拡張可能ＤＢＭＳにおける部品の管理と実行について

2Ｔ－8

増永 浩二　宝珍 輝尚　都司 達夫

福井大学 工学部 情報工学科

1　はじめに

最近、CAD/CAM、LSI設計、CASE、ネットワーク管理、マルチメディアシステムなど、様々な分野でデータベースを利用したいという要求が強まっており、ＤＢＭＳを拡張可能としておき、必要な機能のみを作成・付加することで、それぞれの分野に特化したＤＢＭＳを迅速に構築しようという拡張可能ＤＢＭＳの研究がある[1]。しかし、ＤＢＭＳを柔軟にして必要な機能を付加可能とすると、そのためのオーバヘッドが生じてしまう。特にＤＢＭＳの場合、データ数に応じた処理が必要となる機能に関しては、１呼び出し当りの性能オーバヘッドが大きくなくてもトータルでは大きなオーバヘッドとなることがあり、性能の良いＤＢＭＳ部品の実行が求められる。

そこで本論文では、部品をダイナミックリンク機能を用いて実行させることで動的な機能拡張を可能とし、かつ、部品のメモリ配置ならびに部品呼び出し方法を細かく制御することで、部品の実行性能オーバヘッドを抑制する方法について述べる。

以下、２で拡張可能ＤＢＭＳ：ＣＯＭＭＯＮについて概説し、３でＤＢＭＳ部品に対する要求を示す。４で部品の管理・実行方法を提案し、５で使用方法を示す。最後に６でまとめる。

2　拡張可能DBMS：COMMON

ＣＯＭＭＯＮでは、ＤＢＭＳ処理に必要な各々のデータをデータベースとして管理する特殊なＤＢＭＳをモジュールとしている。管理するデータの種類でモジュールを決定することと、操作をSELECT、INSERT、DELETE、UPDATEといった基本的な操作に留めることでモジュールの大きさの限定と、分かりやすさの向上を狙っている。

ＣＯＭＭＯＮにおける基本的な拡張方法は、モジュールに用意されている部品を選択することである。モジュールの機能はアプリケーションプログラムに最適な部品を選択することによって変更することができる。適切な部品が無い場合は、必要とする機能を持つ新しい部品を作成しなければならない。

[1] On the management and the execution of software parts in an extensible DBMS
Koji MASUNAGA, Teruhisa HOCHIN, Tatsuo TSUJI
Fukui University, 3-9-1 Bunkyo, Fukui-Shi 910-8507, Japan

3　要求

部品の管理・実行に関する要求を以下に示す。

［要求１］新たな機能をシステム停止・再起動させること無く、動的に追加・削除できること

［要求２］性能要求の厳しい部品は、実行オーバヘッドを少なくすること。

4　プログラム管理部の設計

上記の要求を実現するために、COMMONにおいてその機能を担当するプログラム管理部を作成した。

プログラム管理部の構成は、システム本体とのインターフェース部分、プログラム管理部の初期化・終了処理の部分、部品情報を登録・削除する部分、部品の実行に関する部分からなる。

プログラム管理部では、部品に関する情報として、部品の属するモジュールの名前、部品の名前、オブジェクトファイルのあるパス、部品に対応する関数名、部品の登録タイプを、ハッシュを用いて管理する。ここでは衝突を極力避けるために複数のハッシュ関数を用いている。また、部品の属するモジュールの名前、部品の名前をもとにハッシュ値を得ている。図１にその方法を示す。図１では、第１ハッシュ値が10、第２ハッシュ値が３の場合を示している。第１ハッシュ値を用いて、index1のエントリにアクセスし、第２ハッシュ値を持つエントリを捜す。このエントリのindex2の値を用いてindex2のエントリにアクセスし、求める部品情報を得る。

次に、部品の実行に関する処理について述べる。ここでは、部品のオープン、クローズ、実行を行なう。部品のオープンではダイナミックリンク関数を用いて、部品の登録のときに指定されたパスにあるオブジェクトファイルをオープンし、同じく指定された関数へのアドレスを部品情報に登録する処理を行う。この時点では、部品はまだメモリ上に配置されず、部品が初めて実行されるときに配置される。部品のクローズ要求が来るまで部品はメモリ上に配置される。

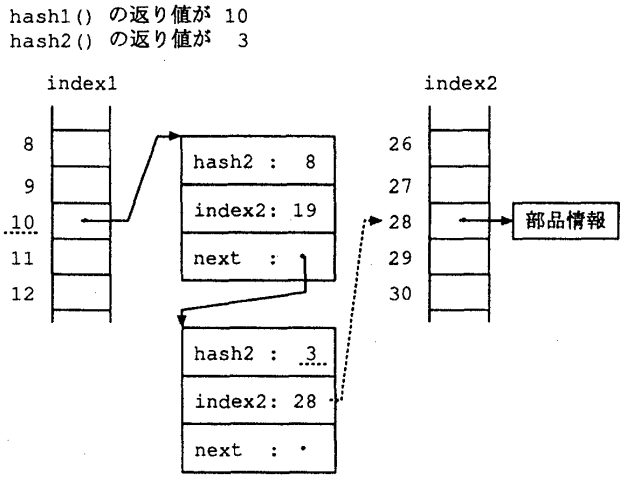

図 1: 複数のハッシュを用いた管理

部品の実行に関しては要求2を満足させるために、部品の実行を通常実行と高速実行の2種類に分けることにした。通常実行を行なう部品は、前述のようにハッシュを用いて管理する。ここでは、ハッシュ値の衝突が避けられないため、部品の属するモジュールの名前、部品の名前が合致しているか検査してから部品を実行することになる。一方、高速実行を行なう部品は、登録の際にハッシュ値が既存のものと衝突した場合、登録を許可しない。従って、このような場合、部品の属するモジュールの名前、部品の名前を変更する必要がある。しかし、ハッシュ値に衝突が無いことを保証するので、部品実行時に部品の属するモジュールの名前、部品の名前が確かに合致しているか検査する必要は無く、この検査を行わずに部品を実行する。

5 部品の使用方法

部品を登録・使用するときは、プログラム管理部に引数として、操作の種類と、構造体に登録・使用したい部品の情報 (部品の属するモジュールの名前、部品の名前など) と、部品を使用する場合は、部品に対する操作の種類、部品が使用する引数を格納したものを渡す。図2にプログラム管理部への引数に使う構造体の内容と、図3に部品の使用例を示す。図3では"datamng"というモジュールに"fetchindex"という部品を登録し、オープン、実行、クローズしている。

6 まとめ

本論文では、ＤＢＭＳに対して新たな機能の追加や削除を行ったときに、ＤＢＭＳ本体自体は再コンパイル、またはシステムの停止をすること無く柔軟に機能の変更の出来るプログラム管理部を設計した。

今後は、実際のＤＢＭＳへの適用が課題である。

```
manager    : 部品が属するモジュール名
parts      : 部品の名前
path       : オブジェクトファイルがあるパス
function   : 部品に対応する関数名
entrytype  : 部品の登録タイプ
pknd       : 部品の操作の種類
parg       : 部品が使用する引数を格納した構造体へのポインタ
```

図 2: 引数の内容

```
/* 部品の登録 */
pmarg1.manager   = "datamng";
pmarg1.parts     = "fetchindex";
pmarg1.path      = "../../datamng/datamng.o"
pmarg1.function  = "datafetidx"
pmarg1.entrytype = NORMAL;

progmng( INSERT, &pmarg1 );

/* 部品のオープン */
pmarg2.manager   = "datamng";
pmarg2.parts     = "fetchindex";
pmarg2.entrytype = NORMAL;

progmng( OPEN, &pmarg2 );

/* 部品の実行 */
pmarg3.manager  = "datamng";
pmarg3.parts    = "fetchindex";
pmarg3.pknd     = INIT;
pmarg3.parg     = &dataarg;
    /* dataarg は部品が使用する引数を格納した構造体 */
progmng( EXECUTE, &pmarg3 );  /* 通常実行 */
progmng( FEXECUTE, &pmarg3 ); /* 高速実行の場合 */

/* 部品のクローズ */
progmng( CLOSE, &pmarg2 );
```

図 3: 使用例

謝辞

本研究は、一部、文部省科学研究費 (課題番号09780258) による。

参考文献

[1] Hochin, T. and Inoue, U. : "An Extensible DBMS Composed of Specific DBMSs," Proc. of Int's Sympo. on Next Generation Database Systems and Their Appl. , pp.180-187 (1993).