Prologと関係データベースとの結合システム DB-Prolog†

今中 武†† 上原邦昭††† 豊田順一†††

Prologには，プログラムの大規模化に伴う実行効率の低下，高速集合演算機能などの実用的なデータベース操作機能の欠落などの問題点があることが指摘されている．これらの問題点を解決するために，本論文ではPrologと関係データベースを結合したDB-Prologを提案する．DB-Prologでは，関係データベースとの結合に非評価方式と評価方式と呼ばれる二方式を用いている．非評価方式を用いて結合されているデータベースは，Prologの内部データベースを拡張したものとみなされ，ファクト集合が格納される．このファクト集合は，データベースの検索機能を用いて実現したユニフィケーションによって高速に検索される．評価方式を用いて結合されているデータベースには，数値，文字列データなどが格納され，DB-Prologのシステム組み込み述語を用いて操作される．システム組み込み述語には，データの追加，削除，検索などを行う述語のほかに，既存のデータベースにアクセスするための述語などが用意されている．また，DB-Prologの評価実験を行ったところ，ファクト集合が3,000個を越えると，DB-Prologの方が従来のPrologに比べて高速にプログラムを実行できることがわかった．さらに，評価方式を用いて結合されている複数の関係データベースに閉じた世界の仮定を行うことで，知識の多世界化などが可能となっている．

1. まえがき

Prologを用いた知識情報処理では，プログラムの大規模化に伴い実行効率が低下すること，高速集合演算機能が欠如していることなど，実用的なデータベース操作機能の不備が問題となっている．たとえば，自然言語処理や知識工学の研究などでは辞書や知識ベースが大規模になり，これらを効率よく取り扱うことが困難な状況が生じている．また，従来のProlog処理系には関係データベースに蓄えられたデータに直接アクセスする機能がないために，既存のデータベースがある場合でもProlog上にデータベースを再構築しなければならないという問題がある．以上の問題点を解決するために，著者らは関係データベースとPrologを結合したシステムDB-Prologを作成したので報告する．両者の結合には，評価方式，非評価方式[5]と呼ばれる二方式を用いており，以下に示す機能が実現されている．

1) 大量のファクトを含むPrologプログラムを効率よく実行する機能．

2) 関係データベース中の数値や文字列などのデータにアクセスする機能．

DB-Prologでは，ファクト集合は非評価方式を用いて結合されたデータベースに，数値・文字列データは評価方式を用いて結合されたデータベースに格納される．ファクト集合の検索は，関係データベースの検索機能を用いて実現したユニフィケーションを通じて行われる．数値・文字列データの検索は，DB-Prologが関係演算子に対応させてあらかじめ用意しているデータベース操作用の組み込み述語を用いて行われる．これらの組み込み述語には，データベース中のデータに集合演算を施す述語や統計的処理を行う述語などがある．また，既存のデータベースにアクセスするための述語なども含まれている．

以下，2章でPrologと関係データベースとの結合方式について議論する．3章ではDB-Prologの概要を示し，4章，5章で外部データベース，ユーザデータベースについて詳説する．さらに，6章でDB-Prologの評価，および検討を行う．

2. Prologと関係データベースとの結合方式

Prologと関係データベースの結合には，評価方式と非評価方式の2通りの方式がある．Prologを関係データベースの操作言語とみなし，Prologプログラムを関係データベースへの問い合わせに変換し，データベースの操作を行う方式を評価方式と呼ぶ[2]．評価方式では，関係データベースへの検索要求がPrologプログラムの実行要求から区別されており，関係演算を用いてデータベースを操作することができるために，高速集合演算が可能である[15]．

一方，関係データベースをProlog処理系の一部分とみなし，関係データベースへの検索要求とPrologプログラムの実行要求を区別しない方式を非評価方式

† DB-Prolog: An Extended PROLOG Capable of Accessing Relational Databases by TAKESHI IMANAKA (Faculty of Engineering Science, Osaka University), KUNIAKI UEHARA and JYUN'ICHI TOYODA (Institute of Scientific and Industrial Research, Osaka University).

†† 大阪大学基礎工学部情報工学科
††† 大阪大学産業科学研究所

と呼ぶ．非評価方式では，Prolog プログラムの実行要求，関係データベースへの検索要求が入り混じっているために，実行効率が悪くなるという問題点が指摘されている[7]．しかしながら，関係データベース中のデータと Prolog プログラムを明確に区別しないために，①Prolog で扱われるような構造を持つデータを関係データベースに格納し，利用できること，②Prolog の持つ強力な推論機能をそのまま関係データベース中のデータに適用できることなどの利点がある．また，非評価方式では，インデックシングによる関係データベースの高速検索や関係演算 selection などを用いて実現したユニフィケーションなどによって，従来の Prolog 処理系よりも効率よく大規模な Plolog プログラムを実行する機能が実現できると考えられる．

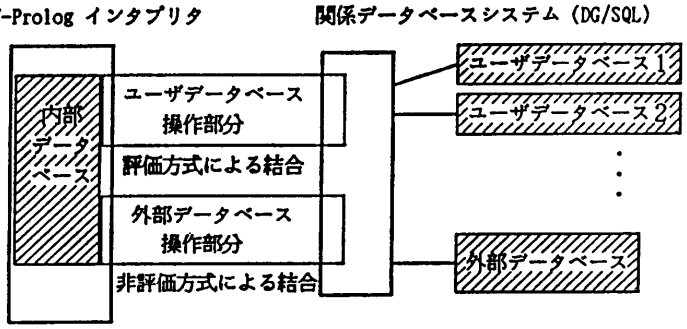


図 1 DB-Prolog システム構成図
Fig. 1 System configuration of DB-Prolog.

以上のことから，既存の関係データベースのように構造を持たないデータにアクセスする時には集合演算が可能な評価方式を用い，Prolog プログラムのように構造を持つデータを関係データベース中に格納し，利用する時には非評価方式を用いるというように，両方式を使い分ければ1章で述べたいくつかの問題点を解決することができる．

3. DB-Prolog の概要

3.1 システム構成

DB-Prolog は，日本データゼネラル社のスーパーミニコン MV-8000 II 上で，Prolog インタプリタ C-Prolog と関係データベースシステム DG/SQL を結合して実現している[3),9)]☆．**図1**に DB-Prolog のシステム構成を示す．非評価方式を用いて結合された関係データベースを外部データベースと呼ぶ．外部データベースには，変数を含まないファクト集合が格納される．評価方式を用いて結合されている関係データベースをユーザデータベースと呼ぶ．ユーザデータベースには，数値や文字列データが格納される．また，内部形式に変換されたプログラムが格納されている Prolog 処理系のデータベースを内部データベースと呼ぶ．

3.2 システムの基本動作

DB-Prolog は，ゴールを実行すると以下のように動作する．まず，ゴールがユーザデータベース操作用の述語かどうかを判断する．ユーザデータベース操作用の述語である場合は，ユーザデータベースへの検索要求とみなし，ゴールから DML（データベース操作言語）あるいは DDL（データベース定義言語）によるユーザデータベースを操作するためのプログラムを合成し，直ちにそのプログラムを実行する．ユーザデータベース操作用の述語でない場合は，ゴールをプログラムの実行要求とみなし，内部データベースから実行可能な Prolog プログラムを検索し，そのプログラムを起動する．内部データベース中に実行可能なプログラムが見つからなければ，ゴールを外部データベースへの検索要求とみなし，外部データベースを検索する．外部データベースを検索した結果，ゴールとユニフィケーション可能なファクトがなければ，ゴールの実行は失敗し，バックトラッキングが生じる．

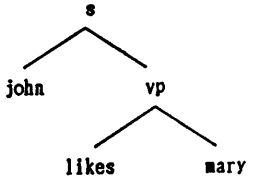
4. 外部データベースへのアクセス

4.1 外部データベースにおけるファクトと関係の対応

従来より指摘されているように，引数が構造を持たないファクトは，引数を属性値に割り当て，関係データベースとうまく対応づけることができる[6),10),16)]．しかしながら，ファクトの引数が構造を持つ場合は，関係データベースと直接対応づけることが困難である．これは，関係が正規化されているために，属性値として構造を持つことができないことによる[1)]．このような問題を解決するために，ファクトの持つ構造を木構造に展開して関係データベースに対応づける手法を提案する[8)]．

ファクトは**図2**に示すような木構造で表現することができる．この木構造の各ノードに対し，木の深さを X 座標，木の拡がりを Y 座標とする2次元座標を割

☆ エジンバラ大学で開発された C-Prolog を，本研究室で MV-8000 II 上に移植したものであり[4)]，MV-Prolog とも呼ばれている．



s(john, vp(likes, mary))

図 2 ファクトと木構造
Fig. 2 Fact and its tree structure representation.

り当てれば，ファクトを1次元的な配列で表すことができる．これをファクトの線形化（linearization）と呼ぶ*．

ファクトの線形化に用いる2次元座標系には，以下の2通りが考えられる．一つは各ノードに割り当てられる X，Y 座標を親ノードの座標から直接計算することができる座標系である．この座標系は，各ノードの座標が他のノードに関係なく親ノードの座標のみによって決定できることから，絶対座標系と呼ぶ．たとえば，図3に示した座標系は絶対座標系である．この図で m は木構造の枝の数を表しており，木構造は m 進木となっている．各ノードの座標は図中の計算式によって与えられる．絶対座標系では，深さが増加するにつれて Y 座標の値が指数関数的に増加するという問題がある．このために，図3の座標系ではファクトの木構造が深さ10以上の左方枝分かれ構造になると，Y 座標が処理系の扱いうる整数の最大値を越えてしまい，外部データベースに登録できないという事態が生ずる．

もう一つは，木構造中の深さを X 座標に，それぞれの深さごとに右端（または左端）から数えた位置を

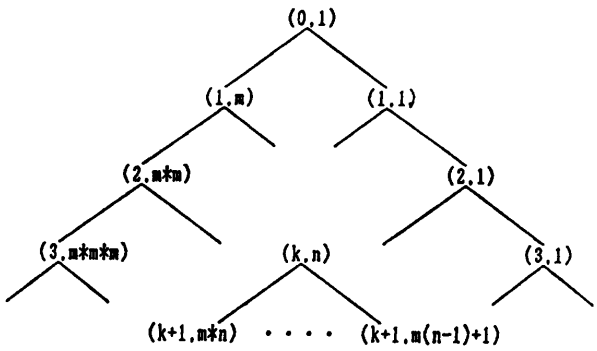


図 3 2次元座標
Fig. 3 Two-dimensional coordinates.

* システム組み込み述語 assert, retract を用いて内部データベースにファクトの登録や削除をするのと同様に，外部データベースに対しては，新たにシステム組み込み述語 assertout, retractout を用意しており，線形化されたファクトの登録や削除が行えるようになっている．

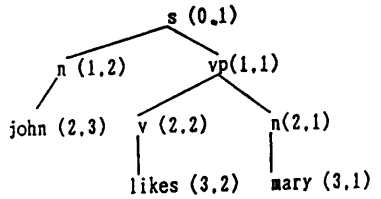


図 4 相対座標系
Fig. 4 Relative coordinates.

Y 座標に割り当てる座標系である．この座標系は，各ノードの座標が同一の深さにある他のノードによって影響を受けることから，相対座標系と呼ぶ．例として，ファクト s(n(john), vp(v(likes), n(mary)) に相対座標を割り当てたものを図4に示す．相対座標系は Y 座標のとりうる値の範囲が深さごとのノード数と等しいために，Y 座標が深さに対して指数関数的に増加するといった問題は生じない．しかしながら，変数を含んだファクトに相対座標を割り当てた場合，変数に値（特に木構造の場合）が代入されるたびに，既に割り当てられた他のノードの座標を再計算し，新たな座標を与えなければならないという問題が生じる．

このように，各座標系にはそれぞれ欠点がある．しかしながら，絶対座標系では，①ゴールとファクトのユニフィケーション時に1回だけの座標計算ですむこと，② Prolog でよく用いられるリスト構造は右方枝分かれ構造であり，Y 座標の指数的増加という事態は実用レベルではほとんど起き得ないことなどから，ユニフィケーションの実行効率を考慮して，DB-Prolog では図3の絶対座標系を用いている．また，Prolog のような論理型言語で記述されたプログラムでは，9引数を越える述語は非常にまれにしか現れないために[20]，m の値を9としている．

絶対座標系を用いて線形化されたファクトは，さらに関係に展開される．関係の各属性は，X 座標，Y 座標，木のノードが持つ値 node，およびノードから下に伸びる枝の数 arg からなる．属性 node の値はアトム，あるいは関数名（ファンクタ）である．また，ノードから下に伸びる枝がない場合，すなわちノードがアトムの場合は arg の値を0としている．以降では，この四つの値からなる組を4項組と呼ぶ．例として，図2のファクトを木構造に展開し，絶対座標を割り当て関係データベースに対応づけたものを図5に示す．

実際の外部データベースは，ファクトの述語名と引数の数でインデックシングされており，図5の関係のほかに，検索を高速化するためのいくつかの属性や識

x	y	node	arg
0	1	s	2
1	2	john	0
1	1	vp	2
2	2	likes	0
2	1	mary	0

図 5 ファクト s(john, vp(likes, mary)) と関係の対応
Fig. 5 4-Argument relation representing the fact "s(john, vp(likes, mary))".

別子が加えられている．識別子は4項組に線形化されたファクトを区別するために用いられ，1個のファクトから線形化された4項組集合には，すべて同一の識別子が与えられている．以降では，簡単のために4項組と識別子 id を合わせてタップルと呼ぶ．

4.2 検索機能を用いたユニフィケーション

ゴールが外部データベースへの検索要求とみなされると，DB-Prolog はゴールを木構造に展開し，座標を割り当て，4項組集合に線形化する．例えば，ゴール |?-s(john, Vp) は**図6**に示すような4項組集合に線形化される*．次に，線形化された4項組集合のうち属性 node の値が変数でない任意の4項組を選択し，4項組の各属性値と等しい値を持つタップルの属性 id を外部データベースから検索する．求められた属性 id の値を持つファクト集合のみがゴールとのユニフィケーションを実行する対象となる．このように，4項組を用いて外部データベース中のファクト集合全体からユニフィケーションを実行する対象となるファクト集合を求めることをファクト集合の絞り込みと呼ぶ．たとえば，**図7**に示す三つのファクトが外部データベースに登録されているときに，図6の4項組集合から4項組 (0, 1, s, 2) を選択し，各属性値と等しい値を持つタップルを検索するとファクト集合は id=1, 2 に絞り込まれる．

さらに，ファクトのタップル集合とゴールの4項組集合を比較し，X 座標，Y 座標の値が互いに等しいタップルと4項組の対を順に取り出し，属性 node と arg の値を比較する．4項組の属性 node の値が変数

(x,	y,	node,	arg)
(0,	1,	s,	2)
(1,	2,	john,	0)
(1,	1,	Vp,	)

図 6 4 項 組
Fig. 6 4-Argument relation.

* 外部データベース中のファクトはタップル集合で表されているが，ゴールは実行時に展開されるために，固有の識別子が不要であり，4項組集合として表現される．

id	x	y	node	arg
1	0	1	s	2
1	1	1	runs	0
1	1	2	tom	0
2	0	1	s	2
2	1	1	vp	2
2	1	2	john	0
2	2	2	likes	0
2	2	1	mary	0
3	0	1	s	1
3	1	1	yes	0

s(john, vp(likes, mary)).
s(tom, runs).
s(yes).

図 7 外部データベース
Fig. 7 External database.

の場合は，対応するタップルの値を Prolog の内部表現に変換し，変数に代入する．対応するタップルの属性 node の値がアトムの場合は直接変数に代入される．また，ファンクタの場合（属性 arg が1以上のとき）は，そのタップルをルートノードとする部分木構造が複合項に変換され，変数に代入される．

先の例では，id=1 で識別されるタップル集合と4項組集合を比較すると，4項組 (1, 2, john, 0) とタップル (1, 1, 2, tom, 0) が一致しない．したがって，id=1 で識別されるファクトとゴールはユニフィケーションできない．id=2 で識別されるタップル集合に対して同様の操作を実行すると，すべての4項組は対応するタップルの値とそれぞれ一致し，ユニフィケーションできる．最終的に，タップル集合 {(2, 1, 1, vp, 2), (2, 2, 2, likes, 0), (2, 2, 1, mary, 0)} から複合項 vp(likes, mary) が合成され，Prolog の内部形式に変換されて変数 Vp に代入される．

5. ユーザデータベースへのアクセス

5.1 システム組み込み述語の実現手法

ユーザデータベースの作成，および検索を行うための組み込み述語は，HLI (Host Language Interface) を用いて実現している．HLI は，DML とC言語のインタフェースであり，C言語で記述されたいくつかのライブラリ関数を持っている[14]．これらのライブラリ関数は組み込み述語の実行過程で呼び出され，DML あるいは DDL を用いて記述されたユーザデータベース操作用プログラムを受け取り，インタプリタ形式で実行するものである．たとえば，ユーザデータベースを検索する組み込み述語 select_db を実行する

と，select_db はあらかじめ用意している検索プログラムに，テーブル名，検索キーなどの情報を埋め込んで実行可能な DML のプログラムを合成し，ライブラリ関数に受け渡すようになっている．

5.2 ユーザデータベース操作述語

DB-Prolog の組み込み述語には，ユーザデータベースを作成・削除する述語，データベース中の関係を定義・削除する述語，データベースのキー，インデックスを操作する述語，データベースを検索する述語，データベースに対するデータの登録，削除を行う述語などを用意している．さらに，C, Fortran などの手続き型言語によって記述したユーザデータベース操作プログラムを呼び出す述語も用意している．本節では，ユーザデータベース操作述語のうち，データベースの定義とデータベースの検索を行う述語の実行過程について説明する．

5.2.1 データベースの定義用述語

ユーザデータベースの作成と関係の定義を行う Prolog プログラムを図 8(a) に示す．このプログラムは，組み込み述語 create_database を用いてユーザデータベース db を作成するものである．また，組み込み述語 make_table を用いて，関係名が supplier，属性名が sname, status, city であるような関係を定義している．述語 make_table の第1引数中にある status (int) は属性 status が整数型であることを示している．属性 sname, city のように属性値の型が指定されていない場合は，デフォルト値として 40 字の文字列型が自動的に割り当てられる．

```
#- create_database (db).
   make_table (supplier (sname, status (int), city), key
   sname, status)).
```

(a) プログラム

```
CREATE DATABASE "db" DEFINED BY
  "DATABASE DEFINITION; TABLE attribute;
  table_name: CHAR (40); number: INTEGER;
  attribute_name: CHAR (40);
  KEY (table_name, number);
END DATABASE DEFINITION;"
ALTER DATABASE db APPLYING
  "ALTER DATABASE; ADD TABLE supplier;
  sname: CHAR (40); status: INTEGER;
  city: CHAR (40); END TABLE supplier;
END ALTER DATABASE;"
```

(b) プログラムより合成される DDL

図 8 システム組み込み述語から合成される DDL プログラム

Fig. 8 DDL program synthesized from built-in predicates "create_database" and "make_table".

このPrologプログラムを実行すると，あらかじめ用意しているデータベース定義プログラムに各組み込み述語の引数に与えられた情報が埋め込まれて，図8(b) の DDL プログラムが合成される．合成された DDL プログラムは，HLI のライブラリ関数に渡された後に，インタプリタ形式で実行される．

5.2.2 データベースの検索用述語

図9に示す関係 supplier から，タップル検索する Prolog プログラムを図 10(a)に示す．このプログラムは，①属性 status の値が20であるタップルを検索し，②検索されたタップルの属性 sname, city の値をカーソル領域変数 Cursorl に蓄え*，③カーソル領域変数から検索結果を取り出すものである．①，②の操作は組み込み述語 select_db を用いて，③の操作は組み込み述語 fetch_db を用いて行われる．述語 select_db の第1引数は，ファンクタが関係名に，各引数が属性に対応する複合項で，アトムは検索キーを，変数は検索したい属性を表しており，第2引数は

sname	status	city
smith	20	london
jones	10	paris
blake	30	paris
clark	20	london
adams	30	athens

図 9 関係 supplier

Fig. 9 Relation "supplier".

```
#- select_db (supplier (Sname, 20, City), Cursorl),
   fetch_db (Cursorl, [Sname, City]).
```

(a) プログラム

```
SELECT attribute_name INTO: att_name 1
  FROM attribute IN db
  WHERE table_name= "supplier" AND number=1
SELECT attribute_name INTO: att name 2
  FROM attribute IN db
  WHERE table_name= "supplier" AND number=2
SELECT attribute_name INTO: att name 3
  FROM attribute IN db
  WHERE table_name= "supplier" AND number=3
SELECT sname, city INTO cursorl
  FROM supplier IN db WHERE status=20
FETCH FROM cursorl IN db SET: arg 1,: arg 2
```

(b) プログラムより合成される DML

図 10 システム組み込み述語から合成される DML プログラム

Fig. 10 DML program synthesized from built-in predicates "select_db" and "fetch_db".

* カーソル領域変数 Cursorl には，DG/SQL で用意されているカーソル領域が割り当てられる．カーソル領域は検索結果を蓄えておくために用いられる．

検索結果を蓄えるためのカーソル領域変数を表している．また，述語 fetch_db の第1引数には，カーソル領域変数，第2引数には検索結果を代入する変数がリスト形式で与えられる．

この Prolog プログラムを実行すると，述語 select_db, fetch_db の引数から図 10(b)の DML プログラムが合成される．DML プログラムは，まず関係 attribute から関係 supplier の属性名を検索し，求められた属性名を変数 att_name1, att_name2, att_name3 に代入する．関係 attribute は DB-Prolog が自動的に管理している関係であり，データベース中に定義されているすべての関係の関係名と属性名が登録されている．この例では，ユーザデータベース db 中に関係 supplier が存在していること，関係 supplier の属性名はそれぞれ sname, status, city であることが登録されている．この検索によって，変数 att_name1, att_name2, att_name3 には，それぞれ属性名 sname, status, city が代入される．これらの属性名を用いて，関係 supplier から属性 status の値が 20 であるタップルの属性 sname と city が検索され，Cursor1 に結果が蓄えられる．

次に，DML プログラム中の変数 arg1, arg2 に検索結果の一つが代入される．さらに，これらの値は述語 fetch_db の変数 Sname, City に渡される．また，カーソル領域に蓄えられた検索結果が複数個ある場合は，fetch_db を繰り返し呼び出すことで，検索されたすべての値を取り出すことができる．

6. 考　　察

本章では，外部データベース中に蓄えられた大量のファクトの検索効率について，いくつかの評価および検討を行う．さらに，ユーザデータベースへのアクセスによって可能になる処理とデータベースの共有化について考察する．

6.1 外部データベースの定量的評価

ファクトを外部データベースに登録した場合と内部データベースに登録した場合について，ユニフィケーションに必要な時間を測定した．この測定によって，データベースの検索機能を用いたユニフィケーションと従来の Prolog インタプリタのユニフィケーションを実行速度の面から比較する．外部データベースの検索は，Prolog インタプリタのユニフィケーションルーチンから，DG/SQL の SQL 文を埋め込んだ C 言語のサブルーチンを呼びだす方法で実現している．したがって，関係代数の実装方式は日本データゼネラル社の関係データベースシステム DG/SQL に従ったものである．一方，内部データベースの検索は，Prolog インタプリタをそのまま利用しているために，蓄えられているファクトはインデックシングされていない．また，外部データベース検索用のサブルーチンの埋め込みによる Prolog インタプリタの実行速度の低下はほとんどなく，約 1.8 kLips で稼動している．

6.1.1 評価実験の内容

内部データベースのアクセスが，ディスクアクセス時間の影響を受けないのに対して，外部データベースへのアクセスは，ディスクアクセス時間が支配的である．さらに，MV-8000II 上では，ディスクアクセス時間が他のプロセスによって大幅に影響を受けるために，評価実験では特定のプロセスのみが動いている一定の環境をつくり，その環境下で1回のユニフィケーションに要する応答時間を測定した．データベースに登録するファクトは，自然言語処理で用いられる辞書を想定したものである．なお評価を簡単にするために，本実験で想定した辞書は図 11 に示すような動詞のみからなるものとする．ファクトの第1引数には動詞の原形（見出し語），第2引数には動詞の活用形，第3引数には現在分詞，第4引数には意味が登録されているものとする．辞書の検索は以下の三つの場合を考えた．

1) 第1引数を検索キーとする場合

これは，動詞の原形から意味を引く場合に相当する．動詞の意味を表す第4引数のリストが検索される．

2) 第2引数のリストの第2要素を検索キーとする場合

これは，動詞の過去形から動詞の原形と意味を引く場合に相当する．動詞の原形を表す第1引数と動詞の意味を表す第4引数が検索される．

3) バックトラッキングをする場合

これは，同じ動詞の別の用法を引く場合に相当する．たとえば，自動詞と他動詞の原形が等しく意味が異なる動詞を検索する場合に相当する．実験ではそれぞれの動詞をファクト集合の先頭と末尾に登録し，先頭のファクトとユニフィケーションした時点から末尾のファクトとユニフィケーションするまでの時間を測

動詞（原形，[現在形，過去形，過去分詞]，現在分詞，[意味_1，意味_2，..]).

図 11 辞書データ
Fig. 11 Sample data.

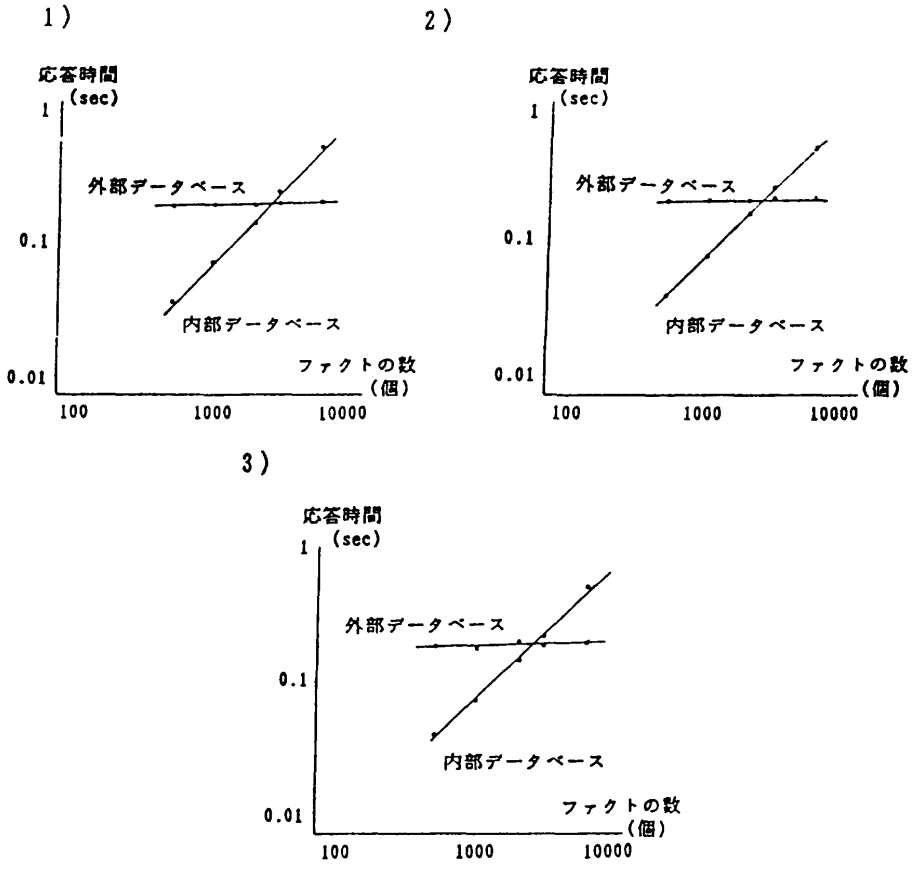


図 12 実験結果
Fig. 12 Comparison of execution time while retrieving facts in internal database and external database.

定した.

6.1.2 実験結果と検討

実験結果を図 12 に示す. このグラフからわかるように, ファクト数が 3,000 個を越えると, 内部データベースに蓄えるよりも外部データベースに蓄えた方が高速に検索することができる. また, バックトラックをおこして次の解を得るまでの実行時間についても, グラフの 3) から同様のことが言える.

内部データベース中のファクトの検索時間は, ファクト集合の先頭から逐次的にユニフィケーションを試みているために, ファクトの数に比例して単調的に増加している. これに対し, 外部データベースは①既存のデータベースシステムが管理していること, ②実験で用いた辞書は, 同一の見出し語を持つファクト数がほとんどない (実験ではたかだか 3 個しかないようにしている) ために少数のファクトに絞り込めることからユニフィケーションに要する時間はほぼ一定となっている.

ファクト集合を高速に検索する方法として Prolog コンパイラを用いる方法があるが, 辞書などの知識は利用するたびに追加, 削除, 変更などの更新処理が行われる可能性があり, 毎回コンパイルし直すのは現実的でない. したがって, 更新する可能性のある大規模なファクト集合は外部データベースに格納することで, 効率のよいユニフィケーションが実現できることが判明した.

6.1.3 検索方式に対する検討

4.2 節で述べたように, DB-Prolog ではゴールの実行時に選択される 4 項組を検索キーとして, ユニフィケーション可能なファクトの候補を絞り込んでいる. したがって, 選択される 4 項組によってユニフィケーションの効率が異なっている. これは選択された 4 項組に対応する引数でファクト集合をインデックシングした場合と同様の効果を持っている. たとえば, 第 2 引数に対応する 4 項組を選択すると, ファクト集合を第 2 引数でインデックシングするのと同様の効果

がある．一方，Quintus-Prolog や DEC-10 Prolog などでは，プログラムのコンパイル時に，常に第1引数でインデックシングしており[17),18)]，ゴールの第1引数が変数になると，検索キーを失い，逐次的にファクトを検索しなければならないという問題点がある．本インデックシング方式は，従来の固定的なインデックシングに比べて，実行時に動的に検索キーを選択していることから，動的インデックシングと呼ぶ．また，組み込み述語 define_key を用いて任意の引数でのインデックシングを指定することも可能になっている．引数が指定されていないときには，ゴールの木構造の左方縦型探索で最初に見つかった，アトムまたはファンクタに対応する4項組を検索キーとして用いるようにしている．たとえば，実験2)では，ゴールの第2引数の第2要素が動的インデックシングの検索キーとして用いられている．

6.2 ユーザデータベースの定性的評価

外部データベースだけでなく，ユーザデータベースをアクセスすることによって得られる利点として，1) データベースの統計的処理，2) 一時的な変更を含む処理，3) 複数の利用者によるデータベースの共有化，4) 世界，役割による知識の分類などがあげられる．以下ではこれらの4点について考察する．

1) データベースの統計的処理

ユーザデータベースに対する集合演算機能を用いた処理として，データベースの統計的処理がある．たとえば，会社の給料を管理するデータベースに対して，「A社の社員から高額所得者を選べ」という問い合わせを行う場合について考える．この問い合わせに対して，DB-Prolog では「平均の給料の2倍以上なら，高額所得者であろう」といったような高額所得者に関する知識と，ユーザデータベース中にあるデータの平均値を求める組み込み関数 (aggregate function) を用いて，データベース検索を行うことができる．これと同様のデータベース検索を手続き型言語のプログラムで行うと，高額所得者に関する知識が明示的に表現できず，またプログラム自身も複雑なものになる．さらに，従来の Prolog では既存のデータベースにアクセスすることができないために，このような検索自体が不可能である．以上のように，DB-Prolog を用いれば，既存のデータベースをユーザデータベースとみなし，知識工学的手法を用いたデータベース検索を行うことができる．

2) 一時的な変更を含む処理

DB-Prolog の外部データベースやユーザデータベースは，ともに既存の関係データベースシステムによって管理されている．したがって，外部データベース，あるいはユーザデータベースのいずれにおいてもトランザクションの管理が行われている．このトランザクション機能を用いて，利用者の一時的な変更をあたかも恒久的な変更のように扱うことができる．たとえばトランザクションの機能を用いて一時的な知識の変更をデータベースに施すことができ，特定の知識を省いた推論，あるいは一時的に適用可能な知識の表現が可能となっている．

3) 複数の利用者によるデータベースの共有化

DB-Prolog は，既存のデータベースシステムを用いて構築しているために，データベースの共有化が可能である．また，データベースの共有化はデータベースシステムの管理機能を用いて実現されているために，共有化に伴って生じる問題，たとえばアクセス権の管理，データ破壊に対する健全なデータベースの復帰などは，すべてデータベースシステムが対処している範囲内で解決できる．従来の知識ベースシステムでは，このような厳密な知識ベースの管理はなされていない．また，データベースの管理者の概念を用いて，知識ベースの管理者を設定し，処理系と独立に知識ベースを管理できるという利点がある．

4) 世界，役割による知識の分類

知識情報処理の分野では，知識を世界ごとに管理することが重要であるということが従来より指摘されている．Prolog を用いて知識を世界ごとに分類，管理するには，引数に世界を割り当てる手法[11)]，あるいは述語名に世界を割り当てる手法[22)]などが提案されている．これに対し，各ユーザデータベースごとに閉じた世界の仮定 (Closed World Assumption) を行い[12)]，各世界の知識を別個のユーザデータベースに割り当てれば，引数や述語名に世界を指定する情報を与える必要はなくなる．ユーザデータベースに割り当てられた各世界の知識は，必要に応じてユーザデータベースをオープン・クローズしながら使い分けることができる．

また，知識は世界ごとだけでなく，役割ごとにも分類して用いることが重要である．たとえば，談話理解システムでは常識的な知識は常に利用される可能性があり，知識の適用は即時に行わなければならない．しかしながら，専門的な知識は各問題分野固有の問題解決のときに初めて必要となる知識である．このような

分類は，外部データベースに常時利用するような知識を，ユーザデータベースに適宜利用するような知識を割り当てることで可能になっている．

6.3 他の結合システムとの比較

従来より，Prolog と関係データベースの結合システムは数多く提案されている．これらのシステムは，結合に用いている方式によって二つに分けることができる．一つは評価方式のみを用いたシステムで，PSI と Delta をネットワークを介して結合した ICOT のシステムなどがある[21),22)]．もう一つは非評価方式のみを用いたシステムで，C-Prolog と Ingres を結合した東京大学のシステム[23)]，日立の LONRI[19)] などがある．これらのシステムに対し，DB-Prolog は評価方式，非評価方式の両方式を同一システム内で実現しているという点が特長としてあげられる．

次に，DB-Prolog で用いた結合手法を他のシステムと比較する．DB-Prolog で用いている評価方式は，データベース操作述語を評価後，即時実行するものである．ICOT のシステムはデータベースとの通信回数を減らすために[21)]，データベース操作部分の実行を遅らせ，最適化などの処理を施してから実行しているのに対し，本方式はデータベース操作用述語が呼び出されると同時に実行するという点でインタプリタの自然な拡張となっている．

DB-Prolog で用いた非評価方式は，関係演算 selection と動的インデックシングの二つを用いて実現したユニフィケーションを通じて，関係データベース中のファクトを検索することによって実現されている．他にも関係演算を用いてユニフィケーションを実現しているシステムがあるが，join を用いている点，および関係演算自体の再設計を行っている点が DB-Prolog と異なっている．join を用いた場合には，ユニフィケーション可能なファクトすべてを一度に検索できるという利点があるが，どのようなファクト集合の検索でも常に蓄積されたファクトの量によって影響を受けるという欠点がある[23),24)]．これに対し，DB-Prolog の手法では，動的インデックシングによってファクト集合を少数のファクトに絞り込むことができれば，ファクト数は検索の実行時間にあまり影響を及ぼさないために，従来のインタプリタよりも高速にユニフィケーションを実行することが可能になる．しかしながら，いずれの引数によっても少数のファクトに絞り込みができないファクト集合は，内部データベースに格納した方が効率的である．これは，外部データベース中のファクトとゴールのユニフィケーションに必要な selection の回数が，動的インデックシングによって絞り込まれたファクトの候補数に比例して多くなり，外部データベースの検索効率が悪くなるためである．

6.4 システム上の制約

・カーソル領域の数

現在のシステムでは，5個しか用意していないが，既に使用されたカーソル領域を再利用することも可能なために，実用上は問題ない．しかしながら，最大限 50 個まで拡張可能である．

・外部データベース中のファクト

現在のシステムでは，引数を9個以上持たない変数なしファクトだけを扱っている．変数を含むファクトをデータベース中に格納する場合には，新たに変数領域などを Prolog インタプリタのスタック上に確保しなければならないために，ファクトを一度内部データベースに転送してからユニフィケーションを実行する必要があり，オーバヘッドが大きくなる．したがって，現在のシステムでは，変数を含むファクトは内部データベースに格納する方が効率的である．引数の個数に対する制限は，座標を割り当てるシステム内の変数のビット数を現在の 32 bit よりも大きくし，図3の m を9よりもさらに大きくすることで解決することができる．

・ゴール呼出し

goal (X, X) のように変数の束縛を含んだ呼出しについては，外部データベースを検索する時に，goal (X, Y), $X==Y$ と展開してから実行している．

7. むすび

本論文では，Prolog と関係データベースを評価，非評価の両方式で結合する DB-Prolog を示した．今後は，ファクト集合だけでなくルール集合が大規模化した場合にプログラムを効率よく実行するための機能[13)]について検討する予定である．

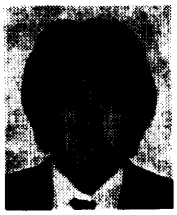
謝辞 本研究の初期段階において御協力いただいた本学大学院生の林 浩一氏（現在，富士ゼロックス），および河合和久氏（現在，豊橋技術科学大学）に感謝いたします．

参考文献

1) Date, C. J.: *An Introduction to Database Systems*, 3rd edition, Addison-Wesley, Reading, Mass. (1981).
2) Deyi, L.: *A PROLOG Database System*,

John Wiley & Sons Inc., New York (1984).
3) DG/SQL User's Manual, Data General Co., Westboro, Mass. (1984).
4) 藤井ほか：C-Prolog のスーパーミニコン MV-8000Ⅱ への移植，第 28 回情報処理学会全国大会講演論文集，2G-7 (1984).
5) Gallaire, H. and Minker, J.: *Logic and Data Base*, Plenum Press, New York (1978).
6) 林ほか：大規模知識ベースシステム実現のための Prolog の拡張，第 30 回情報処理学会全国大会講演論文集，IL-4 (1985).
7) 伊草ほか：データベースと Prolog，情報処理，Vol. 25, No. 12, pp. 1380-1385 (1984).
8) 今中ほか：大規模知識ベースシステム実現のためのProlog の拡張(2)，第 31 回情報処理学会全国大会講演論文集，6M-1 (1985).
9) 今中ほか：Prolog と関係データベースとの結合システム DB-Prolog，情報処理学会，知識工学と人工知能研究会資料，45-8 (1986).
10) Kowalski, R.: *Logic for Problem Solving*, North-Holland, Amsterdam (1979).
11) Nakashima, H.: Knowledge Representation in Prolog/KR, *International Symposium on Logic Programming*, pp. 126-130 (1984).
12) Reiter, R.: On Closed World Data Bases, in Gallaire, H. and Minker, J. (eds.), *Logic and Data Bases*, Plenum Press, New York (1978).
13) 田中英彦：ＡＩ用コンピュータ・アーキテクチャ，人工知能学会誌，Vol. 1, No. 2, pp. 178-183 (1986).
14) The Host Language Interface User's Manual, Data General Co., Westboro, Mass. (1985).
15) 角田ほか：関係代数演算専用エンジンを備えた関係データベースマシン Delta，日経エレクトロニクス，1985 年 9 月 23 日号, pp. 235-280 (1985).
16) 上野晴樹：知識工学入門，オーム社，東京，(1985).
17) Warren, D. H. D.: An Abstract Prolog Instruction Set, SRI International, Technical Note 309 (1983).
18) Warren, D. H. D.: Implementing Prolog—Compiling Predicate Logic Programs—, Dept. of Artificial Intelligence, University of Edinburgh, Research Reports 39 & 40 (1977).
19) 山口ほか：論理型言語処理系 LONRI の開発—RDB インタフェース—，第 32 回情報処理学会全国大会講演論文集，1M-5 (1986).
20) 山本ほか：論理型言語 ESP のプログラム特性評価，*Proc. of The Logic Programming Conference*, pp. 259-270 (1985).
21) 横田ほか：Prolog による推論機構と関係データベースの結合，ICOT Technical Report, TR-031 (1983).
22) Yokota, H. et al.: An Enhanced Inference Mechanism for Generating Relational Algebra Queries, ICOT Technical Report, TR-026 (1983).
23) 吉田ほか：推論機能と関係データベースの融合—関係データベースを用いた Prolog インタプリタの試作と評価—，第 32 回情報処理学会全国大会講演論文集，1M-5 (1986).
24) 吉田ほか：関係データベースを用いた論理型言語実行系の試作と検討，電子通信学会技術研究所報告，AI 86-13 (1986).

(昭和 61 年 8 月 14 日受付)
(昭和 62 年 2 月 12 日採録)



今中　武（正会員）

昭和 37 年生．昭和 60 年大阪大学基礎工学部情報工業科卒業．現在，同大学院基礎工学研究科在学中．自動プログラム合成に興味を持つ．

上原　邦昭（正会員）

昭和 29 年生．昭和 53 年大阪大学基礎工学部情報工学科卒業．昭和 58 年同大学院基礎工学研究科博士後期課程退学．同年，大阪大学産業科学研究所勤務．現在，同研究所助手．工学博士．現在，人工知能，特に自然言語理解，および自動プログラム合成の研究に従事している．ACM，電子情報通信学会，計量国語学会，人工知能学会，日本ソフトウェア科学会各会員．

豊田　順一（正会員）

昭和 13 年生．昭和 36 年大阪大学工学部通信工学科卒業．昭和 41 年同大学院博士後期課程単位取得退学．同年大阪大学基礎工学部助手．昭和 44 年助教授．昭和 57 年大阪大学産業科学研究所教授．工学博士．現在，主として，自然言語理解，画像理解，文書画像処理，および ICAI システム等の研究に従事している．電子情報通信学会，日本認知科学会，人工知能学会各会員．