データベース・システム 81－8
（１９９１．１．１７）

IKD:
知識型システム － データベース統合化のための一構造について

伊藤　秀昭

中京大学
情報科学部　情報科学科

本稿では，知識型システムとデータベースを統合化するためのソフトウェアツールIKDの設計および構造について述べる．これら２つのシステムをより柔軟に概念レベルにおいて統合化するためには，双方のシステムが備える概念構造を記述することが必要である．本システムは，この目的に供するためのツールであり，知識表現システムの構成要素となる．さらに，IKDは，知識型システムとして実現されており，フレーム型知識表現システムを用い記述されている．この知識ベースは，５個のサブフレームシステムより成り，２種のサブフレームシステム，問題構造記述およびデータ構造記述サブフレームシステムが重要な機能を果す．前者は応用領域に存在する概念対象を，後者は関係スキーマを記述し，互いに参照する．

IKD:
The Structure of an Interface for Integrating Knowledgebase and Database

Hideaki Ito
Department of Information Science
School of Computer and Cognitive Sciences
Chukyo University
Tokodate 101, Kaizucyo, Toyota, Aichi 470-03, Japan

This paper presents an architectural overview, design concepts of the system, called IKD, a knowledge-based system, and describes data structure of some components of its knowledgebase. The system functions as an interface between a knowledge-based system and a relational database management system. In order to integrate them more flexibly and naturally at conceptual level, it is necessary to represent conceptual structure provided in them. This system is implemented in the frame-based knowledge representation system FKBUS. The knowledgebase of it consists of five sub-frame-systems. Here, two sub-frame-systems play more important parts, they are Problem-Structure-Description and Data-Structure-Description sub-frame-systems. The first is defined depending on requirements from a knowledge-based system, while relational schemata are represented in terms of the second. Both the sub-frame-systems are made reference to each other.

1 はじめに

近年，知識型システム (KBS と略記する) の適応分野の拡張が進められている．これに伴い，KBS が大量かつ巨大なデータを取り扱うことを可能とするという要求が生じている [OOSU]．例えば，CAD/CAM，情報検索，など．このとき，多くの KBS は，複雑かつ構造化された (知識) 情報またはデータを取り扱うが，既存の大規模なデータは比較的取り扱われることが少なかった．一方，種々の分野では，問題解決のために必要となるより基本的なデータは，既存のデータベース管理システム (DBMS と略記する) を用いデータベース (DB) またはデータベースシステム (DBS) として構築されていることも多い．このとき，ある特定の目的のために開発され，固有の知識表現を提供する KBS のために DB を再構築することは，負担であろう．したがって，KBS および DBS の両システムを統合化することが必要であり，この機能を備えるメカニズムが，汎用性および柔軟性の観点から，知識表現システムに備わっていることが望ましいと考えられる．

本研究は，統合化の対象となる KBS および DBS の備える概念構造を互いに共有することにより統合化を図ることを目的としている．ここで，概念構造は，あるドメインに設定される概念対象と，それらの関連を記述したものである [UENO]．一般に，知識表現は，個々の問題領域に応じて設計および実現される．このことは，2 つのシステムの間に存在するインタフェースは，応用分野，および DB の構造を反映するものでなければならないと考えられる．

これらのモデルをより柔軟に概念レベルにおいて統合化するためには，知識表現により記述された概念対象，および DB の記述する対象の関係を記述し，変換のためのメカニズムが必要となる．この要求に応えるために，KBS と外部の DB とのインタフェースとして機能する IKD (interface for Integrating a Knowledge-based system and a Database system) の研究，開発を進めている [ITO]．これは，知識表現システム KBUS (Knowledgebase BUilding System) の構成要素の一つとして実現されている．KBUS は，2 つの知識表現システムより成り，それらは，フレーム型知識表現システム FKBUS，およびルール型知識表現システム PKBUS である．ここで，IKD は，FKBUS の応用システムの一種として実現された KBS の一種である．

現在までに，KBS および DBS を統合化したシステムが，既にいくつか提案されている．それらは，統合化のモデルとして論理，フレーム，などを用いている．例えば，関係型 DB(RDB と記す) に格納されたデータの集合をフレーム型データ構造により記述する方法，特殊な問い合わせ処理メカニズムを付加する方法，および RDB にルールにより記述された文を処理するアルゴリズムを付加する方法，などである．この種のアプローチは，技術の利用であり，概念レベルによる統合化ではないと考えられる．

一方，DB と KBS を統合化するためにフレームおよび関係型データ構造の変換に基づくアプローチがある．KBS の備える概念構造はフレーム型データ構造により記述され，この記述が DBMS の提供するデータモデルに変換される．この変換の定義は，Reimer[REIM]，および Twine[TWIN]，などにより提案されている．Reimer は，フレーム型データ構造を非正規形データ構造に変換するための基本的な関数を定義した．Twin は，NAIM 概念スキーマとフレームの変換を提案している．

IKD と同様に，DB と KBS を統合化するためのインタフェースとして動作するシステムを実現する試みに KADBASE[HOWE] がある．このシステムでは，KBS および DBS のそれぞれのスキーマおよびデータ構造は，互いに独立に記述される．この記述された内容の実際の対応関係は，規則 (または手続き) により記述される．一方，IKD では，双方のシステムは，一つの概念構造を共有することにより統合化を達成する．したがって，変換規則はその構造の記述の中に埋め込まれる．これは，知識と手続きの一体化という観点から，有効であると考えられる．

なお，本システムでは，DBMS として関係型データ構造を提供する関係型 DBMS(RDBMS)[DATE] を用い．その RDBMS の備える言語は，SQL[DATE,JIS] である．

2 データベースの構造の記述

一般に，KBS の開発では，知識をより柔軟かつ直感的に理解の容易な形式により格納し，それを利用する推論機構に必要となるデータを供給するという立場から知識表現が決定されることに対して，DBS では，データを定形化された形式により記述する．

KBS および DB の開発では，まず，知識およびデータを格納するためには問題分野または対象領域が分析される．ここでは，問題分野に存在する概念的な対象を決定し，それらの間に存在する関係を明確にする．このような分析により得られた構造は，概念構造または概念モデルと呼ばれている．概念モデルを定義，設計するための一つの方法論として，DB では，E-R モデルが提案されている [TEOR]．このモデルでは，データアイテムグループは，大きく 2 つに分類され，それらは，実体の集合である実体集合および実体集合間の関連を記述するための関連集合である．これらの集合は，共にデータを物理的に格納するための基底関係として定義される．このような基底関係から問題に応じ設定されるビューが生成される．ここでは，この関係をビュー関係と呼ぶ．

一方，DBMS の構成要素の一つである DD/D[LEON] は，DB に定義される情報のスキーマ，応用プログラムに関するメタ情報，および利用者などの DB を記述する．本システムでは，次のような理由により，DD/D により記述および格納されるような情報が重要な役割を果たす．

(1) DB に定義されたデータを KBS において利用することが可能なデータ構造に変換するため．

(2) スキーマ情報を記述することにより，あるデータ (の集合) がその DB のどこの存在し，いかなる情報が検索することが可能であるのかいうことを記述するため．

(3) スキーマのデータ構造をより明確にするため．基底関係のみではなく，ビュー関係の生成方法などを定義することが可能とする．

(4) 問題の記述 (問題構造) とデータ構造 (スキーマ情報) を明確に区別し，各々の構造に応じた手続きを適切に記述するため．

このような要求事項を備えるためには，次のような機構が備わっていることが望ましい．

(1) 関係および属性の明確化．利用者は基底関係の定義に際して，ある固有の意味に基づき DB スキーマを定義する．さらに，各々のスキーマがいかなる構成要素 (属性) より成り，如何に解釈されるのかということが記述されていることが望ましい．

表 1 フレーム型データ構造と関係型データ構造の対応

フレーム型データ構造	関係型データ構造
フレーム	インスタンス(レコード) 関係スキーマ(実体・関連)
	フィールド(属性)
クラス	関係スキーマ
インスタンス	インスタンス(レコード)
スロット	フィールド(属性)
データ型	データ型　ドメイン
ドメイン	ドメイン
(ファセット)	
階層構造(汎化)	ビュー
	データベーススキーマ
(全体-部分)	ビュー
リンク(is-a, ako)	汎化
スロット	関連
メッセージ交信	手続きの起動
attached procedures	SQL
	レコード単位の基本操作

(2)　DB スキーマ定義の支援. 一般に, DB スキーマの設計は, 従属性に基づき定義される基底関係, およびそれらを組み合わせることにより問題領域および要求から生じるビュー関係を定義することにより行われる. しかしながら, このような基底関係のスキーマおよびビューの生成を支援するためのメカニズムは, 既存の DD/D には備わっていない. これを支援するためにビュー関係または基底関係として定義すべきデータ構造を記述することを支援する方法がある. これは, ある関係を定義するために必要となる項目 (例えば, 関係名, 構成する属性, スキーマ間の関係など) を指定し, それを実現するための具体的な情報を設定するというものである.

(3)　DB に対する操作の明確な分離. DB の操作には, 次の 2 種がある.

- DB の管理, 更新および問い合わせのための操作.
- KBS が推論を実行するための操作.

これらは, それぞれ固有の目的および機能を果たす操作であり, 明確に区別すべきであろう. 例えば, DB 言語 SQL を用い, 新しいビュー関係を問題領域記述のために生成する. 次に, このビュー関係を推論を行うために検索し, KBS が実行するパタンマッチングに利用する.

(4)　オブジェクト指向によるデータのモデル化. RDB のデータ構造は平坦な構造である. したがって, これを知識表現に直接利用することは困難である. 一方, いわゆるオブジェクト指向は, データの集合に対する構造化または階層化のメカニズムを備えており, データの集合および要素はクラスまたはオブジェクトとして定義される [BLAH]. さらに, このようなオブジェクトを操作するための手続きの定義は, オブジェクト内部に有する. これは, フレームも同様である.

次に知識表現システムの提供するフレーム型データ構造の構成要素および RDB の構成要素の対応関係について述べる. この概要を表 1 に示す. この表は, DB の構成要素を FKBUS の提供するいかなる要素により記述するのかという観点に基づき整理したものである. ただし, この表では, 関係型データ構造固有および対象領域固有の対象の記述を区別していない. フレームは, RDB における基本的な関係である実体, 関連およびレコードの実現値 (インスタンス) を表したものであり, さらに, 関係の構成要素である属性もまたこれらを用い記述される. また, フレームにはクラスフレームおよびインスタンスフレームがあるが, これらはそれぞれ関係スキーマおよびレコードインスタンスに相当する.

多くのフレーム型知識表現システムでは, 汎化階層を利用するための環境を提供している. この階層は, DB 設計 [TERO] における汎化に相当する. したがって, この構造を用い定義された階層はビュー関係および基底関係の定義する.

3　システムの概要

3.1　本研究の動機および目的

現在までに, 我々は, NIRS(Network-based Information Retrieval System) と呼ぶ, 知識型情報検索システムの開発を進めてきた [ITO]. NIRS は, KBUS の構成要素である PKBUS, および本システムの記述言語である Lisp により記述されている. このシステムの開発に伴い, NIRS が大量のデータを処理するという要求が生じてきた. この種のデータはむろん DB に格納されることとなることを想定している. これを達成するための文献データまたはドキュメントを既存の RDB を用いることにより記述する方法は既に検討されている [ASHF]. 我々は, IKD を上記の要求に応えるために, DB の統合化を始めた. このとき, PKBUS および NIRS に対し, ある特定の機能 (関数) を組み込むことにより RDB をアクセスするメカニズムを実現することが可能であると考えていた. これは, PKBUS の備えるインタプリタに DB を検索するためのある手続きを埋め込むことにより達成されることになるであろう. しかしながら, このアプローチは採らなかった. なぜなら, NIRS および RDBS を結合するための固有のメカニズムを開発および提供することよりむしろ重要なこととして, KBS と RDBMS を統合化するためのインタフェースを提供することが有効でると考えたからである.

IKD の設計および開発の目標は, 次のようなことである.

- KBUS に RDBMS のためのインタフェースの提供. KBUS の適用分野が拡張され, システムの実用性の向上が期待できる.
- モデル化のためのシステムの提供. DB を柔軟に操作するためには, 関係スキーマの集合 (DB スキーマ), メタ情報およびその概念構造を KBS が保持していることが望ましい. DBMS ではこの種の情報の一部は DD/D において記述されているが, KBS がこの種の情報を保持することより, DB および KBS がモデル化している概念対象の相互関係がより明確となる.
- 概念構造の記述. システムがこの種の構造を操作することが可能であるならば, KBS はそれ自身の権限により DB を操作することが可能となる.

ただし, 本稿においては, 情報検索に対する適応については述べない. これについては, 機会を別に述べる予定である.

3.2　IKD のソフトウェア構造

図 1 に知識表現システム KBUS および IKD を構成するサブシステムのソフトウェア構造を示す. KBUS は三種のサブシステムより成り, それらは, DB 統合化インタフェース IKD, フレーム型知識表現システム FKBUS およびルール型知識表現システム PKBUS である.

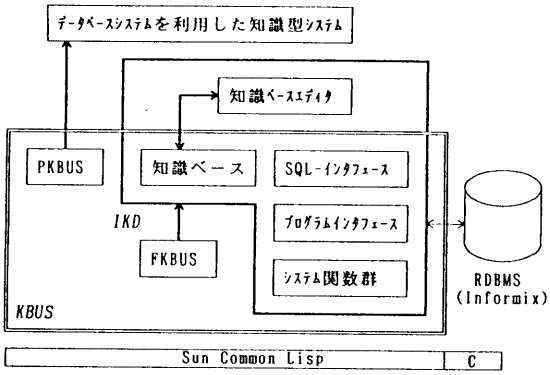

図 1 KBUSおよびIKD のソフトウェア構造の概要

IKD は，KBS として実現されており，次の 3 個の構成要素より成る．

(1) 概念構造を記述するための知識ベース．

(2) 知識ベースを構成するフレーム (群) を更新，編集するための知識ベースエディタ．これを，IKD エディタと呼ぶ．

(3) システム関数群．DB を物理的にアクセスするための関数群，および概念構造を記述するフレーム群を操作するための関数群，などより成る．むろん，これらの関数群を用い IKD は実現されており，KBS のための推論プログラムが記述される．

IKD エディタは，知識ベースを構成するフレーム群を操作することが可能であるばかりではなく，DB を操作することが可能である．また，IKD 固有の目的を果たすために設けられている種々のフレーム群を操作する知識ベースエディタとなっている．

本システムの応用システムは，KBUS の備える 3 種のサブシステム (IKD，FKBUS，PKBUS) を組み合わせ実現される．FKBUS は，IKD を実現するためにのみ用いられているのではなく，むろん他の応用システムの開発を行うことも可能である．しかしながら，応用システムが DB を利用するならば，FKBUS および IKD を利用することが必要である．

DB と IKD との相互作用を行うために，SQL(SQLI) およびプログラムインタフェース (PI) がある．

－ SQLI:SQL 文の解析と実行の依頼．DB 言語 SQL により記述された文は，SQLI を通じ DBMS にその評価が依頼される．この SQLI では，個々の SQL 文が解析され，1 つのフレームシステムとなる．

－ PI: システム関数群の一部．DB の検索をタプル単位に行うための関数，関係のカーソルのオープンを行う関数など，がある．

次に KBS と DBS との結合の方法について述べる．既に，2 つのシステムを結合するための方法には，そのアーキテクチャの観点から，大きく疎結合および密結合と呼ばれる方法がある [GRAD]．前者に基づくアプローチでは，利用者は二つのタイプの言語を利用する．すなわち，KBS に備えられている言語および DB 言語である．さらに，これらのシステムは，互いに独立に機能する．一方，後者のアプローチは，一方のシステムを拡張することにより他方のシステムの備える機能を実現または組み込もうとするアプローチである．

IKD は，図 1 に示したように，疎結合に基づくシステムを実現するためのツールとして実現されたシステムである．このようなアプローチを採った理由は，次のようなことである．一般に DBMS は，データの集合を操作するための極めて精巧な手続きの集合体より成る．このような手続きを個々にある特定の KBS のために実現することは，極めて困難なことである．したがって，DBMS が備える機能および既存の DB を有効に利用することが適切であると考えられる．さらに，既に我々は，PKBUS を実現していた．

IKD は，フレーム型知識表現システム FKBUS および SunCommonLisp[SUN] を用い記述され，DBMS として Informix[ASCII] を用いている．Informix との物理的な結合を図るプログラムは，C 言語により記述されている．なお，本システムは，富士通 S-4/1 ワークステーションにおいて稼働する．

3.3 知識ベースの構造

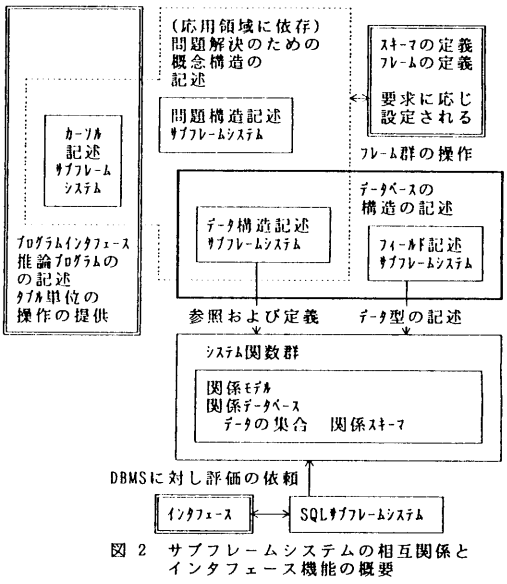

図 2 サブフレームシステムの相互関係とインタフェース機能の概要

IKD を実現する知識ベースは，次の 5 個のサブフレームシステムより成る．

(1) 問題構造記述サブフレームシステム

(2) データ構造記述サブフレームシステム

(3) フィールド記述サブフレームシステム

(4) カーソル記述サブフレームシステム

(5)SQL サブフレームシステム

それぞれのサブフレームシステムを構成する個々のフレームを，問題構造記述，データ構造記述，フィールド記述，カーソル記述，および SQL フレームと，それぞれ，呼ぶことにする．なお，本稿では，問題構造記述およびデータ構造記述フレームをそれぞれ，PSDF および DSDF と記すことにする．

サブフレームシステム (1)，(2) および (3) は，DB の構造を記述し，格納された情報を利用するために設けられ

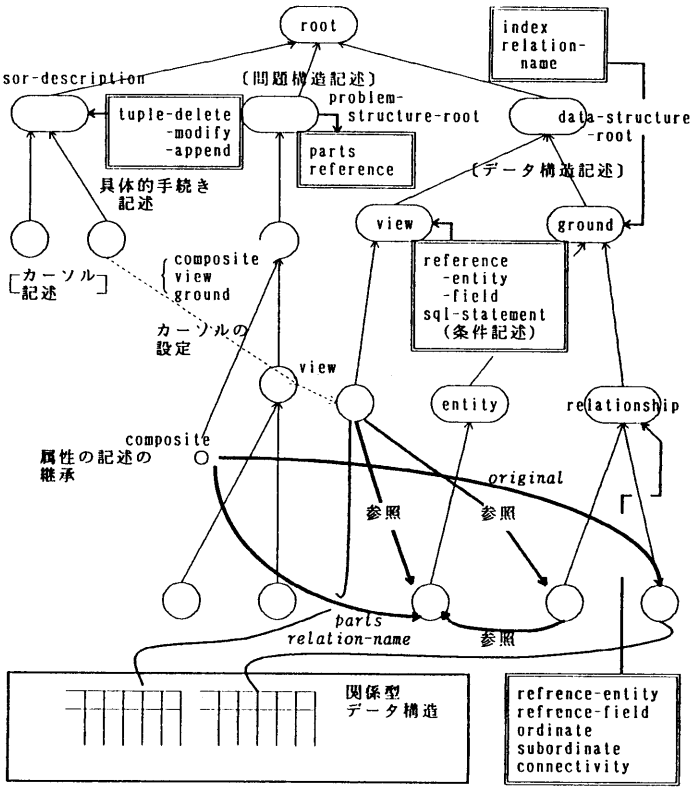

図 3　問題構造記述、データ構造記述フレーム
および関係スキーマの相互関係

ている．それらを構成するフレームは，物理的に存在するDBスキーマと互いに関連づけられている．DSDFは，DBに定義された基底関係およびビュー関係のデータ構造を記述したフレームである．一方，PSDFは，問題分野に存在する概念対象を記述する要求に応じ定義されるフレームである．この種のフレームにおいていくつかのフレームがDBに定義されている適切な関係を参照する．この参照は，DSDFを通じて行われる．このとき，PSDF群が定義されることに応じて新たな関係が，以前に定義されている関係より導出されDBに定義されする必要がある場合がある．さらに，いくつかのPSDFは，直接一つのDBスキーマへの参照を生成するだけではなく，応用分野の要求に応じDBを参照しないPSDFが生成される場合がある．

フィールド記述フレーム(FDF)は，関係を構成する個々の属性のデータ型を定義するフレームである．この種のフレームには，Lispシステムと利用しているDBMSの備えるデータ型の対応関係，データ型変換のための手続きを記述したスロット，などを有する．

サブフレームシステム(1)および(4)は，応用システムを実現するための要求に応じて構成されるサブシステムである．これらのサブシステムに属するフレーム群は，次のように利用される．まず，カーソル記述フレーム(CDF)は，適切なあるPSDFを参照する．この参照されたPSDFの内容に基づき1つのビュー関係が設定される．CDFは，この生成されたビュー関係に設定されるカーソルを記述するフレームとなる．ここで，カーソルは，ある関係を構成する一つのタプルを指し示すポインタである．KBSからのDBの有するデータの検索，更新は，すべてこのCDFを通じてなされ，タプル単位に処理される．この種の操作は，個々のCDFに付加されたattached procedures(AP)により実現されている．ただし，一般的なDBの更新に関わる手続きを想定し，その手続きのためのメカニズムを実現することは現時点において困難であるため，これを記入するためのスロットのみを提供している．したがって，利用者がこのメカニズムを利用しようとするならば，注意深くその手続きを具体的に記述しなければならない．むろん，これを利用するためには，CDFにより参照され，そのフレームを定義するDSDFが参照するPSDFの定義に関わる関係は，DBにおいて更新可能なものでなければならない．IKDは，DBの更新のための基本的な関数群を備えているのみである．

SQLサブフレームシステムは，利用者により記述されたSQL文を解析する．このとき，与えられた個々の文を

```
frame-name: struct-r
frame-type: data-r   ;データ構造記述フレームへのポインタ
relation-type: { composite    ;
                 view         ;*undefined*は概念構造を
                 ground       ;記述するために設けられた
                 *undefined*  ;フレームであることを示す
original-table: ; 関係型が"composite" が選択されたならば
                  合成操作を行うためのフレームが
                  指定されていなければならない
field-names:    ; 属性の名前のリスト
; 属性を記述するためのスロット
; これらのスロットは下位のフレームに継承される
field-1: [value]     ;このフレームにおけるスロット値
       ;以下のファセットは必要に応じ設定される
         [data-type] ;このスロットのデータ型
         [domain]    ;ドメイン
         [default]   ;デフォルト値.
         [reference] ;フレームを通じ参照される属性
           :
parts:     ; ある関係を構成するために部分となる関係を
           ; 記述する他のフレームへのポインタ
join-key:  ; join操作を行うために必要となるフレーム
           ; およびスロットへのポインタ
key-1:     ;  "join-keys"の具体的な記述
           :
```

スロット"parts"および"join-keys" は、"relation-type" が"composite" であるとき、自動的にそれらのスロットが生成される。属性を記述するためのスロットは下位のフレームに継承される。

図 4　問題構造記述フレームのフレーム型データ構造の一部

```
frame-name: data-r
frame-type: instance
relation name:  ;外部データベースにおける関係名
struct-relation-name: struct-r   ;相当する問題構造記述
                ;フレームへのポインタ
table-type:  { entity       ; 実体
               relationship ; 関連
               view         ; ビュー
field-names:    ; 属性名のリスト
; 属性を記述するためのスロット
field-1:    [value] ; フィールド記述フレームへのポインタ
            ; 以下の要素はファセットとして実現されている
            [unique] ;ユニーク値であるかどうかのフラグ
            [name]  ; 関係における実際の名前
              :
primary-key:   ;主キーの設定

index:
index-type:    } ; インデックスの記述
index-field:
connected-entity:  ;テーブル型が"relationship"であるとき
                   ;参照するフレームへのポインタ
connections:   ;結合を記述するスロットへのポインタ
con-k:         ;結合の具体的記述
foreign-keys:   ; 外部キーの設定
sql-statement:   ;ビュー関係ならば、
                 ;それを定義するSQL文の記述
```

図 5　データ構造記述フレームのフレーム型データ構造の一部

構成するパラメタは，適切なフレームのスロットに設けられているスロット値に埋め込まれる．

サブフレームシステムの相互関係と個々のサブフレームシステムの機能の概要を図 2 に示す．

図 3 に IKD 知識ベースの階層構造の概要および知識ベースとなるフレーム間の参照の概略を示す．この図において，二重線により囲まれている情報は，フレーム階層においてそのレベルに定義される情報である．なお，この図では，SQL および FDF サブシステムは省略されている．

4　サブフレームシステム

4.1　データ構造および問題構造記述フレーム

図 4 に PSDF のフレーム型データ構造の一部を示す．この種のフレームには，次の 2 種のシステムスロットが設けられており，一方は DB を結合するために，他方は FKBUS 知識ベースを管理するために設けられている．前者の主たるシステムスロットには，次のようなものがある．

data-relation-name: ある関係スキーマを記述する DSDF へのポインタ．

relation-type: この PSDF により参照される関係の種類．この関係の型の種類には，“composite”，“view” および “ground” の 3 種があり，さらに，ある関係スキーマを参照しないような場合には，“*undefined*” と記入される．上記 3 種の関係の型は，それぞれ次のような関係の種類を指定する．関係型 “composite” は，ある一つの関係および複数の関係から導出される関係を，“view” はビュー関係を，“ground” は基底関係を，それぞれ，参照する．ここでは，関係型 “composite” を有する PSDF を，コンポジットフレームと呼ぶ．

original-table: 関係型 “composite” が指定されたとき，その関係を構成するためのオリジナルとなる他の PSDF または DSDF へのポインタ．このスロットは，自動的に生成される．

field-name: この PSDF により参照される関係を構成する属性名のリスト．このリストを構成する個々の要素をスロット名とするスロットが，この PSDF に生成される．この種のスロットは，参照された関係の属性を記述したものであり，インヘリタンスの対象となる．この種の個々の属性を記述するためのスロットは，いくつかのファセットより成る．これには，個々の属性のデータ型，デフォルト値，およびドメインなどがある．

DSDF の概要を図 5 に示す．これは，具体的な関係スキーマの構造を記述したフレームであり，インスタンスフレームとして定義される．主たるスロットには，次のようなものがある．

relation-name: この DSDF により参照される DB に定義されている実際の関係名．

struct-relation-name: 相当する PSDF へのポインタ．

table-type:DSDF が記述している関係の型の指定．この種のフレームにおいて指定することが可能である関係型には，“entity”，“relationship” および “view” がある．これらは，それぞれ，実体集合，関連集合，およびビュー関係を参照するフレームであることを示している．

4.2　データ構造記述フレームおよびフィールド記述フレーム

上の節において述べたように，DSDF はある 1 つのスキーマを記述するために設定されたフレームである．この種のフレームにおいて，スキーマを構成する個々の属性は，スロットとして記述される．この種のスロットの値部には，ある FDF へのポインタが記入される．いくつかの FDF が DBMS の提供するデータ型に基づき組み込みフレームとして設定されている．例えば，順序付き整数 “serial”，文字列 “string”，整数 “smallint”，などがある．

図 6 に IKD エディタを用い，ある DSDF を表示した例を示す．ここで，定義される DSDF の名前は，‘a’ であ

```
DataObjectEdit: print
ObjectName: a
EntityName:                a
RelationName:              "a"
<<Attributes>>
name:          RealName:           DataTypeInDB:  DatatypeInLisp:
aa             "aa"                smallint       integer
bb             "bb"                char(40)       string
cc             "cc"                char(40)       string
<<Index>>
IndexName: index-a       IndexType unique
Field:                         Order:
aa                             asc
```

図 6　データ構造記述フレーム"a"の表示

```
Lisp => (message 'make-sql-statement-define 'a 'a)
"create table a (aa smallint, bb char(40), cc char(40))"
```

図 7　SQL(CREATE TABLE)文の生成

り，このフレームにより参照される関係は実体集合 (entity) の 1 つである．この関係には，“aa”，“bb” および “cc” の 3 個の属性が定義されている．これらの属性のデータ型は，それぞれ，“smallint”，“char(40)” および “char(40)” である．さらに，一つのインデックスが定義され，その名前は ‘index-a’ である．

FDF サブフレームシステムにおいて，フレーム “smallint” は，インスタンスフレームとして定義されている．これは，この型を有するデータが，整数であるためユニークに解釈することができるためである．

一方，データ型 ‘char’ により定義されるデータの長さは，可変長である．これに対応するため，フレーム ‘char’ は，クラスフレームである．このとき，このフレームのインスタンスフレームは，実際の要求に応じ定義された個々の可変長の長さに対応したデータ型に応じ定義される．この例では，文字列の長さは，属性 “bb” および “aa” において 40 と指定される．この長さを有するデータのためのフレーム char-40 が生成される．ここでは，フレーム ‘a’ に定義されているスロット “bb”，および “cc” スロットの値部には，フレーム “char-40” へのポインタが記入される．ここに示したように相異なる属性の定義において同じデータ型が参照されたならば，そのような属性を定義する FDF を参照する．このとき，インヘリタンスの機能を利用している．この場合，フレーム “char” とフレーム “char-40” との相違は，いくつかのシステムスロットを除き，文字列の長さを記入するスロットのみである．この種のフレームを構成するスロットには，例えば，Lisp システムにおけるデータ型を記入したスロット，対応する DBMS におけるデータ型，データ型に基づき個々のデータを変換するための AP，などが定義されている．

DSDF には，いくつかの AP が定義される．これらの手続きは，個々の DSDF にフレーム階層の上位に位置する “ground-relation” および “entity” フレームなどより，受け継がれる．そのような AP の 1 つに，スロット “make-sql-statement-define” がある．この AP は，設定された内容に基づき，関係スキーマを定義する SQL 文を生成するための手続きである．この AP は，このフレームに対するメッセージ交信により起動され，その例を図 7 に示す．

5　記述例

ここでは，PSDF，DSDF および DB に定義されている関係スキーマとの相互関係を簡単な例により示す．

ここで DB は，‘車’ に関するデータを格納するとする．対象としている分野の概念構造を図 8 に，対象とする DB の E-R ダイアグラムを図 9 に，それぞれ示す．

図 9 において，四角および実線の矢線は，それぞれ，フレームおよびそれらを結合する ‘a-kind-of’ リンクである．フレーム “roof-root”，“automobile”，および “door-root” が頂点となっており，個々のフレームは，それぞれ，“屋根”，“車両” および “ドア” を記述する概念対象を記述するとする．さらに，“車両” の下位概念として “乗用車 (car)”，その下位概念として “セダンタイプの乗用車 (sedan)” がある．また，乗用車の部分となる構成要素には種々のものがあるが，ここでは屋根およびドアを部分となる構成要素であるとする．

3 個の頂点となっているフレームの間には，全体－部分関係がある．それは，‘車両－屋根’ および ‘車両－ドア’ である．フレームを用いた全体－部分関係の記述は，次のようになされることになるであろう．まず，全体を記述するフレームにあるスロットを設ける．このスロットには，部分となる構成要素を記述するフレームへのポインタが記入される．個々の部分を記述するフレームには，全体を記述するフレームへのポインタが記入され，これらの双方向ポインタによりネットワークが管理される．

フレーム “car-root”，“car” および “car-door” は，DB に定義された関係を用い格納されたデータを参照するフレームである．このとき，この種のクラスフレームのインスタンスは，DB から得られるインスタンスに相当する．このためには，タプルからフレームへのデータ構造の変換が必要である．ここで，フレーム “car” に指示されるインスタンスの集まりは，フレーム “car-roof”，“car-door”，関連集合を記述するフレーム “ground-car”，および DB に設定されている具体的な関係より得られる．ここで，フレーム “car” の有するスロット “relation-type” は，“composite” であり，これより乗用車の具体的なインスタンスは，複数の関係より得られる関係のインスタンスであることが示されている．この関係の合成は，関係代数

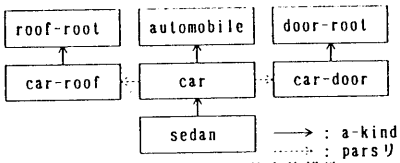

図 8　対象領域における基本的構造

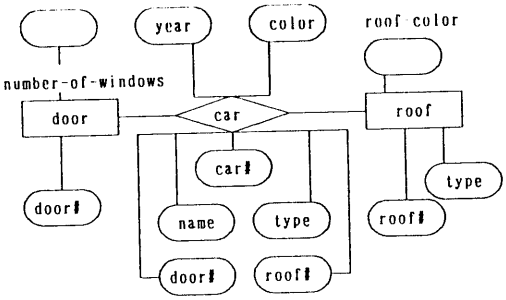

図 9　E-Rダイアグラムの例

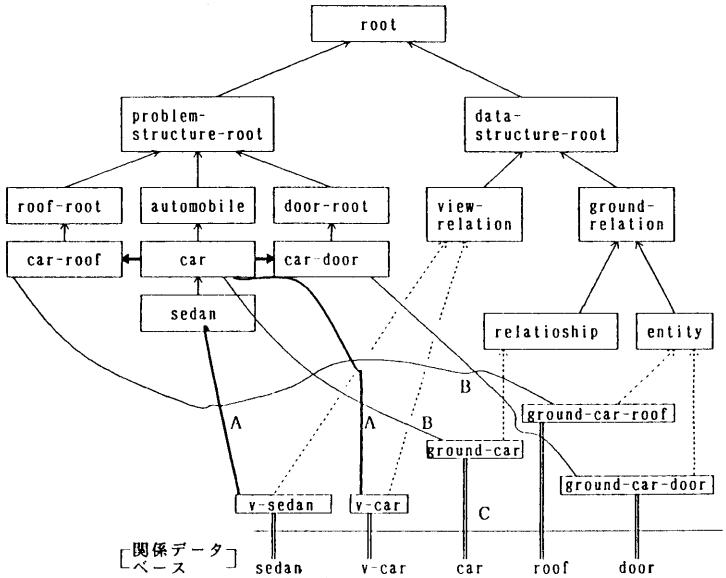

図10　問題構造記述フレーム、データ構造記述フレームおよび関係スキーマの相互関係

に定義されている結合演算により得られる.

図10に，この節において想定している対象領域を記述するための知識ベースの階層構造の概要を示す．この図には，DSDF，PSDFおよび統合化される関係の名前を示している．さらに，DBには，3個の基底関係が定義されており，それらは関係“car”，“roof”および“door”である(図9参照)．これらの関係において，関係“car”は関連集合であり，“roof”および“door”は実体集合である．これらの基底関係を構成する属性は，定義されている基底関係と関連づけられている．また，関係“car”は関連集合として定義されているが，これは実体集合“door”および“roof”を互いに関連づける関連集合である．これを実現するためにそれらの関係には，それぞれの識別子を設定する属性“door#”または“roof#”が定義されている．

このようなDBを利用するとき，例えば，属性“roof#”または“door#”のように単純なデータだけではなく，乗用車に関するデータを求めることが必要である．例えば，ある乗用車に適するドアのタイプ，ドアの識別番号，屋根の色，ドアの色などである．これらの情報を得るためには，関係演算である結合，選択などを用い，ある1つのビュー関係を設定することが必要であることが多々あると考えられる．

例えば，次のようなビュー関係を定義することが要求されたとする．

```
v-car(car#, name, type, year, color
      number-of-windows, roof-type, roof-color)
```

例えば，次のselect文を用いることにより，要求されたビュー関係を記述することができる．

```
select
  car#, name, type, year, color,
  number-of-windows, roof-type, roof-color
 from car, roof, door
 where car.door# = door.door#
  and car.roof# = roof.roof#
```

このselec文により，‘v-car’において要求されたスキーマを有する関係が得られる．さらに，このselect文を伴うcreate view文により，条件を満たす1つのビュー関係が得ら

```
EditProblemStructure: pr
ObjectName: car
RelationObjectName:          car
DescriptionInformation:      *undefined*
GenericRelationObjectName: automobile
DataObjectName:              v-car
RelationType:                composite   OriginalRelationName: ground-car
<<Attributes>>
name:          type:      domain:        default:       reference
car#           serial     integer        nil            <inherent>
name           (char 20)  t              nil            <inherent>
type           char       t              nil            <inherent>
year           integer    t              nil            <inherent>
color          char       t              nil            <inherent>
number-of-window integer  t              1              <car-door : number-of-window>
roof-type      integer    t              nil            <car-roof : type>
roof-color     char       t              nil            <car-roof : color>
PartsObjects:              car-roof car-door
<<< Join Keys >>>
ground-car     door#          ::car-door     door#
ground-car     roof#          ::car-roof     roof#
```

図 11　フレーム"car"のIKDによる記述

れ，ここで望まれている情報は，このビュー関係のインスタンスとなる．

図10に，これまでに述べてきたPSDF，DSDF，および外部DBの提供するDBスキーマの関係を示す．この図において，A，BおよびCの3種のリンクがある．リンクAは，PSDFからDSDFへのポインタであり，BはDSDFよりPSDFへのポインタ，さらにCはDSDFよりそれが参照する関係スキーマへのポインタである．

図11にPSDFフレーム“car”のIKDエディタによる表示例を示す．スロット‘a-kind-of’はそのフレームの階層における上位フレームを示すスロットであが，これは，アイテム“GenericRelationName”により示されている．

フレーム“car”は，コンポジットフレームである．これは，アイテム“RelationType”に示され，この合成を行うために基となる関係は，アイテム“OriginalRelationName”に示されている．このアイテムの値はDSDFとして定義されており，それはフレーム“ground-car”である．

PSDF“car”により参照される関係を構成する属性の名前の列は，アイテム“≪Attributes≫”により示される．個々の属性の記述は，5種の項目より成り，それらの項目には，次のような構成要素がある．

value: そのスロットの値部

type:DBMSに備えられているデータ型．

domein: ドメイン．特に指定されていないならば，如何なるデータであってもその属性の値となることを示すtが設定される．

default: デフォルト．デフォルトを有しない場合，nilが設定される．

reference: 他の関係の属性を参照するかどうかの設定．その属性の参照する他のDSDF，またはPSDF，およびこのような属性を記述するスロットの名前の対により参照を記述する．

アイテム‘reference’の記述は，2つの種類の値が記入されている．一つは“inherent”であり，他方は2つの要素より成る．この情報は，そのスロットにより記述されている属性がそのフレームに定義されているスロットを参照するものであるかどうかということを示している．そのオブジェクトが記述する関係において固有なものであるならば，“inherent”と記入されている．このとき，コンポジットフレームであるならば，“OriginalRelationName”により示されてるフレームが記述する関係に定義された属性を参照することを示す．

6　まとめおよび今後の課題

本稿においては，IKDのソフトウェア構造，知識ベースの概略およびいくつかの構成要素について述べた．ここに述べた枠組みにより，DBの備える構造を記述することにより統合化を図る．

IKDは，疎結合によりDBSおよびKBSを統合化するためのツールとして開発が進められており，知識型システムの一種として実現されている．DBSに対する操作は，IKD知識ベースを通して行われる．この操作は，フレームへのメッセージを送ることにより実現される．したがって，KBSの推論機構の変更が必要となる．例えば，プロダクションシステムPKBUSのインタプリタでは，あるパタンとその構成要素であるデータベースの構成要素とのパタンマッチングを行うために，インタプリタはあるCDFにメッセージを送る．しかしながら，このメッセージ交信を繰り返すことは，システムの効率を極めて悪くすると考えられる．したがって，今後高効率化のためには，CDFを主記憶上にデータを移動するための機能を果たすように拡張することが，今後必要であると考えている．

現在，IKDについて，次のような観点から検討を進めている．

KBS向きDBの更新への対応．IKDのDBの更新は，現在，DBMS側に備えられている機能を利用している．しかしながら，一般に，KBSのトランザクションはインターバルの長いものとなる．これに対応するための機構が必要であると考えられる．

エンドユーザインタフェースの拡充．IKDのフレームシステムは，互いに関連づけられており，それを構成するフレームの修正は他のフレームの修正を伴う場合が多い．したがって，利用者のフレームシステム管理の負担は重いものとなる．これを解消するためのインタフェースには，知識ベースの一貫性の管理と柔軟性のある管理機構が必要となると考えている．

現在，本システムを用いPKBUSと統合化し，本システムをNIRSに適応する試みを進めている．これを通じて，

実行効率および操作性などの評価を行う予定である．これについては，機会を別に述べたいと考えている．

謝辞: 日頃ご指導下さる中京大学 福村晃夫教授に深謝いたします．また，IKDの一部は，(財)日本情報処理開発協会在職中に開発されたものである．IKDの利用およびその拡張を許して頂いた同協会 山本欣子氏，市川隆氏に深謝いたします．

References

[ABAR] Abarbanel, R.M., et al: KEEconnection: A Bridge between Databases and Knowledge Bases, Ed. Reicher,M.H., AI Tools and Techniques,Ablex, 1989

[ASCI] ASCII: Informix-esql/C Programmers Manual, 1988

[ASHF] Asford,J.et al: Text Retrieval and Document Databases, Chartwell-Brett, 1989

[BLAH] Blaha, M.R.: Relational Datbase Design an Object-Oriented Methodorogies, CACM, Vol.31, No4, 1988

[DATE] Date,C.J.: An Introduction to Database Systems, Addison-Wesley

[GARD] Gardarin,G., et al: Relational Databases and Knowledge Bases, Addison-Wesley, 1989

[HOWE] Howerd,H.C.,Rehac,D.R.: KADBASE: Interfaceing Expert Systems with Databases, IEEE Expert, Fall, 1989

[ITO] 伊藤: フレーム型知識表現によるデータベースの統合化に関する研究～知識表現システムのために～，(財)日本情報処理開発協会，1990

[JIS] JIS X 3005: データベース言語SQL, 1987

[KIM] Kim,W., et al: Composite Object Revised, Proc.SIGMOD, 1989

[LEON] Leon-Hong,B., et al: Data Dictionary/ Directory System, John-Wily, 1982

[MAKA] MaKay,D.P., et al: The Intelligent Database Interface: Integrating AI and Database Systems, Proc. 8th AAAI, 1990

[MORI] Morisons,L.,et al; Using Structural Knowledge in Database and Knowledge Base Integration, Proc. Data Eng., 1989

[OOSU] 大須賀: データベースと知識ベース，オーム社，1989

[REIM] Reimer,U. et al: A Framework Knowledge Representation Model and its Mapping to Nested Relations,Data & Knowledge Eng., Vol.4, 1989

[SUN] SUN: Sun Common Lisp Reference Manual, 1989

[TEOR] Teorey,T.: Database Modeling and Design, Kaufmann, 1990

[TWIN] Twine,S.: Mapping between a NAIM Conceptual Schema and KEE Frames,Data & Knowkedge Eng., 1989

[UENO] 上野: 知識工学入門, オーム社, 1989