知識工学と人工知能　52－10
（１９８７．５．７）

知的インタビューシステムI²Sに基づくデータベース構築・利用支援システム

川口　敦生　　沼田　薫*　　溝口　理一郎　　山口　高平　　角所　収
大阪大学産業科学研究所
（*　現在、野村コンピュータシステム(株)）

データベースの構築作業についてあまり経験のないユーザを対象とするデータベース構築・利用支援システムについて報告する．本システムは論理設計を担当する知的インタビューシステムI²Sを中心に，データベース構築用知識ベース，データベース管理システムとのインタフェースから構成されており，データベース構築作業のマニュアルレス化を目指している．システムはユーザにインタビューを実施して論理設計を行ない，データベースの試作を支援する．次にシステムが提供する自然言語による検索インタフェースを通してユーザに試作データベースを使用してもらい，問題点が検出されると再設計，再試作を繰り返す．検索インタフェースは，検索結果が空になったときにその理由をユーザに提示することも試みる．本稿では，このようなプロトタイピングの手法を用いてユーザが所望するデータベースの構築を支援するために必要な機能およびその実現法について述べている．また知識獲得における動的解析（dyanmic analysis）の観点から，本システムが用いたプロトタイピングの手法について検討を加えている．

A Support System for Database Construction based on Intelligent Interview System

Atsuo KAWAGUCHI, Kaoru NUMATA, Riichiro MIZOGUCHI,
Takahira YAMAGUCHI and Osamu KAKUSHO
ISIR, Osaka University
Mihogaoka 8-1, Ibaraki, Osaka, 567, Japan.

This paper describes a support system for database construction. The system consists of three parts, such as I²S:Intelligent Interview System for logical design, a knowledge base for database construction and an interface to database management systems. It interviews a user to design logical structure of the target database and constructs its prototype. Then, the system gets the user to use the prototype trying to detect problems of it. If some problems are found, the system redesigns and reconstructs the database. In this paper, several functions necessary for supporting users and their implementation are presented. We also discuss the prototyping method of the system from the aspect of dynamic analysis for knowledge acquisition in expert system.

1．はじめに

我々はこれまでデータベース構築作業全般をマニュアルレス化することを目指していくつかのシステムの検討，開発を試みてきた[磯本84][Mizoguchi85]．特に従来あまり検討されていなかった論理設計の支援を中心に研究を進めており，データベースの論理設計支援をそのタスクとする知的インタビューシステムI²Sを開発している［川口86］．I²Sはデータベースの発注者にインタビューを実施し，構築するデータベースの概念スキーマを出力するものである．論理設計作業の本質はドメインに関する知識を発注者からデータベースの専門家に的確に移行することであるという立場から，インタビューという概念を中心に据えている．

本稿では，このI²Sを核に据えたデータベース構築・利用支援システムについて報告する．支援の対象としては，データベースについての基本的知識は有するが，構築作業の経験は少ない発注者を想定している．本システムは発注者にインタビューを実施して論理設計，データベース生成を行ない，その後発注者に自然言語による検索機能を提供するものである．さらに検索時に問題が生じたときは，データベースの修正，データの追加などの修正支援を行う．なお，本システムは動的解析を行う知識獲得システムとみることもできるので，随時その立場からも検討を加える．

以下，次節でデータベース構築作業について述べる．同時に知識獲得の方法としての静的解析(static analysis)と動的解析(dynamic analysis)の考え方を紹介する．次に3節で本システムの概要を述べる．4節では本システムの第1版データベース構築支援機能について，また5節では第2版以降の修正支援機能および本システムが提供する自然言語による検索機能について述べる．最後に6節で本システムの現時点での問題点，限界などについて述べる．

2．データベースの構築作業

発注者がデータベース構築に関してあまり経験を持たない場合は，データベースの構築作業は一般に図1に示すように行なわれる．まず，データベースの専門家が発注者にインタビューを実施し，構築するデータベースの仕様を固める．この仕様から専門家は第1版データベースを構築し，発注者に引き渡す．発注者がデータベースを利用中に問題が生じると，発注者は専門家に連絡をとってデータベースを修正してもらう．これを問題が生じなくなるまで繰り返す．

このようなプロトタイピングが必要となるのは，次のような理由によって適切な論理設計を一度で行えないためである．

(1) 構築するデータベースのドメインについて，発注者が十分に整理できてない場合が多い．
(2) データベース構築において何が問題となるかということについて，発注者が知識を持っていない．
(3) 設計者は一般にドメインについての詳細な知識を持たないため，そのドメインでの常識が通用しない．

ここで(1),(2)は発注者がデータベース構築に関してあまり知識を持っていないということが背景となっている．

知識獲得の観点からは，以上のことはデータベースを構築するのに必要なドメインの知識を発注者からデータベースの専門家に移していく際の問題と見ることができる．第2版，第3版と修正が進むにつれて，発注者が持つドメインの知識により近い知識がデータベースの専門家に形成されていく．ただし，第1版構築時と第2版以降の構築時ではこのような知識の移行を行うための手掛りがまったく違う．すなわち第1版構築時には，データベースがまだ存在しないため，発注者が専門家に提示できる知識は曖昧にならざるを得ない．一方第2版以降では，利用中に生じた問題をもとにするので，経験が少ない発注者とでも設計者はより具体的に検討を進めることができる．

McDermott らは，エキスパートシステム開発のための知識獲得支援システムに関して同様の議論を行ない，MOLEと呼ばれるシステムを開発している[Eshelman86]．彼らは，開発するエキスパートシステムがまだ試作されていない時点での知識獲得作業を静的解析(static analysis)，試作システムを使用中に問題が生じたとき行う作業を動的解析(dynamic analysis)と呼んでいる．静的解析と動的解析の違いは，専門家から知識を獲得するための手掛りの見つけ方にある．静的解析においては，その時点での知識ベースを解析して手掛りを得る．一方動的解析では，試作システムを使用中に生じた問題点(診断型エキスパートシステムの場合は診断の誤り)が手掛りとなる．したがって問題が生じない限り何もしないため，動的解析はその意味で受動的といえる．

注意しなければならないことは，静的解析と動的解析の違いは知識獲得のための手掛りの得方だけであって，それが得られた後の処理には本質的な違いはないということである．もし，動的解析時に得られるような手掛りが静的解析時に得られたならば，当然それをもとに知識獲得をできねばならない．

なお，経験を持たない発注者が独力でデータベース構築を試みる場合は，これまでの議論に加えさらにデータベース管理システム（以下DBMSと略す）にまつわる様々な困難を克服しなければならない．DBMSはオペレーティングシステムに近い複雑さ持った大規模ソフトウェアである．そのた

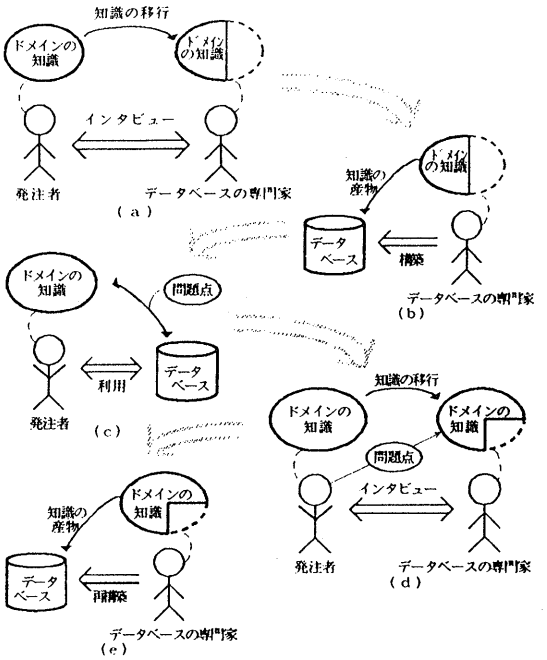

図1　データベースの構築作業

め多くの機能，オプションあるいは制約が存在し，さらに大部なマニャルがそれらの把握を困難なものにしている．本システムはＤＢＭＳの利用に関してマニャルレス化を試みたシステムという性質も持っている．

３．システムの概要

本システムは，図１のデータベースの専門家の役割を果たすシステムである．システムはＩ²Ｓにデータベース構築用知識ベース，ＤＢＭＳとのインタフェースを加えた構成となっており，プロトタイピングの手法を用いてデータベースの構築を支援する．使用するＤＢＭＳとしては関係型のＤＢＭＳのみを対象としている．現在のところ，Prolog で記述した関係型ＤＢＭＳシミュレータを対象としている．このシミュレータはＳＱＬ型のデータ操作言語（以下，ＤＭＬと略す）を処理できる．

プロトタイピングの手法を用いて支援するためには，次のような機能がシステムに要求される．

(1) 論理設計
(2) データベース試作の自動化
(3) 問題点発生の検出
(4) 再設計

本システムでは(1),(4)にＩ²Ｓを使用する．また，プロダクションルール形式で記述された知識を用いて(2)を試みている．(3)については，データベース発注者に自然言語インタフェースを提供し，検索要求文処理中に問題点発生を自動的に検出することを試みている．また，あるデータが存在しないといったデータに起因する検索失敗に関して，その原因を指摘する協調的な応答の生成も同時に試みた．

システムはまず発注者（以後ユーザと呼ぶ）インタビューを実施し，ドメインの知識を抽き出す．この知識はプラン構造と呼ばれる形式でシステム内に蓄えられる．プラン構造からデータベースの論理構造を決定し，使用するＤＢＭＳに合わせてデータベース定義ファイル，初期データ格納用プログラムおよび初期データファイル仕様書を作成する．データベース定義ファイルをＤＢＭＳに渡すことによってデータの入っていないデータベースの枠組が計算機上に作成される．次に仕様書をユーザに提示してデータファイルを作成してもらう．これを先のプログラムでデータベースに格納すると，第１版データベースが完成する．

第１版が完成すると，これをユーザの利用に供する．この段階では，本システムはデータベースの自然言語による検索インタフェースとして機能する．結果が空になる検索失敗に対しては，システムはその原因を指摘する応答[Kaplan83]の生成を試みる．一方ユーザが検索しようとした属性や関係がデータベースにないといったデータベースの構造に起因する検索失敗に対しては，その検索要求文をデータベースへの新しい（あるいは潜在していた）要求と見て，修正版の構築に入る．

修正版の構築は，Ｉ²Ｓによる再インタビュー，再論理設計を経て，古いデータベースを新しいデータベースに変換することによって行う．

なお知識獲得の観点からは，第１版構築時のインタビューが静的解析に，また修正版構築時のインタビューが動的解析に当たる．静的解析時の手掛りはプラン構造上で推論を行うことによって得る．一方動的解析時の手掛りは，ユーザが行おうとして失敗した検索の内，データベースの構造に起因するものの検索要求文である．

４．構築支援

Ｉ²Ｓは，構築するデータベースでユーザが処理したいと考えている検索要求文を手掛りとしてユーザにインタビューを実施し，プラン構造と呼ばれるドメインのモデルを構築する．なお，本稿では紙面の都合上その詳細について[川口86]は省略する．．図２にシステムとの対話例を示す．以下に示す例は，すべてこの対話例に対応するものである．

図３はプラン構造の例で，図２に示した対話で構築されるものである．プラン構造では，現実世界における個々の活動が動詞フレームで，また各実体が名詞フレームで表される．動詞フレームには action(その活動が持つ意味)，goal(その

```
1) Please enter the name of your database.
   > makerDB.
2) Now, let's start designing conceptual schema
   of makerDB.
   Please enter an information requirement.
   > I want to know the names of parts which
   are constructed fromo p325.
   Is 'part' the name of a instance ? > No.
   Please enter the class name of 'p325' > Part.
   Do you want to know the names of part-1's
   which are constructed from part-2's ? > Yes.
3) Please enter the next information requirement.
   > Find the names and quantities of parts from
   which p413 is constructed and which are
   supplied from KL Ltd.
   Is 'p413' the name of a instance ? > Yes.
   Please enter the class name of 'p413' > Part.
   Please enter the class name of 'KL_Ltd'
   > Company.
   Do you want to know the names and
   quantityB's of part-2's from which one part-1
   is constructed and which are supplied from
   company-1's ? > Yes.
4) Please enter the next information requirement.
   > I want to know the period necessary for
   supplying p320's.
   Please enter the class name of 'p320' > Part.
   Do you want to know the period B necessary
   for supplying C units of part-1's ? > Yes.
   What does the period B mean, 1) period for
   transporting part-1's or 2) period for making
   and transporting part-1's ? > 2.
   I can't discuss this information further since
   time is too complicated for me to deal with.
   Please contact a human expert.
5) Please enter the next information requirement.
   > No more.
6) Wait a moment. I think you should store the
   information about
        'stock'.
   Do you agree ? > Yes.
   Please enter information requirement about
   'stock'.
   > Some parts are stocked.
   Do you want to store the information about
   the fact "B units of part-1's are stocked" ?
   > Yes.
7) Wait a moment.
   Let's discuss the following queries:
   Q1:I want to know the names of part-1's which
      are constructed from other part-2's.
   Q2:I want to know the names and quantityB's
      of part-2's from which one part-1 is
      constructed and which are supplied from
      other_company-1.
```

図２　インタビューの例

I think you should store the information about the following fact:
F:One part-1 is constructed from C units of part-2's.
This fact corresponds to the following query:
Q:I want to know the names and quantityC's of part-2's from which part-1 is constructed.
Is it right ? > Yes.

8) Let's discuss the following facts:
F1:One part-1 is constructed from B units of part-2's.
Is the next fact true ?
F:One part-2 is constructed from C units of part-3's.
Please choose one from the followings:
1) sometimes true. 2) always fail. > 1.

9) Let's discuss the following facts:
F1:One part-1 is constructed from B units of part-2's.
F2:C units of part-2's are supplied in order to construct one part-1.
The following fact is true about F1:
F:One part-2 is constructed from C units of part-3's.
Is part-2 in F2 supplied when F is not true ?
> Yes.

10) Let's discuss the following facts:
F1:One part-1 is constructed from B units of part-2's.
F4:C units of part-1's are stocked.
Is part-1 in F1 constructed from part-1 in F4 ?
> Yes.

11) Let's discuss the following facts:
F2:B units of part-2's are supplied in order to construct one part-1.
F4:C units of part-1's are stocked.
If C units of part-2's in F2 are stocked,
is part-2 supplied ? > No.

13) Let's discuss the following noun:
N:Part.
I hink it's one of the physicalObject.
Is it right ? > Yes.
Please choose the attributes of the part from:
1.size 2.weight 3.color 4.price 5.others)
> 2, 4.

13) Ok, I start designing conceptual schema. Wait a moment, please.

図2 (つづき)

活動が達成する状態)，precondition(その活動を行うための前提条件)が１０個の意味素を用いて記述してある．この例の場合，ある製造業の会社における組立て(construct)，在庫(stock)および材料となる部品の別会社からの納入(supply_from)という三つの活動が表現されている．

この知識から次のようにして構築するデータベースの論理構造を得る．ただしここでいう「関係」は，関係型データベースと同じ意味での関係（いわゆる「表」）である．また論理構造の表現として，実体－関連モデル［Chen76］を用いている．

(1) 動詞フレームのすべてについて
フレーム名 →関係名
各スロットの値→その関係の属性
として，その動詞に関する関係(関連)を定義する．

(2) すべての検索主体(selectEntity)を表す名詞フレーム(これはContentフレームのselectEntityスロットで示される)について
主体名→関係名
attributes スロットの値→その関係の名前
として，その主体に関する関係(実体)を定義する．

(3) 各関連について，関連する実体間の対応関係(1:1, 1:n, m:n)をユーザに質問して決定する．

(4) 各関係のキー属性を決定する．動詞に関する関係については change_how_many, change_where, change_state という状態の変化を示す意味素に関係する属性以外のすべての属性をキー属性とする．主体に関する関係については，ユーザに質問して決定する．

(5) 各属性についてデータ型を決定する．value_example スロットが存在するときは，その値例から文字型，整数型あるいは浮動小数点型のいずれかとする．存在しないときは文字型とする．

以上のようにして図４に示す論理設計結果が得られる．図中，ent, rel がそれぞれ実体（名詞フレーム）と関連（動詞フレーム）に関する関係を示している．ここで両者の第２，３引数は，それぞれ関連する実体(entの場合はnil)および属性を示している．また，key が各関係のキー属性を，type が各属性のデータ型を表している．また hie は実体間の対応関係を表しており，第１引数が関連名，第２，３引数が実体名である．

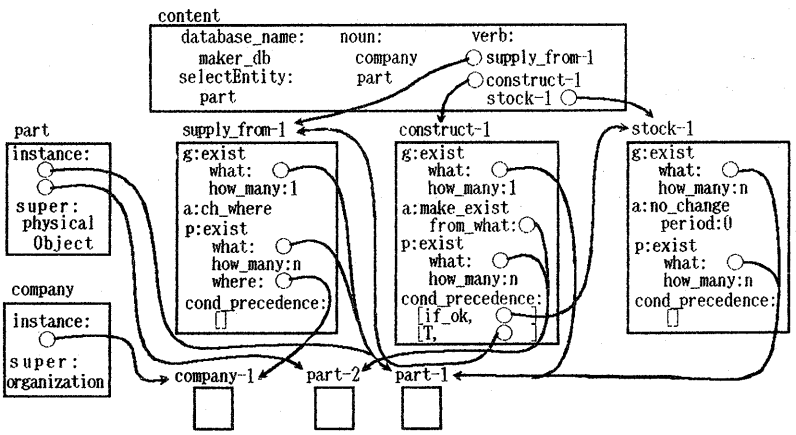

図3 プラン構造の例

```
selEnt(part,[],[name]).
ent(company,[],[name]).
rel(construct,[part(1),part(2)],[qty]).
rel(stock,[part],[qty]).
rel(supply_from,[part,company],[]).
key(part,[name]).
key(construct,[part(1),part(2)]).
key(stock,[part]).
key(supply_from,[part,company]).
hie(construct,part(1),part(2),'m:n').
hie(stock,part,[],'1:1').
hie(supply_from,part,company,'m:n').
type(part,name,char).
type(company,name,char).
type(stock,qty,int).
type(construct,qty,int).
type(supply_from,company,char).
```

図4　論理設計結果

なお，設計結果は関係データベースにおける第１正規形で，より高次の正規形へ分解できる余地がある．また対応関係，キー属性の決定などに関しては，システムからの質問が非常に多い．これらはドメインに関する知識を表したプラン構造の表現力とI^2Sが持つ知識による問題であり，今後検討を加える予定である．

この論理設計結果にプロダクションルール形式で記述された知識を適用し，データベース定義ファイルを作成する．図５に作成されたデータベース定義の例を示す．table，attributes,key に続く名前がそれぞれ関係名，属性名およびキー属性である．またchar_not_null は，その属性のデータ型が「空でない文字列」であることを示しており，データベースの一貫性保持に用いられる．このデータベース定義をＤＢＭＳに渡すと，データがまだ入っていないデータベース（の枠組）が計算機上に作られる．

```
database_definition
    table part
        attributes
            name: char_not_null
        key name
    table stock
        attributes
            part: char_not_null
            qty: int
        key part
    table construct
        attributes
            part_1: char_not_null
            part_2: char_not_null
            qty: int
        key part_1, part_2
    table supply_from
        attributes
            part: char_not_null
            company: char_not_null
        key part, company
end_database_definition.
```

図5　データベース定義

なお，このようなデータベース定義の作成は使用するＤＢＭＳに大きく依存する作業であり，ユーザが独力で試みる場合の最初の難関となるものである．本システムは使用するＤＢＭＳごとに必要な知識を用意しておくことにより，この作業を自動化している．

次に論理設計結果から初期データ格納用プログラムおよびデータファイル仕様書を作成する．本システムは両者の作成もプロダクションルール形式で記述された知識を用いて行う．なお，ここで作成するプログラムは，ＳＱＬ文を埋めこんだ Prolog プログラムである．（このようなデータ格納用のユーティリティを持つＤＢＭＳも存在する．この場合は，この機能は特に必要ではない．）

この作業では，格納用プログラムとファイルの仕様との間の一貫性維持ファイル作成時のユーザの負担軽減が課題となる．一貫性を維持するために両者を作成するためのルールは条件部を共有しており，並行して作成を行う．

データファイルを作成する際のユーザの負担を出来るだけ小さくするため，次のような原則にしたがってデータファイルのフォーマットは決定される．

```
load(DataFile) :-
        see(DataFile),
        repeat,
                readln(Part),
                partRead(Part),
        next,
        abolish(next,0),
        repeat,
                readln(Part1),
                constructRead(Part1),
        next,
        seen,
        !.
partRead([]) :-
        asserta(next),
        !.
partRead(Part) :-
        execSQL(insert_into part: (Part)),
        readln(Quantity),
        stockRead(Part,Quantity),
        repeat,
                readln(Company),
                supply_fromRead(Part,Company),
        next,
        abolish(next,0),
        !.
stockRead(_,[]) :-
        !.
stockRead(Part,Quantity) :-
        execSQL(insert_into stock: (Part,Quantity)).
supply_fromRead(_,[]) :-
        !.
supply_fromRead(Part,Company) :-
        execSQL(insert_into supply_from: (Part,Company)).
constructRead([]) :-
        asserta(next),
        !.
constructRead(Part1) :-
        repeat,
                readln(Part2),
                constructRead2(Part1,Part2),
        next,
        abolish(next,0),
        !.
constructRead2(_,[]) :-
        asserta(next),
        !.
constructRead2(Part1,Part2) :-
        readln(Quantity),
        execSQL(insert_into construct: (Part1,Part2,Quantity)).
```

図6　データ格納用プログラム

```
You have 4 relations, "part", "construct", "stock" and "supply_from".
Please make a datafile in the format as shown bellow:

    name (char)             ; name of part (char)
    quantity (int)          ; quantity of the part if it is stocked, or <nl>
    company (char)          ; name of company if the part is
                              supplied from the company, or <nl>
        :
        :
      < repeat company name >
        :
        :
    <nl>                    ; < the delimiter for company name >
/* Here, 1 block for "part", "stock" and "supply_from"
   ends. Put <nl> to show the end of blocks, or repeat
   another block in the same format as shown above.
*/
    part_1 (char)           ; name of a part which is constructed
    part_2 (char)           ; name of a part from which the
                              part_1 is constructed
    quantity (int)          ; quantity of part_2
        :
        :
      < repeat part_2 and quantity >
        :
        :
    <nl>                    ; < the delimiter for part_2 and quantity >
/* Here, 1 block for "construct" ends. Put <nl> to show
   the end of file, or repeat another block in the same
   format as shown above.
*/
```

図7　データファイルフォーマット仕様書

(1) 関係する関連の数が最も多い実体から順にデータを格納する．

(2) ある実体について一組のデータを入力するたびに，関係する関連の内で入力可能なデータをすべて格納する．

(3) ある関連に関して実体間に'1:n'の対応関係があるときは，n側の実体から先にデータを格納する．

ここで(1)，(2)はデータファイル作成時のユーザの見通しをよくするためで，(3)はデータファイルの量的効率を高めるためである．

こうしてデータ格納用プログラムとデータファイル仕様書が作成される．それぞれの例を図6，7に示す．プログラムはデータファイル名を引数としてload述語を呼び出すことにより起動される．プログラム中，abolishはarityが第2引数の値と等しく，名前が第1引数と等しい述語を消去する述語である．またreadInはデータファイルから1行読み込み，第1引数に単一化する述語である．ただし，改行コードのみからなる行については'[]'（空リスト）を単一化する．なおプログラム例からわかるように，ファイルフォーマットの誤りに関してその検出，回復を行えるプログラムは，現在生成していない．本システムは経験の少ないユーザを対象としており，そのような機能を持ったプログラムが必要であるので，今後この点については改良する予定である．

ＤＢＭＳによってはデータベースの一貫性維持機構が強力なため、あまり経験のないユーザはデータの依存関係に注意しないとデータを格納できない場合がある。本システムはデータの依存性に注意してプログラムを作成するため、そのような問題をあらかじめ回避していると言える。

仕様書をユーザに提示してデータファイルを作成してもらい，プログラムを用いてデータベースに格納する．このようにしてデータベースの第1版が完成する．

5．修正・利用支援

第1版（あるいは第2版以降）のデータベース（以下，試作データベースと呼ぶ）が完成すると，修正・利用支援の段階に入る．この段階では，自然言語による検索機能をユーザに提供して，

・データベースの構造に起因する問題発生の検出

・検索結果が空となる検索失敗時の，協調的応答の生成

を試みる．

試作データベースの問題点検出は，ユーザに特別な負担をかけることなく行えることが望ましい．そこで本システムでは，ユーザに自然言語による検索インタフェースを通して試作データベースを利用してもらい，同時にその際入力される検索要求文を監視することにした．このような検索インタフェースは，ＤＭＬを覚える必要がなくなるのでユーザにとっても有益であろう．試作データベースに対して不適切な検索要求文が入力されたときは，その検索要求文自体を構築すべきデータベースへの要求と考える．また検索結果が空になるときは，ユーザが想定しているデータと実際にデータベースに格納されているデータの間に隔りがあるので，データレベルでの試作データベースの修正を試みる．

処理の流れを図8に示す．ユーザが入力した検索要求文は，一旦プラン構造表現に変換される（以下，これを検索文のプラン表現と呼ぶ）．このプラン表現は，システム内に蓄えられている試作データベースに対応するプラン構造と比較される．同義語の処理もこのとき行う．検索文のプラン表現がプラン構造に適合しているときは，検索処理に入る．一方不適合の時は検索要求文が想定しているデータベースの構造に誤りがあるので，不適合箇所と検索文のプラン表現をI^2Sに引き渡し，修正版のためのインタビューに入る．

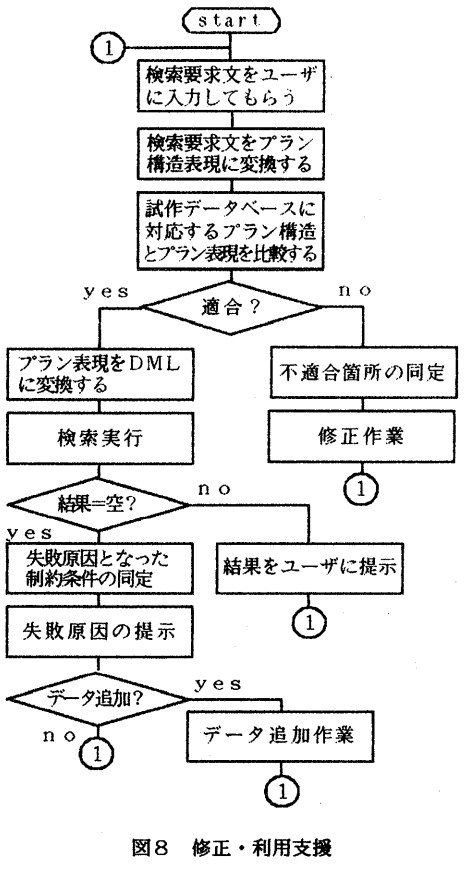

図8　修正・利用支援

検索処理では，使用しているＤＢＭＳのＤＭＬに検索文のプラン表現を変換し，まず実際に検索を行う．検索結果が空でないときは，検索が成功したと考え結果をユーザに提示する．一方検索に失敗したときは，失敗原因を同定するために検索の制約条件を緩和しながら検索を繰り返す．この過程で結果が空にならなかったときに緩和されていた条件に対応するデータが，試作データベースに存在していない（データの失敗）ことがわかる．データの失敗に対しては，ユーザに二つの選択肢が与えられる．１つは利用を一旦中断して，データの追加作業を行うというものである．もう１つは，検索要求文の入力に戻ることである．

５．１　検索文のプラン表現とプラン構造の比較

検索文のプラン表現は検索要求文が想定しているドメインをプラン構造表現で表している，すなわちユーザが想定しているデータベースの構造を表している．一方，プラン構造は試作データベースの実際の構造に対応している．両者の不適合としては，以下のようなものが存在する．

(a) 検索文のプラン表現内のあるフレームがプラン構造内に存在しない．
(b) 検索文のプラン表現内のあるフレームのあるスロットが，プラン構造内で対応すると考えられるフレームに存在しない．
(c) 検索文のプラン表現内でフレーム名，スロット名が対応するフレームがプラン構造内に存在するが，スロットの値が異なる．

(a)はユーザが存在していると考えた関係が，実際には試作データベースに存在しない，また(b)は同様にある属性が存在しないことに対応する．一方(c)は，プラン構造におけるスロットの値のほとんどが他のフレームへのポインタなので，ドメインでの（システムにとって）未知の論理的つながりを示していると考えられる．

動詞フレームについては，同義語についても正しく処理できるように意味素レベルで比較を行う．一方名詞フレームについては名前で比較する．この場合は，同義語の場合があるので，名前が一致しないときは必ずユーザに問い合わせる．

５．２　修正作業

データベースの修正は，図９に示す順で行う．先ず，比較の結果とそのときの検索要求文（のプラン表現）をもとにユーザにインタビューを実施する．この結果，修正版に対応する新プラン構造が得られ，これより修正版の論理構造が決定，修正版の枠組が計算機上に作られる．また第１版構築時と同様に，プロダクションルール形式で記述された知識を用いて追加データファイル仕様書およびデータベース修正プログラムを作成する．そして図に示すようにこのプログラムを用いて旧版のデータベースから修正版へデータを移しながら，追加データを修正版に格納していくことによって修正版データベースが構築される．

この段階での論理設計は，インタビューの手掛りとして問題を生じた検索要求文を用いる．２節で述べたように，手掛りが得られた後では静的解析と動的解析には本質的な違いはない．本システムでもインタビューは，第１版構築時とまったく同様の戦略を用いて行なわれる．結果として得られる修正版の論理設計結果は，第１版構築時に問題となった検索要求文も手掛りとしてシステムに入力した場合と同じになる．

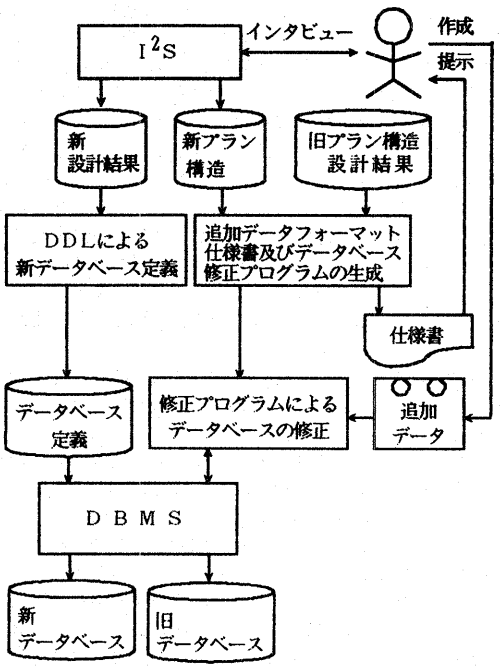

図９　修正作業

旧版から修正版へデータを移していくという方法は，データベースが大規模になると著しく効率の悪いものとなることが予想される．またＤＢＭＳの中には論理構造を後から変更する機能を持ったものも存在する．しかし，本システムではデータベース修正プログラムが複雑になるのを避けるため，このような機能は使用していない．本システムの主たる用途は，個人レベルでの小規模なデータベースの構築・利用支援であり，当面問題はないと考えている．なお，現在のところ修正のパターンとしてある関係への属性の追加と新しい関係の追加のみを考慮しており，大幅な論理構造の修正については検討課題となっている．

５．３　利用支援

検索要求文（のプラン表現）がデータベースの論理構造（プラン構造）に適合していると，検索文のプラン表現をさらにＤＭＬ表現に変換していよいよ実際に検索を行う．検索結果が空でなければユーザに提示して再び検索要求文の入力に戻る．結果が空になったときは，検索失敗と考え，CO-OP [Kaplan83] の Corrective Indirect Response と同様の応答生成を試みる．この機能は先に述べたデータ失敗の検出あるいはユーザのデータベースに対する誤解の訂正を目指したものである。

CO-OP は検索インタフェースの移植性を重視していたので，データベースの論理構造と使用する自然言語自体の表層の知識のみを用いて，ＭＱＬと呼ぶ中間表現上で協調的応答の生成を試みていた．本システムでは，インタビューによってユーザから得たドメインに関する知識がすでにプラン構造という形でシステム内に存在しているので，これの活用を試みている．

検索結果が空となるのは，検索要求文の制約条件を満たすデータが存在していないときである．CO-OP のＭＱＬは検索の対象となる集合をノード，制約をリンクとするグラフである．検索失敗の原因は，ノードごとに検索を行なうことによって同定できる．例えば結果が空となるノードが存在するときは，そのノードに対応する条件が原因である．またあるリンクに関してjoinをとったときに結果が空となるときは，そのリンクに関する条件（そのリンクで結ばれたノードに対応する条件のＡＮＤ）が原因である．本システムの検索要求文のプラン表現は，制約条件を（属性，値，属性と値の関係）という３つ組の集合で表している．そこでこの集合から３つ組を取り除くことによってCO-OPと同様の制約条件の緩和を行う．

CO-OP ではドメインの知識に依存する意味処理を一切行わないため，ＭＱＬ上のすべてのノードについて調べ，Corrective Indirect Response を生成する．一方本システムの中間表現はプランの概念を含んでいるので，より大きな前提となっている制約条件から順に緩和していき，原因を同定する．データを生じる活動の流れは常に「precondition → goal」の向きなので，要求するデータに対して，そのpreconditionが成立していなければデータを生じない（したがってデータベースにも存在しない）．そこで検索結果が空になったときは，precondition が満たされていないことを疑う．具体的には求めるデータから最も遠い precondition（最も大きな前提）から順に制約を緩和していき，原因を探す．

この処理の利点は，制約条件の重要度についての順序付けをできるため，原因の同定をある程度効率的にできる場合があることである．ただしこのような処理がいつでも意味を持つかについては若干の疑問がある．プラン構造の利用支援における活用についてはまだまだ議論の余地が残されており，現在検討を進めている所である．

６．検討および課題

データベースのマン・マシンインタフェースとしての本システムの特徴は，以下の４点である．

(a) 論理設計から試作，利用までをマニュアルレス化している．
(b) Domain-Specific Knowledge をユーザが記述する必要はない（インタビューに答えればよい）．
(c) Corrective Indirect Response を生成できる．
(d) 同義語について学習機能を持つ．

一方，知識獲得支援システムの観点からは，次のような特徴が挙げられる．

(e) ユーザは検索要求文を用意するだけでよい．
(f) データベースの試作・利用支援機能を通じて動的解析を実現している．

現在の設計では，システムは次のような限界を有している．まず処理できる検索要求文に関して，

(A) 解析できる検索要求文は，"Find", "I want to know" で始まる命令文のみ（論理設計時にはこれに平叙文が加わる）である．
(B) 重文，複文は処理を誤る場合がある（多い）．
(C) 省略，照応などには対処できない（文脈処理を行っていない）．

また構築するデータベースに関して，

(D) 汎用ＤＢＭＳの使用を前提としており，図形データなどを対象としたデータベースは構築できない．
(E) 利用支援に関しては，対話型処理のみを対象としている．

実用化に向けては，つぎのような課題が挙げられる．

(1) Ｉ²Ｓの動詞その他の辞書の充実
(2) パーサの強化
(3) 汎用ＤＢＭＳとの結合と，ＤＢＭＳ操作に関連するエラーの回復機構の開発

一方，今後の研究課題としては，

(ｱ) プラン構造の応答生成時における活用
(ｲ) プラン構造で記述されている知識のパージング時における活用
(ｳ) 論理設計における正規形分解の試み

などを考えている．また，既存のデータベースを手掛りとして開発者にインタビューを実施し，自然言語インタフェースを自動生成するシステムもインタビューおよびデータベースのマン・マシンインタフェース両方の観点から興味深いのではないかと考えている．

７．むすび

本稿では，論理設計からデータベースの試作，利用まで総合的に支援するデータベース構築・利用支援システムについて述べた．今後は実用化を意識しながらさらに検討を進めていく予定である．なお，本システムは横河ヒューレットパッカード社の HP9000 model320 上で C_Prolog を用いて開発中である．

参考文献：

[磯本84] 磯本征雄，「データベース構築・管理支援システムKDBMS使用の手引き」，大阪大学大型計算機センターニュース，Vol.13，No.4，1984.

[川口86] 川口他，「データベースの論理設計を支援する知的インタビューシステム」，知識工学と人工知能48-1，1981.

[Chen76] Chen, P. P., "The Entity-Relationship Model Toward a Unified View of Data," ACM Transaction of Database Systems, Vol.1, No.1,1986.

[Eshelman86] Eshelman, L. and McDermott, J., "MOLE: A Knowledge Acquisition Tool That Uses Its Head," Proc. AAAI'86, 1986.

[Kaplan83] Kaplan,S.J., "Cooperrative Responses from a Portable Natural Language Database Query System," Brady, M. and Berwick, R.(eds.), Computational Models of Discourse, The MIT Press, 1983.

[Mizoguchi85] Mizoguchi, R., et al., "Interactive Synthesis of Conceptual Schema Based on Queries -Towards an expert system of relational database design-," Journal of INFORMATION PROCESSING, Information Processing Society of Japan, Vol.8, No.3, 1985.