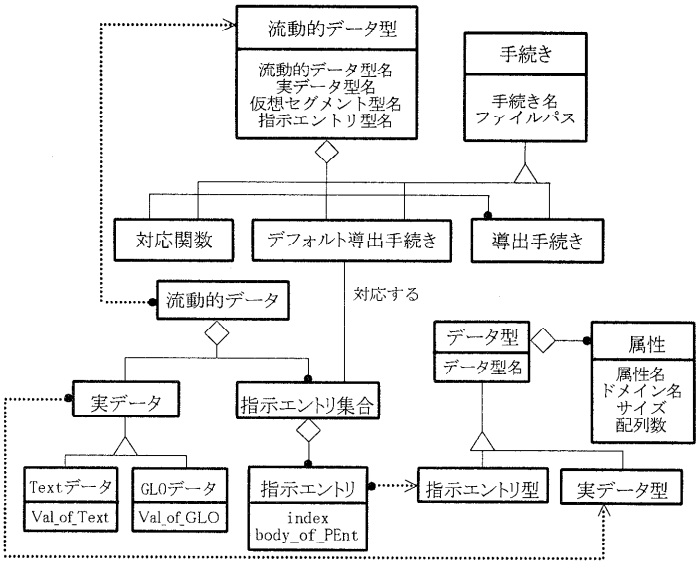


図 2 オブジェクト図
Fig. 2 Object diagram.

く必要がある．仮想セグメントの導出もユーザ定義の関数である対応関数により行う．したがって，ユーザは，実データと指示エントリから仮想セグメントを導出する関数を作成し，対応関数として登録しておく必要がある．また，実データと仮想セグメントのデータフォーマットは同一とは限らないので，データフォーマット変換を考慮する必要がある．

流動的データ型は，フォーマルには，データフォーマット，ならびに，手続きの集合から構成される抽象データ型である．データフォーマットの集合には，実データ，仮想セグメント，指示エントリに対するデータフォーマットが含まれる．また，手続きの集合には，対応関数，導出手続き，データフォーマット変換関数が含まれる．

3. マルチメディアデータ参照機構

ここでは，MMRM のプロトタイプシステムの実現について述べる．プロトタイプシステムを迅速に構築するために，データの二次記憶への格納に商用のオブジェクト指向データベース管理システム UniSQL☆を用いる．また，HP9000/730（HP_UX9.0.1）上で C++ を用いて実装する．設計には，オブジェクト指向方法論 OMT[6]を用いる．

以降では，MMRM の設計と実現方法について述べ，その後，MMRM の特徴であるデータフォーマット変換について述べる．

3.1 MMRM の設計

OMT におけるオブジェクト図を図 **2** に示す．オブジェクト図に示されている黒丸，菱形，三角形は，クラス間の関連を表す記号であり，それぞれ，多重度，集約，特化を表す．多重度は，あるクラスの 1 個のインスタンスが，関連するクラスの何個のインスタンスと関係するかを示す．集約は，あるインスタンスが他のインスタンスから構成されるということを表す．特化は，あるクラスとそれを特殊化したクラスとの関係である．特殊化のもとになるクラスをスーパークラス，特殊化された結果得られるクラスをサブクラスと呼ぶ．

MMRM の実現においては，流動的データ型において，対応するデータの作成と流動的データ型に関する定義情報の管理を行わなければならない．

「流動的データ」クラスは，実データを管理する「実データ」クラスのインスタンスと指示エントリの順序集合を管理する「指示エントリ集合」クラスのインスタンスから構成されている．

「指示エントリ集合」クラスのインスタンスは 1 つ以上の「指示エントリ」クラスのインスタンスを保持する．このとき，使用される「指示エントリ」クラスは，流動的データ型定義における指示エントリ型名より決定される．「指示エントリ」クラスは，「流動的データ」クラスと同様に，ユーザが指示エントリデータ型を定義するごとに作成される．これらにより，MMRM の実現が可能になる．

☆ UniSQL は UniSQL Inc. の登録商標である．

「流動的データ型」クラスはユーザが定義した流動的データ型を管理するクラスである．主な属性として，流動的データ型名，実データ型名，仮想セグメント型名，指示エントリ型名がある．「流動的データ型」クラスのインスタンスは，後述する「対応関数」クラス，「導出手続き」クラス，「デフォルト導出手続き」クラスのインスタンスから構成される．

「手続き」クラスはユーザが指定した手続きや関数を管理するクラスである．ここでは，手続きや関数の名前，ならびに，手続きや関数のオブジェクトファイルへのパスにより管理する．

「対応関数」クラス，「導出手続き」クラス，「デフォルト導出手続き」クラスは，各々，対応関数，導出手続き，デフォルト導出手続きを管理するクラスであり，「手続き」クラスのサブクラスである．「デフォルト導出手続き」クラスについて説明する．MMRM では，導出手続きの指定は，指示エントリ作成時に，ユーザによって行われる．このとき，ユーザが導出手続きを指定しない場合は，流動的データ型定義時にユーザが指定した導出手続きをデフォルトとして使用する．MMRM では，流動的データ型定義時にユーザにより指定された導出手続きをデフォルト導出手続きとして「デフォルト導出手続き」クラスで管理し，指示エントリ作成時にユーザにより指定された導出手続きを「導出手続き」クラスで管理する．

「実データ」クラスは実データを管理するクラスである．「GLO データ」クラスと「Text データ」クラスについては 3.2.1 項で説明する．

「データ型」クラスは様々なデータ型に関する情報を管理するクラスである．「データ型」クラスとデータの属性に対する情報を管理する「属性」クラスの間に一対多の関連がある．「データ型」クラスは属性としてデータ型名を持つ．「属性」クラスは属性として属性名，ドメイン名，データサイズ，配列数を持つ．

「実データ型」クラス，「指示エントリ型」クラスは，各々，実データのデータ型に関する情報，指示エントリのデータ型に関する情報を管理するクラスであり，「データ型」クラスのサブクラスである．

3.2 MMRM の実現

実現にあたり，データの格納のための UniSQL クラスとデータ操作のための C++クラスを作成している．これらについて説明する．

3.2.1 UniSQL クラス

図 2 に示した 15 個のクラスのうち「流動的データ」クラス，「指示エントリ集合」クラス，「指示エントリ」クラス，「GLO データ」クラス，「Text データ」クラスを除く 10 個のクラスは，3.1 節で述べたとおりに UniSQL クラスとして定義している．以降では，「流動的データ」クラス，「指示エントリ集合」クラス，「指示エントリ」クラス，「GLO データ」クラス，「Text データ」クラスについて述べる．

「流動的データ」クラスは，個々の流動的データ型のインスタンスを管理するクラスである．したがって，個々の流動的データ型ごとに対応するクラスが必要である．また，3.1 節で述べたように，「流動的データ」クラスのインスタンスは，「指示エントリ集合」クラスのインスタンスと「実データ」クラスのインスタンスから構成される．そこで，MMRM では，ユーザが流動的データ型を定義するたびに，それに対応する「流動的データ」クラス，「指示エントリ集合」クラスを UniSQL クラスとして作成する．たとえば，ユーザが指示エントリ型名が “Text_Segment” の流動的データ型 “Test” を定義するとする．このとき，MMRM は，「指示エントリ集合」クラスとして UniSQL クラス “seq_of_Text_Segment” を作成し，「流動的データ」クラスとして UniSQL クラス “Test” を作成する．

「指示エントリ」クラスは，前述のように導出手続きにより生成された指示エントリを管理するクラスであり，MMRM では，ユーザが指示エントリのデータ型を定義するたびに，それに対応するクラスを UniSQL クラスとして作成する．

「GLO データ」クラスは，MMRM が対象とする画像データ，音声データなどのマルチメディアデータを管理するクラスである．UniSQL では，マルチメディアサポートのための基本的な構築ブロックとして汎用大規模オブジェクト（General Large Object: GLO）クラスが提供されており，ユーザ定義クラスの属性を GLO クラスとして定義することにより，異なる型のどのような非構造的なデータも保存できるようになる．MMRM では，GLO クラスを利用し，MMRM が対象とするマルチメディアデータが保存されている GLO クラスのインスタンスを「GLO データ」クラスの属性 Val_of_GLO に保持させ，マルチメディアデータの管理を行う．データベースに格納されているマルチメディアデータの読み出しは GLO クラスのインスタンスメソッドを用いて行われる．

「Text データ」クラスは，MMRM が対象とする文字情報を管理するクラスである．「Text データ」クラスの属性 Val_of_Text にはユーザが提供した可変長文字列が保持される．

3.2.2 C++クラス

図 2 のオブジェクト図で示した 15 個のオブジェク

トクラスをC++クラスとして実装している．データの操作はC++クラスのメンバ関数によって行う．また，ユーザが定義したデータ型に対応するUniSQLクラスの作成，データの格納もC++クラスによって行う．ここでは，中心的な役割を持つ「流動的データ型」クラス,「流動的データ」クラスについて説明する．

「流動的データ型」クラスは流動的データ型を管理するクラスであり，利用者が流動的データ型を定義するたびに，対応するUniSQLクラスを作成する．HP_UX9.0.1では関数へのダイナミックリンクをサポートしていないため，対応関数を実行するために，ユーザが定義した対応関数名とそのオブジェクトファイルへのファイルパスに基づき，対応関数のコンパイルを行う．また，ユーザが定義した実データ型名，仮想セグメント型名をもとに，データフォーマット変換方法を決定する．データフォーマット変換については3.3節で詳述する．

指示エントリの作成，ならびに，仮想セグメントの導出は「流動的データ」クラスのメンバ関数より行う．前述のように，HP_UX9.0.1では，関数へのダイナミックリンクをサポートしていないために，MMRMでは，指示エントリ作成時に，ユーザが指定した導出手続き名とそのオブジェクトファイルへのファイルパスに基づき，導出手続きのコンパイルを行い，導出手続きを実行する．MMRMでは，データベースに格納されているマルチメディアデータより指示エントリを作成することを前提としているため,「実データ」クラスにはマルチメディアデータが格納されているUniSQLクラスのインスタンスが保持される．ユーザが指示エントリ作成を要求した場合,「流動的データ」クラスのメンバ関数は,「実データ」クラスに格納されているUniSQLクラスのインスタンスより実データを取り出し，導出手続きを適用し，指示エントリを作成する．導出手続きで作成された指示エントリは対応するUniSQLクラスに格納される.「指示エントリ」クラスは，様々な指示エントリのデータに対応できるようにUniSQLクラスのインスタンスを保持する．ユーザが仮想セグメント導出を要求した場合も同様に,「実データ」クラスに格納されているUniSQLクラスのインスタンスより実データを取り出し，対応関数を適用し，仮想セグメントを導出する．このとき，データフォーマット変換が必要な場合は，流動的データ型定義時に決定したデータフォーマット変換関数により仮想セグメントのデータフォーマットを変換する．

3.3 データフォーマット変換

データフォーマットの変換は，ユーザが流動的データ型で定義した実データ型名，仮想セグメント型名よりデータフォーマット変換表を用いて決定する．データフォーマット変換表の例を**表1**に示す．たとえば，実データ型名が“JPEG”，仮想セグメント型名が“GIF”の場合，データフォーマット変換関数はconvertとなる．ここで，convertは，フリーソフトウェアImageMagick[8]のコマンドである．この例のように，画像データのデータフォーマット変換はconvertを使用し一時ファイルを作成して行っている．テキストデータのデータフォーマット変換は，システム内に用意した関数を用いて行う．

表1 データフォーマット変換表
Table 1 Data format conversion table.

実データ型名	仮想セグメント型名	変換関数
JPEG	GIF	convert
PostScript	GIF	convert
⋮	⋮	⋮

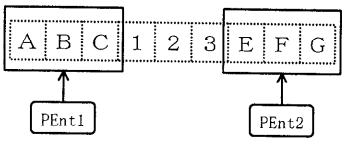


図3 実データの例
Fig. 3 An example of real data.

4. マルチメディアデータの参照

ここでは，MMRMの使用方法を例を用いて説明する．

ユーザは**図3**のテキストデータ“ABC123EFG”から文字列“ABC”と文字列“EFG”を参照したいものとする．このために，まず，流動的データ型を定義する必要がある．この定義の後に，指示エントリを作成する．これにより，文字列“ABC”と文字列“EFG”が仮想セグメントとして参照可能となる．ここで，ユーザは指示エントリを以下のように定義しているものとする．

```
text_segment{
        int offset;      // オフセット
        int length;      // 文字列の長さ
};
```

(1) 流動的データ型の定義

ユーザは仮想セグメントを導出するために，以下のような流動的データ型Testを定義する．

流動的データ型名 ： Test
実データ型名 ： SJIS

仮想セグメント型名 : JIS
指示エントリ型名 : text_segment
導出手続き名 : derived_data
ファイルパス : /users/st/derived_data.o
対応関数名 : mapped_data
ファイルパス : /users/st/mapped_data.o

この場合の処理を以下に示す.

(a) ユーザが流動的データ型 Test を定義する.

(b) 指示エントリの順序集合を管理する UniSQL クラス "seq_of_text_segment" が, 流動的データ型で定義された指示エントリ型名より作成される.

(c) 流動的データ型 Test に対応する UniSQL クラス "Test" が作成される. UniSQL クラス "Test" は 3.2 節で述べた「流動的データ」クラスである. UniSQL クラス "Test" のインスタンスは UniSQL クラス「実データ」のインスタンスと "seq_of_text_segment" のインスタンスから構成される.

導出手続きで作成されるすべての指示エントリは, ユーザが指定した UniSQL クラス "text_segment" に格納される. UniSQL クラス "seq_of_text_segment" は指示エントリの順序集合を管理するクラスであり, 指示エントリの順序集合は UniSQL クラスのオブジェクトとして属性 PENT に格納される. 指示エントリの順序集合を管理する UniSQL クラスのクラス名はユーザが流動的データ型で定義した指示エントリ型名より決定される. PENT のドメインは SEQUENCE (domain list) であり, domain list には実行時に使用する指示エントリのデータ型名が指定される.

(2) 指示エントリの作成

導出手続き derived_data により指示エントリを作成する. ユーザが指定した実データに対して導出手続きを適用する. この例では, 図 3 に示したような指示エントリ PEnt1, PEnt2 が作成される. このときのクラス「流動的データ」のインスタンスを図 **4** に示す.

(3) 仮想セグメントの導出

仮想セグメントを導出するには, まず, 実データと指示エントリ作成時に使用した導出手続きをもとに指示エントリを求める. この例では, 指示エントリの順序集合 {PEnt1, PEnt2} が求まる. 次に, 求められた指示エントリの順序集合に対応関数 mapped_data を適用して仮想セグメント "ABCEFG" を導出する. 仮想セグメントはデータフォーマット変換関数により SJIS コードから JIS コードに変換されてユーザに参照される.

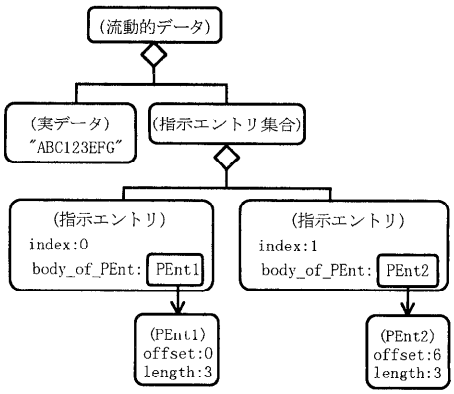


図 **4** クラス流動的データのインスタンス図
Fig. 4 An instance diagram of class liquid data.

5. 応 用 例

ここでは, MMRM を応用した 2 つの例を示す. 1 つは一次データベース参照機構であり, もう 1 つは画像データへのメモ入れ機構である. これらについて述べた後, MMRM の使用に関する考察を行う.

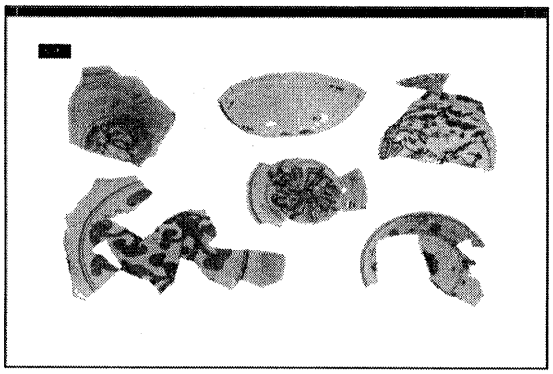
5.1 一次データベース参照機構

著者らが開発中のサイエンティフィックデータベース管理システム DREAM では, データを構造化せずに格納している一次データベース (基本データベース) から自分の参照したいデータを整理, 分別し, 独立した二次データベース (導出データベース) を構築する[7]. 導出データベース中のデータは, すべてデータエレメントと呼ばれる要素として扱われる. ここで, 導出データベースから基本データベース中のデータを参照可能とする機構に MMRM を利用した. 一次データベース参照機構はデータエレメントに指示エントリを格納することで容易に実現できた.

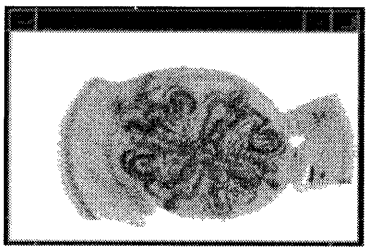
たとえば, ユーザ (ここでは, DREAM のデータベース管理者) は以下のように流動的データ型を定義したとする.

流動的データ型名 : data_elm
実データ型名 : X11bitmap
仮想セグメント型名 : Postscript
指示エントリ型名 : image_segment
導出手続き名 : derived_image
ファイルパス : /users/st/derived.o
対応関数名 : mapped_image
ファイルパス : /users/st/mapped.o

システムは図 **5** (a) に示した基本データベース中の



(a) 基本データ
(a) Basic data.



(b) 基本データ中の一部分の参照
(b)Reference a part of a basic data.

図 5 基本データ中の一部分の参照例
Fig. 5 An example of referencing a part of a basic data.

X11bitmap 形式の画像データを取り出し，導出手続き derived_image を実行し，基本データ中の構成要素を導出する．ここでは，図 5 (a) の中央の画像データを抽出したとする．導出手続きより作成された指示エントリの集合はデータエレメントを管理する UniSQL クラスに格納される．導出データベース中のデータ（仮想セグメント）は対応関数 mapped_image より生成される．生成されたデータは X11bitmap 形式から Postscript 形式に変換され，図 5 (b) のようにユーザに参照される．

5.2 画像データへのメモ入れ機構

次に，MMRM を用いて，画像データにメモ書きできる機構を作成した．画像データへのメモ書きは導出手続きにより行われ，メモの参照は対応関数により行われる．指示エントリ中には，ユーザが入力したテキストデータ，ならびに，メモを入れる場所の情報を格納しておく．この機構で使用する指示エントリのデータ型を以下に示す．

```
text_position {
        int     x, y;
        string  text;
    };
window_position {
        int     x, y;
        set(text_position)  text_data;
    };
```

ここで，メモ入れ機構の実行例を示す．

ユーザは以下のような流動的データ型を定義したとする．

流動的データ型名	: memo_elm
実データ型名	: X11bitmap
仮想セグメント型名	: X11bitmap
指示エントリ型名	: window_position
導出手続き名	: write_memo
ファイルパス	: /usr/st/write_memo.o
対応関数名	: read_memo
ファイルパス	: /usr/st/read_memo.o

上記の定義を行うと，MMRM は図 2 における「流動的データ型」クラスに上記の情報を格納し，「流動的データ」クラスに相当する UniSQL クラス “memo_elm” を作成する．

システムは，**図 6** (a) に示した画像データに，導出手続き write_memo を実行し，画像データにメモを書き込むものとする．

次にメモ入れ機構の導出手続き実行手順を示す．

(1) ユーザはウィンドウ上に表示されている画像データに対しメモを書き込むスペースをマウスで指定する．指定されたスペースにメモ書き用ウィンドウが生成される（図 6 (b)）．

(2) ユーザはメモをキーボードから入力する．メモはメモ書き用ウィンドウに表示される（図 6 (c)）．

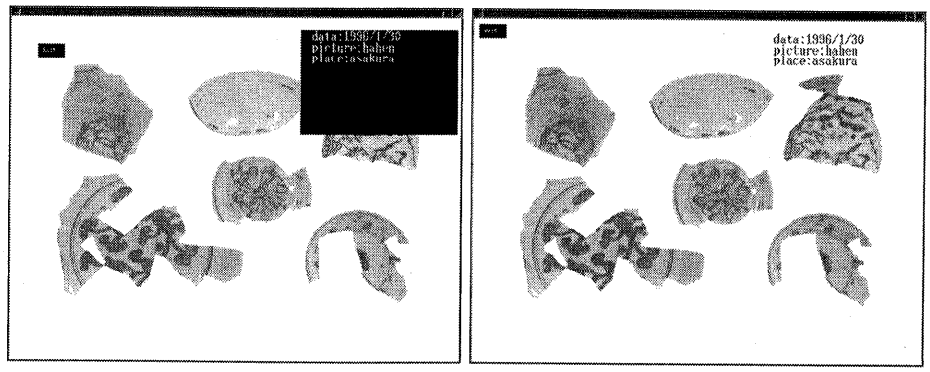
(3) メモである文字列の情報は UniSQL クラス “text_position” のインスタンスに格納される．ユーザが指定したメモを書き込むスペースの情報と “text_position” のインスタンスの集合が UniSQL クラス “window_position” のインスタンスに格納される．

これらの操作を経て指示エントリが作成される．導出手続きが終了すると，MMRM では，UniSQL クラス “memo_elm” のインスタンスに保持されている「指示エントリ集合」クラス “seq_of_window_position” のインスタンスにクラス “window_position” のインスタンスを追加する．

画像データに書き込まれたメモの参照は対応関数 read_memo により行われる（図 6 (d)）．

(a) 画像データ
(a)Image data.

(b) メモ書き用表示ウィンドウ生成
(b) Creation of a window for displaying memo written by user.



(c) テキストデータ入力
(c)Input of text data.

(d) メモの参照
(d)Referencing memo.

図 6 画像データへのメモ入れ機構
Fig. 6 An example of adding memo to an image data.

この機構を用いれば，図 6 に示したように，陶器の破片の写真の余白にメモを書くことが可能になる．

6. 評　　価

6.1 評価方法

プログラム記述量，ならびに，実行時性能の観点から MMRM の評価を行う．

(1) 記述量

前述の MMRM を用いた一次データベース参照機構と画像データへのメモ入れ機構のプログラム記述量と，MMRM を用いずに同様の機構を実現した応用プログラムの記述量を比較する．

(2) 性能

MMRM を使用した一次データベース参照機構と画像データへのメモ入れ機構と，MMRM を用いずに同様の機構を実現した応用プログラムにおいて，指示エントリ生成時間，ならびに，仮想セグメント導出時間を測定し比較する．測定は 10 回行い，平均値を求める．

6.2 評価結果

6.2.1 記　述　量

一次データベース参照機構を実現するためにユーザが記述しなければならないプログラムの規模は，C++ で約 360 行（コメントを含まず）である．内訳は，導出手続きが約 180 行，対応関数が約 180 行である．MMRM を用いずに一次データベース参照機構と同様の応用プログラムを作成した場合，プログラムの規模は約 1100 行である．

画像データへのメモ入れ機構を実現するためにユーザが記述しなければならないプログラムの規模は，C++で約 430 行（コメントを含まず）である．内訳は，導出手続きが約 250 行，対応関数が約 180 行であ

表 2 プログラム記述量の比較
Table 2 Comparison of lines of code.

応用例	MMRM 使用時のプログラム記述量（A）	MMRM 不使用時のプログラム記述量（B）	A/B（%）
一次 DB 参照機構	約 360 行*	約 1,100 行*	33
メモ入れ機構	約 430 行*	約 1,200 行*	36

* コメントを含まない．

表 3 指示エントリ生成時間
Table 3 Time in creating a pointing entry.

MMRM	一次 DB 参照機構（秒）	メモ入れ機構（秒）
使用	20.10	22.88
不使用	1.99	4.23

表 4 仮想セグメント導出時間
Table 4 Time in deriving a virtual segment.

MMRM	一次 DB 参照機構（秒）	メモ入れ機構（秒）
使用	2.07	2.23
不使用	2.05	2.22

る．MMRM を用いずにメモ入れ機構と同様の応用プログラムを作成した場合，プログラムの規模は約 1200 行である．

以上の結果をまとめて**表 2** に示す．一次データベース参照機構においては MMRM を用いると約 1/3 のプログラム記述量で実現でき，また，画像データへのメモ入れ機構は約 2/5 のプログラム記述量で実現できることが分かる．なお，MMRM のプログラム記述量は約 1600 行（コメントを含まず）である．

6.2.2 性　　能

一次データベース参照機構と画像データへのメモ入れ機構における指示エントリ生成時間を**表 3** に示す．表 3 より MMRM を使用した応用プログラムと MMRM を使用しない応用プログラムにおける指示エントリ生成時間に大きな差があることが分かる．これは，指示エントリ生成時に，MMRM を使用した応用プログラムでは，ユーザが指定した導出手続き名とそのオブジェクトファイルのファイルパスに基づき，導出手続きのコンパイルを行うからである．コンパイルの時間が指示エントリ生成時間に含まれているため，MMRM を使用しない応用プログラムと比べて指示エントリ生成時間が 20 秒以上も遅くなる．

次に，仮想セグメント導出時間を**表 4** に示す．表 4 より，MMRM を使用した応用プログラムと MMRM を使用しない応用プログラムにおける仮想セグメント導出時間には差があまりないことが分かる．これは，仮想セグメント導出時に実行する対応関数のコンパイルが流動的データ型の定義時に行われるためである．

また，流動的データ型定義時間は，20.76 秒である．この大部分は対応関数のコンパイルを行う時間である．

6.3 考　　察

プログラム記述量の評価より，一次データベース参照機構，画像データへのメモ入れ機構のいずれにおいても MMRM を使用することで小規模のプログラミングのみで，マルチメディアデータの一部分の参照が可能になることが分かった．

性能評価の結果より，仮想セグメントの導出時間は，MMRM を使用してもほとんどオーバヘッドがないことが分かった．ただし，指示エントリの作成や流動的データ型の定義では，導出手続きや対応関数のコンパイルのために大きなオーバヘッドが生じる．しかし，指示エントリの作成や流動的データ型の定義は仮想セグメントの導出と比較してあまり頻繁に行われないと考えられるため，実用上支障は少ないと考えられる．また，上記のコンパイルは MMRM のプラットホームとして使用した OS が関数へのダイナミックリンクをサポートしていないため必要である．関数へのダイナミックリンクが可能な OS を使用すれば，このコンパイルは不必要となり，指示エントリ生成時間や流動的データ型定義時間は改善されると考えられる．

また，MMRM を使用した 2 つの応用例の実装においては，導出手続きと対応関数の役割分担が非常に有効であった．すなわち，導出手続きは実データから 1 つ以上の指示エントリの作成を行い，対応関数は指示エントリから実データの一部分を導出する．これは一種のプログラム作成のガイドラインとなっており，プログラムの記述が容易になっていると考えられる．

7. ま と め

本論文では，流動的データ型の枠組みを用いた MMRM の実装とその有効性について述べた．設計には，オブジェクト指向ソフトウェア開発方法論である OMT を用いた．実装には，C++，ならびに，商用のオブジェクト指向データベース管理システム UniSQL を用いた．MMRM のプログラム規模は約 1600 行（コメントを含まず）である．さらに，MMRM を用いた応用例として，一次データベース参照機構，画像データへのメモ入れ機構を実現した．これらを用いて，プログラム記述量，ならびに，実行時性能の観点から評価を行った結果，小規模のプログラムによりマルチメディアデータの一部分の参照が可能になること，ならびに，仮想セグメントの導出には MMRM を使用してもオーバヘッドはほとんどないが，指示エントリの作成と流動的データ型定義にはかなりのオーバヘッド

があることを明らかにした．ただし，指示エントリの作成と流動的データ型の定義は，あまり頻繁に行われないため問題は少ないと考えられる．また，これらのオーバヘッドは使用するOSのダイナミックリンク機能により低減可能である．さらに，導出手続きと対応関数の区別がプログラム作成のガイドラインになっており，プログラミングを容易にしていることが分かった．以上により，MMRMはマルチメディアデータの一部分を参照する汎用的な機構として有効であるといえる．

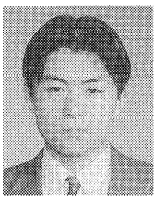
今後は，音声，ビデオといった実時間的なデータへのMMRMの適用，再帰的な仮想セグメントの作成，ならびに，実データの更新への対処が課題である．

謝辞 本研究は，一部，文部省科学研究費重点領域研究（1）（課題番号09204113）による．

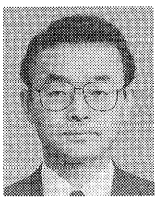
参考文献

1) 宝珍輝尚：データフォーマット可変な流動的データを用いたマルチメディアデータに対する視点の実現，電子情報通信学会論文誌（D-I），Vol.J73-D-I, pp.580–588 (1990).
2) Hochin, T. and Tsuji, T.: On the Application Interface of the LIQUID Data Type for Flexible Manipulation of Multimedia Data, *Proc. Int'l Symposium on Advanced Database Technologies*, pp.200–206 (1994).
3) 原田正則，宝珍輝尚，中田 充，都司達夫：流動的データ型を用いたマルチメディアデータ参照機構の設計と実装，1996年電子情報通信学会総合大会，D-64 (1996).
4) Zdonik, S.B.: Incremental Database System: Database from the Ground up, *Proc. ACM SIGMOD 1993*, pp.408–412 (1993).
5) UniSQL Inc.: UniSQL Database Products Application Program Interface Reference Guide (1995).
6) ランボー，J.，ブラハ，M.，プレメラニ，W.，エディ，F.，ローレンセン，W.（著），羽生田栄一（監訳）：オブジェクト指向方法論OMT，トッパン (1992).
7) 中田 充，宝珍輝尚，都司達夫：サイエンティフィックデータベースのためのデータモデルの一提案，情報処理学会データベースシステム研究会，101-9 (1995).
8) Cristy, J.: *ImageMagick Online Manual*, E. I. du Point Nemours and Company (1995).

(平成8年9月17日受付)
(平成9年6月 3 日採録)



原田 正則（学生会員）
平成8年福井大学工学部情報工学科卒業．現在，同大大学院博士前期課程在籍中．マルチメディアデータベースシステムの研究に従事．



宝珍 輝尚（正会員）
昭和34年生．昭和57年名古屋工業大学電気工学科卒業．昭和59年同大大学院修士課程修了．同年，日本電信電話公社入社．NTT情報通信網研究所を経て，平成5年7月より福井大学工学部情報工学科助手，平成7年6月同助教授，現在に至る．博士（工学）．マルチメディアデータ管理，柔構造データベース，拡張可能データベース管理システム，考古学データベース，グラフィカルなデータベース問合せ，グラフに基づくデータモデルの研究に従事．電子情報通信学会，IEEE，ACM，日本情報考古学会各会員．

中田 充（学生会員）
昭和43年生．平成4年福井大学工学部情報工学科卒業．平成6年同大大学院修士課程修了．現在，同大大学院博士課程在籍中．サイエンティフィックデータベース管理システム，柔構造データベースの研究に従事．

都司 達夫（正会員）
昭和48年大阪大学基礎工学部電気工学科卒業．昭和53年同大大学院博士課程修了．同年福井大学工学部情報工学科講師．現在，同教授．工学博士．データベースシステム，プログラミング言語の研究に従事．著書「Optimizing Schemes for Structured Programming Language Processors」（Ellis Horwood）．電子情報通信学会，IEEE，日本情報考古学会各会員．