# プロセス指向データベースの実行管理
# Control Techniques for the Process Database

庄司　功　　伊藤　英則
Isao Shouji and Hidenori Itoh
（財）新世代コンピュータ技術開発機構
Institute for New Generation Computer Technology

This paper describes a model of a database, called the process database, that is based on the parallel logic programming language GHC. Generally in the field of knowledge base systems which deal with a great quantity of knowledge, it is essential to introduce the techniques for parallel processings. There are many approaches to construct such techniques. Some of them use several processors to manipulate data in parallel, another use parallel programming languages. This paper focuses on the latter case. The use of GHC enables us not only to manipulate data in parallel, but to ignore the complex structures which are caused by conjugating several processors.

## 1．はじめに

一般に，大量な知識を取り扱う知識ベースにおいては，データ検索処理等を高速に実行するため，その並列処理技術が不可欠である。それには，例えば，複数の検索用プロセッサを用意し，これらを並列に実行することによって実現を図るものなど様々なアプローチがある。本稿では，並列論理型言語ＧＨＣを基礎にし，その上で実現されるデータベースを考察する。

ＧＨＣでは，問題解決のためにいくつかのゴールを用意し，それぞれのゴールに対してプロセスを割り当て，これらのプロセスが並列に動作することによって問題解決を図っている。しかし，ＧＨＣにはPrologのassert, retractに相当するような機能を持つ述語がないため，Prologで実現されているようなデータ表現を流用することはできない。そこで，ゴール，言い換えれば，それと対応するプロセスにデータを格納し，ストリーム通信と呼ばれる通信形態を利用してこのデータを操作することを考えれば，ＧＨＣの並列性を活かしたデータベースが実現できる。本稿の目的は，プロセスを基本単位としてＧＨＣの並列性を活かしたデータベース，いわばプロセス指向データベースの一つのモデルを提供することである。

プロセス指向データベース（以下単にＰＤＢと呼ぶ）は，使用者と直接に情報をやりとりするインターフェイス部分とデータが格納されているＰＤＢ本体から構成されている。ＰＤＢ本体は構造化されたプロセス群から組織され，これらのプロセスが実際にデータを保持している。そして，インターフェイス部分はこのようなプロセスのストリームを管理しており，使用者はインターフェイス部分を通じてこのストリームにメッセージを流し，ＰＤＢにアクセスすることができる。本稿では，上記２つの部分のうちＰＤＢ本体について説明しインターフェイス部分に関する説明は省略する。

本稿では第１にＰＤＢの基本構造を述べ，第２にＰＤＢに関する命令体系を説明し，最後にＰＤＢに対する操作例について説明する。なお，本稿では説明の便宜上プロセスをノードと呼ぶことにする。

## 2．構造

ＰＤＢは２つのノード，ｆノードとｄノードから構成される。前者はプリミティブなデータを格納するためのノードであり，後者はこれらのデータの論理的な単位を表現するるためのノードである。

例えば，データＴがプリミティブなデータＸ，Ｙ，Ｚから構成されているとすれば，Ｔに対してはｄノードを対応させ，Ｘ，Ｙ，Ｚにはそれぞれｆノードを対応させる。ここで問題となるのは，ＴとＸ，Ｙ，Ｚの関係をｄノードとｆノードの関係にどのように対応づけるかということである。即ち，ｄノードとｆノードの上記のよう

な関係をＰＤＢの構造としてどのように表現するかということである。

ところで，前節で述べたように，ノードはＧＨＣのプロセスに相当し，データのアクセスはプロセスに対するメッセージの応答を通して行うということであった。前例に基づけば，Ｔに対応するｄノード，Ｘ，Ｙ，Ｚに対応するｆノードには外部からのメッセージを受信するためのストリームを持っている。

そこで，ｄノードがこれらｆノードのストリームを所有する（言い換えれば，ｄノードからｆノードへメッセージを送信できるようにする）ことによって，ＴとＸ，Ｙ，Ｚの関係を表現することが考えられる。つまり，ＰＤＢ上のデータの関係（構造）はノード間のストリームの所有関係で表わそうというのである。

これによれば，ｄノード（現ｄノード）がｆノードあるいはその他のｄノードから構成されるデータの論理的な単位を表わすとは，現ｄノードが，ｆノードあるいはその他のｄノードのストリームを所有すること，と定義される。

このようにしてＰＤＢの構造が決定されるが，ｄノード，ｆノード自体の構造は以下で説明する。

## ２．１．ｆノードの構造

ｆノードはプリミティブなデータを格納するノードである。ｆノードは次の３つの成分から構成される。第１の成分は，ストリームであり，これは外部からのメッセージを受け取る役割をもつ。第２成分は，ノード名であり，これはｆノードを互いに区別するために必要である。第３成分は，格納されるデータである。

## ２．２．ｄノードの構造

ｄノードはｆノードやその他のｄノードのストリームを所有することによってデータの論理的な単位を表現するノードである。

後述するように，ＰＤＢの使用者はｄノード（現ｄノード）のストリームだけにメッセージを送信するという方針をとっているので，現ｄノードの構成要素であるｆノードやｄノードへは現ｄノードを通して送信しなければならない。そのため，現ｄノードストリームには，その構成要素であるｆノードやｄノードへのメッセージが入り交じって流れることになる。このことは，一面において，ｆノードに対するメッセージとｄノードへのそれとを互いに差別化しなければならないという事態を引き起こし，他面において，その差別化されたメッセージを所望のノードへ分配する機能をｄノードが持たなければならないということを意味する。前者に対する対策は後述することとして，後者に対しては，ｄノードにメッセージを分配する機能をもつサブノード（サブプロセス）を付加し，これとｄノードとが一体をなして対処する。

以上により，ｄノードはメッセージ分配機能をもつサブノードと一体となっているが，このサブノードをｆｄ制御，ｆ制御及びｄ制御と定める。

ｆｄ制御，ｆ制御及びｄ制御は以下のような機能をもつ。外部からのメッセージはまずｄノード（現ｄノード）へ送られ，ついでｆｄ制御に渡される。

ｆｄ制御は，そのメッセージがｆ制御へ送信されるものか，ｄ制御へ送信されるものか，あるいは，両制御へ送信されるものかを識別する。そして，第１の場合には，メッセージをｆ制御に送り，第２の場合には，ｄ制御に送り，第３の場合には，両者に送る。

ｆ制御は現ｄノードの構成要素となるｆノードを管理する。ｆ制御は受信したメッセージをその形式に応じてｆノードの全部または一部に送信する。

一方，ｄ制御は現ｄノードの構成要素となるｄノードを管理する。ｄ制御は受信したメッセージをその形式に応じてｄノードの全部または一部に送信する。

次に，これらの構造を説明する。ｄノードは２つの成分からなり，第１成分は，ストリームであり，これは外部からのメッセージを受信し，かつ，ｆｄ制御へメッセージを送信する役割をもつ。第２成分は，ノード名であり，これはｄノードを互いに区別するために必要である。

ｆｄ制御は３つの成分からなり，第１成分は，ｄノードからのメッセージを受信するためのストリームで，これはｄノードの第１成分と同一である。第２成分は，ｆ制御へメッセージを送信するためのストリーム，第３成分は，ｄ制御へメッセージを送信するためのストリームである。

ｆ制御は２成分からなり，第１成分は，ｆｄ制御からのメッセージを受信するためのストリームで，これはｆｄ制御の第２成分と同一である。第２成分は，現ｄノードの構成要素であるｆノードのストリームの集合であり，これはリストで表現されている。

ｄ制御は２成分からなり，第１成分は，ｆｄ制御からのメッセージを受信するためのストリームで，これはｆｄ制御の第３成分と同一である。第２成分は，現ｄノードの構成要素であるｄノードのストリームの集合であり，これはリストで表現されている。

## ３．操作機能

ＰＤＢの使用者はインターフェイス部分において適当なｄノードストリームを選択し，そのストリームにメッセージを送る。ｄノードストリームは複数個選択でき，それらに対して同時にメッセージを送ることができる。ｄノードストリームの選択に際しては，一人が複数のｄノードストリームを選択し，それらの全部または一部を同時に使用する方法と，複数の者がそれぞれｄノードストリームを選択し，各人がそれぞれ独立に使用する方法とが考えられるが，これらいずれの方法も可能である。また，１度選択したｄノードストリームも適当なコマンドを用いることによってその他のｄノードストリームに変更することができる。このように，メッセージ送信対象のｄノードストリームを変更することによって，データ構造に対する自由な取扱いが可能となる。

以上は，インターフェイス部分における機能であるが，その他の機能についてはこれ以上立ち入らず，以下ではＰＤＢ（本体）の機能について説明する。もちろん，インターフェイス部分の機能も以下で説明するＰＤＢの操作命令によって実現されていることは言うまでもない。

本節では，上のように選択したｄノードストリームに対して，どのようなメッセージを送ることができるか，言い換えれば，どのような操作命令があるかを示す。さらに，ＰＤＢに対する操作例について説明する。

### ３．１．操作命令の形式

ＰＤＢに対する操作命令（以下単にコマンドと呼ぶ）は，一般的には次のような形式をとる。コマンド形式は，コマンド名とそれに引き続く引数，リターン変数，コントロール変数から構成され，例えば，コマンド名がｃｏｍなるコマンドは

```
com(Arg, Ret, Ctrl)
Arg  : 引数
Ret  : リターン変数
Ctrl : コントロール変数
```

と表わされる。ここに，リターン変数とは，コマンド実行結果であるリターン値をバインドするための変数であり，コントロール変数とは，コマンド実行中の情報を反映させるための変数である。後者は，例えば，少なくとも一つの解を検索するというコマンドが，複数のｄノードに送られて並列に実行されている場合，そのうちの１つが解を見つけたという情報を他に知らせる役割をもつものである。

ところで，ノードはコマンドを受信しても，必ずしもそれに対して反応するわけではない。つまり，ノードの受信形態には２種類あって，コマンドを受信してもそれに対して何ら反応しない場合と，受信したコマンドに対して何らかの反応を示す場合とがある。さらに，後者の場合には，受信したコマンドをその他のノードに振り分ける場合と，受信したコマンドに応答する場合とがある。前者の場合を，ノードはコマンドを分配するといい，後者の場合を，ノードはコマンドを受理するという。分配に関係するコマンドは，以下に示すように識別子を用いて表現される。以下では，コマンド識別子について説明する。

### ３．２．コマンド識別子

コマンドはその評価方法，ノードへの分配方法により区別される。これらの違いを以下のような識別子を用いて表わす。

#### ３．２．１．評価に基づく識別子

コマンドには，組み込み関数を直接呼出し実行するものと，ｆノードのデータとして格納された組み込み関数を取り出し，それを呼出し実行するものの２種類がある。前者のコマンドに対しては識別子ｓｙｓを設定し，次のように表わす。

```
sys.com(Arg, Ret, Ctrl)
```

一方，後者に対しては識別子ｕｓｒを設定し，次のように表わす。

```
usr.com(Arg, Ret, Ctrl)
```

#### ３．２．２．分配に基づく識別子

上述のように，コマンドはｄノードに送られると，そのサブノードであるｆｄ制御によってｆ制御またはｄ制御に分配される。ｆ制御へ分配されるコマンドに対しては特に識別子を設けないが，ｄ制御へ分配されるコマンドに対しては識別子ｄを設定し，次のように表わす。

```
d.sys.com(Arg, Ret, Ctrl)
d.usr.com(Arg, Ret, Ctrl)
```

そして，一旦分配されると，ｄ制御において，識別子ｄは取り除かれ，現ｄノードの構成要素であるｄノードへ送られる。そこで，識別子ｄをｎ個重ねれば現ｄノードよりｎ階の深さに相当するｄノードへコマンドを分配することができる。

コマンドの分配方法には，上記のように単純にｆ制御またはｄ制御の何れか一方に分配する場合以外に，ｆ制御，ｄ制御の両者に分配する場合もありうる。しかも，この場合には，さらに，ｆ制御に分配したコマンドの実行結果に応じてｄノードへ分配するかしないかを決める場合や，そのような実行結果によらずｄノードへ分配す

る場合が考えられる。このようなコマンドの分配方法は，ＰＤＢ上のデータを並列に検索する場合を想定すれば，頻繁に利用されることが予想される。

本稿では，ｆ制御に分配したコマンドの実行が失敗したときに限り，ｄ制御に分配する場合と，ｆ制御，ｄ制御へ同時に分配する場合についてコマンド識別子を与えることにする。前者に対しては識別子ｆ：ｄを設定し，次のように表わす。

```
f:d.…. sys.com(Arg, Ret, Ctrl)
f:d.…. usr.com(Arg, Ret, Ctrl)
```

但し，…は識別子の列を表わす

後者に対しては識別子ｆ＾ｄを設定し，次のように表わす。

```
f^d.…. sys.com(Arg, Ret, Ctrl)
f^d.…. usr.com(Arg, Ret, Ctrl)
```

但し，…は識別子の列を表わす

また，識別子ｆ：ｄ，ｆ＾ｄは識別子ｄと同様重ねて指定することもできる。

さらに，ｄ制御に分配されるコマンドは，ｄ制御によりその第２成分であるｄノードのストリーム違に分配される。このとき，特定の要素に対してコマンドを送りたい場合もありうる。そこで，リスト上の要素の位置を示すことによって，コマンドの分配先を決めることにする。位置指定に際しては，コマンドを分配するものと，しないものとに分け，前者に対しては，リスト上の位置を番号で示し，後者に対しては，その番号の前に／を置いて指示する。さらに，複数の指定も許すことにする。但し，指定がなければ全てのｄノードへ分配する。具体的には，識別子ｄ，ｆ：ｄ，ｆ＾ｄのｄに位置指定を書き，例えば，次のように表わす。

```
d([1,2,3]).….sys.com(Arg, Ret, Ctrl)
f:d([/1,/2,/3]).….sys.com(Arg, Ret, Ctrl)
f^d([1,/2,3]).….sys.com(Arg, Ret, Ctrl)
```

## 4．操作例

本節では，与えられた構造をもつデータを，ＰＤＢ上に生成する具体例について説明する。いま，次のようなデータが与えられたとする。

```
ａ開発本部　manager:x1;
ｂ開発部　　manager:x2; member:[y1,…,y4];
ｃ開発部　　manager:x3; member:[y5,…,y8];
```

但し，ａはｂ，ｃから構成されているものとする。このときａ，ｂ，ｃに対しｄノードd(Ｓａ，ａ)，d(Ｓｂ，ｂ)，d(Ｓｃ，ｃ)を用意すれば，上記データは各ｄノードストリームに以下のようなコマンドを送ることによって生成される。

```
Sa ← sys.crfnd(manager.x1, _, _)
     d.sys.lnk(Sb.Sc, _, _)
Sb ← sys.crfnd(manager.x2, _, _)
     sys.crfnd(member.[y1,…,y4], _, _)
Sc ← sys.crfnd(manager.x3, _, _)
     sys.crfnd(member.[y5,…,y8], _, _)
```

また，d(Ｓａ，ａ)だけを用意した場合は以下のようなコマンドによって生成される。

```
Sa ← sys.crfnd(manager.x1, _, _)
   sys.crdnd(b, Sb, _)
   sys.crdnd(c, Sc, _)
   d.sys.lnk(Sb.Sc, _, _)
   d.sys.loc(b, Lb, _)
   d.sys.loc(c, Lc, _)
   d([Lb]).sys.crfnd(manager.x2, _, _)
   d([Lb]).sys.crfnd(member.[y1,…,y4], _, _)
   d([Lc]).sys.crfnd(manager.x3, _, _)
   d([Lc]).sys.crfnd(member.[y5,…,y8], _, _)
```

但し，上記コマンドは以下のような機能をもつ。

sys.crfnd(Name, Cont, Strm, Ctrl)

ノード名Name，データContであるｆノードを生成する。リターン値はｆノードストリームである。

sys.crdnd(Name, Strm, Ctrl)

ノード名Nameであるｆノードを生成する。リターン値はｆノードストリームである。

sys.lnk(S1.…Sn, success, Ctrl)

ｆ制御またはｄ制御のストリームリストにストリームS1,…Snを追加する。

sys.loc(Strm, Loc, Ctrl)

ｆ制御またはｄ制御のストリームリストに関するストリームStrmの位置情報Locを求める。

## 5．おわりに

本稿では，ＧＨＣのプロセスをデータの基本単位とするデータベースを考察した。これによれば，データの検索あるいは変更においてＧＨＣがもつ並列性を引き出すことができること，データベースの構造を動的に変更することができること，などを見てきた。今後はこのようなデータベースを様々な問題に適用して，その有効性を検証していく予定である。