データベース・システム 49-3
（１９８５.９.２０）

Prologによる知識データベースシステム

住友電気工業（株）　松尾正信
silogic社　D. Stott Parker, Jr.
Paul Eggert

1. はじめに

データベース研究は、１９７０年代から関係データモデルを中心に数学的概念にもとづく研究がさかんになり、データベース理論として今日におよんでいる。その背景には、各企業のＤＰ部門におけるビジネスアプリケーションのほとんどがデータベース構築を必要としており、そのための設計及び開発のコスト、パーフォーマンス、信頼性が大きな問題となっていたからである。　その現状は現在も変っていないが、１９８０年代に入って、ワークステーション，マイクロコンピュータの普及によって、コンピュータを利用しようとする分野が急激にひろがり、新しい形のアプリケーションの要求がいたるところで生まれてきている。そして、人工知能の研究成果を、そのアプリケーションに応用しようとする試みが、ここ１，２年で急速にすすみつつある。　その中の一つに、知識ベース・システム、又は、ひろくエキスパート・システムとよばれるものがある。　たとえば、設計や製造技術の中には、専門家の長い経験からつちかわれたノウハウや、知識があり，それらの知識の一部をコンピュータの中に入れて、適切な判断を行なったり、問題解決策を導きだすようなシステムが試行されている。

知識ベース・システムの基礎技術として最近、ロジック・プログラミングとデータベースとの融合をめざした研究がすすめられている〔Gallaire 78，81，Kowalski 81〕。ロジック・プログラミング技術をデータベースに応用することによって，演繹機能、スキーマとデータとの結合、制約条件記述が同じワク組の中で可能である。　View，検索言語，null value，のとり扱い方にも統一的なアプローチが可能となる。　一方、ロジック・プログラミング・システムとして代表的なPrologシステムでは、すべてのデータは実行の前にメモリ（仮想メモリの場合もあるが）にロードされる。　したがってOSがページングをサポートしないかぎり、外2次記憶媒体は、最新のデータを貯えたり、検索したりするためには利用されない。

次の二つのアプローチとそれにともなう問題が考えられる。

① データベースアプリケーション向きにロジック・プログラミングを拡張すること。　データベース検索用のインタフェースをどのように追加するのか。　知識表現，データベーススキーマ定義，不完全なデータ等をどのように扱うのか。　トランザクション，マルチユーザのサポートをどうするのか。

② ロジック・プログラミングにデータベースの２次記憶域の管理と検索機能を追加すること。　どんなデータベース・アーキテクチュアがよいか。検索の最適化はどうするのか。

この論文では、②の方向をめざして研究開発をすすめてきたSilogic社の、「Knowledge WorkBench*（KWB）」について説明する。

②を実現するために次の三つのアプローチが考えられる。

Ⓐ 既存のデータベース・システムに強力な推論機能を追加する方法。[Stonebraker 84]

Ⓑ 既存のデータベース・システムとロジック・プログラミング・システムの結合[Chang 85]。

Ⓒ ロジック・プログラミング・システムにデータベース機能を追加する方法[Chomicki 88]。

Ⓐでは、Prologの推論機能をカバーしきれていない。抽象データ型を検索言語に追加する方法は、ホーン節の論理式とかなりの距離がある。
Ⓒでは、データベース・システムの大部分を作りなおす必要がある。
Ⓑでは、既存の二つのシステムを結合する時の不合致が問題となる。
KWBは上記の三つアーキテクチュアをうまく融合することをめざしている。
以下の特徴を持つ。

- 知識ベース・システムの入出力方法として自然言語（英語又が日本語）を採用しており、自然言語処理部によって論理的内部形式に変換される。
- 知識ベース管理システム部（KBMS）は、知識ルールと事実を非二次記憶媒体に貯えて、index等のアクセス方式を採用している。
- 既存のDBMS（unify**）とのインタフェースを持ち、スキーマ記述と自然言語との結合はKBMSの中に、ルール形式で表現されている。
- 開発言語としてprologコンパイラを持つprologを採用している。

2. KWBの基本構成

KWBの基本構成を図1に示す。　KWBへの自然言語による入力は、検索表現、アサーション，コマンドであり、出力は、表形式又は自然言語による応答である。　自然言語パーサは英語（日本語）をSilogとよぶ内部形式に変換する。　KBMSは、推論機能を持ち、検索を実行するとともに、アサーションの矛盾をチェックする。　したがってIntegrity Constraintの一般化が行なわれている。　データ，知識ルールは、MaxwellのDBに貯えられており、Logbaseによって外部のDBMSへのアクセスも可能である。

2.1. 自然言語処理部

KWBのユーザが表現する自然言語は、事実、ルールが正確で、あいまいさのないことが意図されているので、機械翻訳やワードプロセッサの入力文とは、性格が異なる。　自然言語によるロジック・プログラミングという表現に近いであろう。

---

* Knowledge WorkBench は Silogic社の登録商標です。
** unify は unify社の登録商標です。

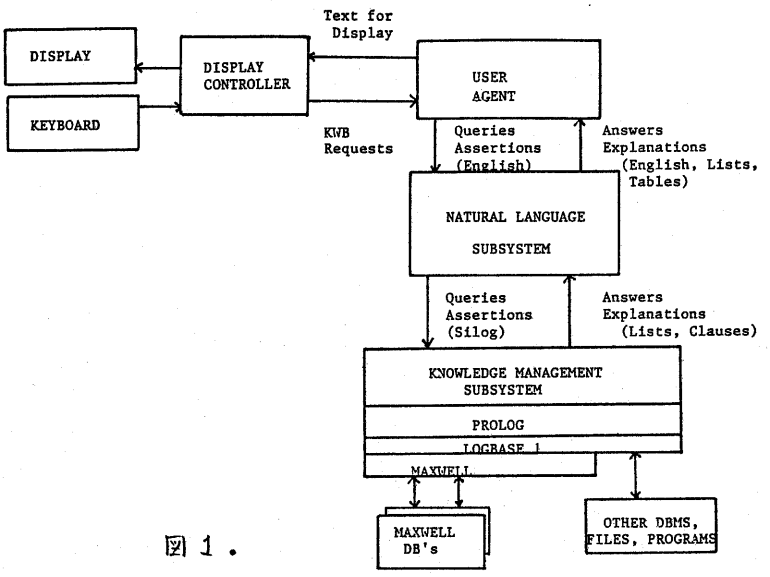

図1.

言語学研究の成果として、チョムスキーのGB (Government and Binding) 理論がよく知られているが、その中でもXバー理論，痕跡理論，等にもとづく文法を開発しパーサの中にくみこんでいる。パーサは次の三つのステップにわけられる。

①. 前処理 : 形態素解析を行ない、イディオムをみつけ、leaf node を作る。

② NPパス : この rewrite パスは noun phrase と prepositional phrase を作成する。

③ Sパス : clause と sentence を作成する。再帰的処理が行なわれる。このパスの出力は，functional structure とよぶ入力文の論理的表現形式である。

パーサは、ボトムアップとトップダウンの両方を使いわけていて，再帰的，shift-reduce，文脈依存型の、rewrite システムである。パーサ・ルールはルールコンパイラによって効率のよい形式に変換される。

文のもつあいまいさは、ユーザとの対話によって解消される。

2.2. 概念モデル

KWBの概念モデルは、Entity-Relationship モデルの拡張型 [Matsuo 84] に近い。Entity は，物，事柄とみなせるものなら何でもよい。たとえば，車，人，会社である。Entity は、0または1つ以上の name を持つ。各 name は，一意的に entity をきめる。Entity は Type (型) を持つ。sub-type, supertype とそれにともなう Inheritance も表現できる。たとえば，

All men are people. (Type : man は type : person の subtype である)
All people are mortal. (man は person の性質である mortal を継承している)

したがって，John is a man. という事実から，John is mortal が導かれる。

Entity は、named entity（たとえば"太郎"）と、skolemized entity（たとえば、太郎の妻）との区別がある。

Relation としては、次のような区別が考えられる。

- 動詞による relation（たとえば、John likes Mary）。
- 前置詞による relation（The book is on the table）.
- Role による relation（John is Joe's father）。
- Attribute。 attribute は、entity の性質を value で説明する。たとえば、The book is heavy.

KWB では、次のような同値表現が可能である。

_X weighs _Y : _X's weight is _Y.

A book _X is heavy : _X weighs more than 10 pounds.

この場合、heavy というのは、明確な定義を持つこととなる。

_X's profits = _X's income - _X's expenses といった計算式も可能である。

量記号（quantifier）は、"the", "a", "all", "both", "every", "each", "any", "few", "several", "many", "most", "some" および "5", "10" といった数字を扱うことができる。

2.3. 演繹（Deduction）

KWB の演繹的（deductive）システムについて述べる。ルールは、1つ以上の type のすべての entity についての情報とみなせる。同じ relation について複数のルールを記述できる。たとえば、

if _X is _Y's father then _X is _Y's parent.

_X is _Y's parent if _X is Y's mother.

2つ以上のステップを必要とする演繹を行なうことができる。

recursion, transitive relation, symmetric relation, 否定を扱うことができる。

2.4. 検索（Query）

3つの質問形式がある。"Yes/No" 形の質問に対して、Yes, No, Unknown の3つの答がかえされる。"wh" 形の質問は、type や、entity の attribute についての質問である。たとえば、

what is the closing price of IBM ?

which stocks does Richard own ?

"How-many" 形の質問の例としては、

How many children does John have? がある。

"How-Many"形の質問に対する答として、下限と上限の2つの部分をふくむ。
下限は、質問をみたすentityの最小数を示し、上限は、最大数を示す。
たとえば、
Every man loves some woman.
John is a man.
Jerry is a man.
に対して、How many women are there? と質問すると、下限1人，上限2人となる。つまりJohnとJerryは同じ女性が好きなのか別な女性が好きなのかわからないからである。

2.5. 知識表現

知識の論理的内部表現はSilogとよばれ次のような形を持つ。

FOR(QUANT--∀:TYPE! CONSTRAINT, PREDICATE)

たとえば、「Every man loves some woman」は次のようになる。

FOR(ALL--M:MAN, FOR(SOME--W:WOMAN, LOVES(M,W)))

その他、time, modal, subject, object 等のcase strudureが追加されている。

2.6. 知識ベース管理システム（KBMS）

従来のDBMSは簡単な事実のみを貯えているのに対して、KBMSは事実とルールの両方を貯えている。下記にその例を示す。（KWBのシンタックスを使う）

```
Facts:  records.
  /*      name              position            company     totalsalary */
  emp('Barry Diller',    'Vice President',  'Gulf &.Western', 2122076).
  emp('John Gutfreund',   'Cochairman',      'Phibro-Salomon', 2110000).
  emp('David Tendler',    'Cochairman',      'Phibro-Salomon', 2080000).
  emp('Henry Kaufman',    'Vice President', 'Phibro-Salomon', 1950000).
  emp('Richard Schmeelk', 'Vice President', 'Phibro-Salomon', 1950000).
  emp('Thomas Strauss',   'Vice President', 'Phibro-Salomon', 1950000).
  emp('Peter Buchanan',   'President',       'First Boston',   1716565).
  emp('Rawleigh Warner',  'Chairman',        'Mobil',          1644083).
  emp('Alvin Shoemaker',  'Chairman',        'First Boston',   1600572).
Rules:  if-then statements
  outrageously_paid(X) :- emp(X,_,_,Salary), Salary ) 2000000.
  outrageously_paid(X) :- emp(X,_,'Phibro-Salomon',_).
  credit_risk(X,modest) :- outrageously_paid(X).
```

KBMSの中のMaxwellでは、prolog節は第2次記憶媒体に入力される。
Maxwellの特徴としては、下記のようなものがある。

- トランザクションのネスト
- パターンによる検索
- 一般的なアクセス・パス

DBMSと既存の知識ベースとMaxwellとの機能比較を図2に示す。

Feature	Databases	Knowledge Bases	Maxwell
simple facts	yes	yes	yes
complex facts	no	yes	yes
rules	no	yes	yes
secondary storage	yes	no	yes
indexing	yes	no	yes
aggregate computation (average, min, max...)	yes	no	yes
range-retrieval	yes	no	yes
selective retrieval	yes	no	yes
recovery facilities	yes	no	yes
query optimization	yes	no	yes
transactions	yes	no	yes
privacy & security of data	yes	no	planned
compressed storage formats	yes	no	planned
multiple user support	yes	no	planned
'what if' processing	no	yes	yes
programming environment	no	yes	yes [Prolog]
triggers (demons)	no	yes	yes [Prolog]
workspace/temporary results	yes	yes	yes [Prolog]
access to foreign DBMS	no	no	yes [Logbase1]
data dictionary	yes	yes	yes [Logbase1]
complex schema modeling	no	yes	yes [KWB]
natural language input	no	yes	yes [KWB]

図2

2.5.1. トランザクションのネスト

トランザクションのネストとdelayed writeによって、"what-if"機能が可能となる。トランザクションは、データベースの概念である。つまり、beginで、トランザクションがはじまり、writeをふくむオペレーションが実行されるが、abortによってすべてのwriteが無効になるか、commitによってすべてのオペレーション実行が「真」になる。一連のオペレーションが1つのオペレーションかのようにみなされる。Delayed writeというのは、トランザクションがcommitされた時のみ知識ベースに書きこまれることである。
たとえば、「すべての会長のボーナスを10%ふやせ」という例では、

```
? - tbegin,
    forall( detract(emp(X, 'Chairman', C, Sal, Bonus, _)),
            ( NewBonus is Bonus * 1.10,
              NewSal is S + NewBonus,
              dessert(emp(X, 'Chairman', C, Sal, NewBonus, NewSal)))
          ),
    tcommit.
```

図3

```
Pattern   ::= Functor( Argument )
          ::= Functor( Argument, Argument... )

Argument  ::= Pattern
          ::= Variable
          ::= constant    (integer, real, or string)
          ::= ?: Template
          ::= Variable ?: Template

Template  ::= Range
          ::= [ Range, Range... ]
          ::= RegExpr
          ::= [ RegExpr, RegExpr... ]

Range     ::= Bound
          ::= Bound - Bound

Bound     ::= numeric     (integer or real)
          ::= Variable    (specifies open range)

RegExpr   ::= string
```

2．5．2．パターンによる検索

Maxwell の unification は図3に示すような表現式をゆるす。つまり特別のパターン・マッチングオペレータ ':' を持つ。たとえば、

```
?- emp( Name ?:'Jones', _,_,_,_, Salary ).
```

emp/6 の最初のアーギュメントに 'Jones' が一部あるものが探ばれる。UNIX** の ed で使われる ^, $, * が、同様の目的で使われる。たとえば、

```
?- emp( A?:'Jones$', ?:'^Cha', _,_,_, D?:[1-1000] ).
```

2．5．3．一般的なアクセス・パス

DBMS では DBA は index を作ったり消去したりできるし、検索時に適切な index を使用する。 KWB では、B-tree と Bucket index を使う。Bucket index とは、key の値によって順序づけられた clause（節）へのポインタのディレクトリである。述語への複数の index，複数フィールドの index key，部分マッチの検索，数値フィールドの領域検索（range），文字フィールドの regular-expression による検索、が可能である。

2．5．4．Aggregate

average，minimum，maximum 等の DBMS で定義されている aggregate オペレータが使用できる。たとえば、「ボーナスを最大 $100,000 を持つ、社長の平均給与を求める」を考えてみる。

```
?- aggregate ( avg, Salary,
                emp(Name, P?:'President',_,_,Bonus?:0-100_000,Salary),
                AvgSalary).
```

3．既存の DBMS とのインタフェース

DBMS へのマッピングは、ルール記述によって行なわれる。たとえば次のような broker，client，transaction の relation を考えてみる。

```
br ( brid, brlast, brfirst, brbrnch)
cl ( clid, cllast, clfirst, clprof )
tr ( trnum, trbrokr, trclid, trtick, trbs )
```

英語 branch をユーザが使うために次のようなルールを記述する。

```
_Br is a branch .if fields ( "br", "brbrnch", _Br )
```

つまり、もし br という relation の brbrnch のフィールドに _Br（prolog の変数）があれば、その _Br で示されている値は、branch の意味を持つ、ということを示す。

** UNIX は Bell 研究所の商標です。

別な例として

-C's investment goal is tax advantage growth if
fields("cl", "clfirst", -C, "clprof", -P) and profHasA(-P).

ここでのprofHasAは別のファイルの中に記述されているPrologプログラムである。
データベースのマッピングルールとそれ以外のルールとが混在している場合も同様に表現できる。 たとえば、次の例のようである。

-B purchased -Se from -C if -B's broker id is Bi and -C's client id is -Ci
and fields("tr", "trbrokr", -Bi, "trclid", -Ci, "trtick", -Se, "trbs", "S").

ここでは「-B's broker id is Bi」を示すために、次のようなルールが必要である。

-B's broker id is Bi if fields("br", "brfirst", -B, "brid", -Bi).

純粋に推論のためのルールの例としては次のようなものがある。

-B's client is -C if -B is a broker and -B sold a security to -C.

ところで、もし'Matsuo'のtupleがbrのrelationにないと仮定しよう。
「Is Matsuo a broker?」という質問に対しては、「I don't know」と答える。しかし次のルールを入れることにより「No」という答がかえる。

-B is not a broker unless -B is a broker.

あるいはopen-world flagを操作することにより、上記のルールを省略することができる。

4. まとめ

DBMSの延長線上にKBMSを位置づけすることにより、今までのデータベース研究がとりくんできたテーマをもう一度KBMSに対して、再考する必要があるであろう。 KWBは、その方向をめざした一つの試行であり、かつ商品である。

参照文献

Chang, C.L. and Walker 'PROSQL', Report RJ4314, IBM Research Lab., San Jose.
Chomicki, J. 'A database support system for Prolog' Proc. Logic programming Workshop, 1983.
Gallaire, H., J. Minker, Logic and Data Bases, Plenm Press, 1978, 1981
Kowalski, R, 'Logic as a Database Language' T.R., Imperial College, 1981
Matsuo, M, Fundional Entity Relationship Model, Ph.D thesis, UCSB, 1984
Stonebraker, M, 'Extending a Relational Interface for Expert Systems Applications', UC Berkeley, 1984.