並列データベース管理システム Kappa-P の概要

河村 元夫[1]、 佐藤 裕幸[2]、 横田 一正[1]

1: (財) 新世代コンピュータ技術開発機構　　2: 三菱電機 (株) 情報電子研究所

1 はじめに

第五世代コンピュータプロジェクトでは，知識情報処理システム (KIPS) の中核的機能であるデータベース，知識ベース管理機能を提供することを目的とし知識ベース管理システム (KBMS) の研究開発を行ってきた [1]．そのなかで，Kappa (Knowledge Application-oriented Advanced Database and Knowledge Base Management System) [2, 3] は，KIPS や KBMS のデータベース エンジンの役割を果たし，知識表現言語 / 知識ベース言語 *Quixote* はその上位層にあたる．中期に逐次推論マシン PSI で動作する逐次データベース管理システム (DBMS) Kappa-II を開発し，後期には並列推論マシンとそのオペレーティング システム PIM/PIMOS の環境で動作する並列 DBMS Kappa-P の試作をおこない，現在初版が動作している．本稿では，Kappa-P の設計方針と特徴を述べる．

2 設計方針

KIPS が扱うデータや知識は複雑な構造をしており，かつ量も膨大なものがある．たとえば，遺伝子情報処理の分子生物学データベース GenBank/HGIR database[4] は，塩基配列と特徴の記述と関連する文献情報からなり，この塩基配列は短いものから非常に長いものまでさまざまである．このような複雑なデータを伝統的な関係モデルで扱うには，データ表現能力の点と効率的な問合せ処理の点で問題が多い．また，このデータベースは，解析技術の進歩にともない近年飛躍的に増大している．さらに，このデータに対する操作は，類似検索など計算能力を必要とするものが多く，そのデータ量と計算量から，並列マシンの能力が必要になっている．

Kappa-P が動作する環境は，並列推論マシン PIM とそのオペレーティング システム PIMOS の環境である．PIM は MIMD 型の疎結合と密結合が混合されたハイブリッド並列マシンある．10 台程度の要素プロセッサが，共有バス / 共有メモリにより結合され一つのクラスタとなり，それが，ネットワークで結ばれている．共有メモリは，数百メガ バイトで，ディスクは各クラスタに取り付けることができる．

このような背景のもと，つぎの設計方針で設計された．

- データモデル
  - 複雑な構造データを扱えること
  - 大量データに対する考慮
- ハードウエア資源の有効利用
  - 疎結合並列処理，密結合並列処理の両方を考慮
  - 大容量主記憶を生かした主記憶データベース機能
  - ディスクに対する並列アクセス
- 並列マシン上の DBMS
  - 応用プログラムとの通信量の削減

Overview of the Parallel Database Management System: Kappa-P
Moto KAWAMURA[1], Hiroyuki SATO[2], Kazumasa YOKOTA[1]
1:Institute for New Generation Computer Technology
2:Mitsubishi Electric Corporation.

3 特徴

前節の方針に基づき，PSI 上で実装された自然言語処理システムや遺伝子情報処理システムで有効利用されている Kappa-II の経験を最大限に生かして，Kappa-P を設計した．つぎのような特徴がある．

- 非正規関係モデル

複雑な構造データを効率的に扱うために非正規関係モデルを採用する．これは，関係モデルの自然な拡張になっており，*Quixote* の無限構造をもたないオブジェクト項のクラスに対応する．

データ構造としては，属性が階層構造を持ち，値として繰り返し値が許されることが，関係モデルとの違いである．また，その意味論として，非正規関係に固有の操作である行ネスト操作に対し独立した意味を与えている．つまり，非正規関係 $\{[a/c_1, b/\{c_2, c_3\}], [a/c_1, b/\{c_3, c_4\}]\}$ と $\{[a/c_1, b/\{c_2, c_3, c_4\}]\}$ は同じ意味を持っている．これにより，ネスト構造を意識しない問合せが可能となる．先の非正規関係に対し問合せ ?−$[a/X, b/\{c_2, c_4\}]$ を出すと，どちらも $X = c_1$ が結果として得られる．これは，意味論としては基本的に関係に基づきながら表現 (および蓄積構造) の効率化をめざしているという点で，関係モデルの自然な拡張であり，多値従属性をもつデータの表現と処理の効率化が可能である．この意味を反映して，関係代数を，行，列のネスト，アンネスト操作を含む拡張関係代数として定義しなおした．

また，さまざまな知識を格納できるようにデータ型としてタームを追加し，大量データに対する検索処理の効率化を目的に，索引としてのみ存在する属性なども追加した．

- 密結合並列と疎結合並列を考慮した構成

DBMS は大量のデータを扱うので，疎結合並列処理における通信量が大きな意味を持ってくる．そのため，密接に関連するデータを同じクラスタに置くなどのデータの配置が重要で，さらに，問合せ処理において通信量が少なくなるようなプランを立てる必要もある．これらをおこなうためには，分散データベースの構成が向いている．

図 1に，Kappa-P の全体構成を示す．各クラスタにローカル DBMS(LDBMS) と呼ばれる DBMS を配置し，全体で一つのデータベースを管理する．この LDBMS は，それ自身で DBMS としての全機能をもち，複数 LDBMS が関与する問合せを処理するために，二相コミットプロトコルに基づく分散トランザクションの機能を持っている [7]．また，複数の LDBMS により一つのデータベースを管理するためテーブル名などの大域情報の管理が問題になるが，それを管理するサーバ DBMS (SBDMS) の複製を作ることにより，アクセスの集中を回避する [6]．クラスタに割り当てられた，ローカル DBMS は，その内部処理で密結合向きの並列処理をおこなう．

問合せ処理は，インタフェース プロセス [5] と呼ばれるプロセスを介して行われる．アプリケーションが並列言語で記

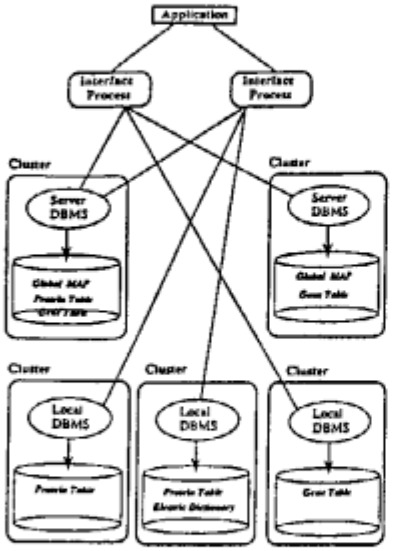

図 1: Kappa-P の全体構成

述されるため，一つのアプリケーションが複数のインタフェース プロセスを利用できるようにしている．また，このインタフェース プロセスが，問合せ処理時に必要となる各種の変換をおこない該当 LDBMS に対する部分問合せに変換する．

• データの配置

データの配置は，並列処理に関係する事柄である．

クラスタ間並列処理をおこなうためには，複数の LDBMS にデータを分散配置する必要がある．これには，単純にテーブルを分散配置すること，テーブルの水平分割，テーブルの複製などがある．テーブルの水平分割は，一つのクラスタでは扱い切れないような大量のデータや特にデータ検索の速度を重視する場合などに有効である．しかし，すべての問合せに有効ではなく，逆に遅くなる場合もあるので注意しなければならない．また，テーブルの複製に関しては，現在の実装では，サーバ DBMS の大域情報管理でのみ実現されている．

クラスタに割り当てられる LDBMS 内でデータを主記憶に置くか二次記憶に置くかの選択ができる．これは，PIM の各クラスタがもつ数百メガ バイトの大容量主記憶を有効利用するための機能である．主記憶上のテーブルは，二次記憶とは無関係な一時テーブルであるが，多数の中間結果テーブルを生成するシステム，たとえば演繹データベースなどでは，これが有効である．

この主記憶上の一時テーブルを利用して，擬似主記憶データベースの機能も提供する．これは，二次記憶テーブルの複製として主記憶上の一時テーブルを作り，読み系操作は一時テーブルに対しおこない，更新系操作は両方のテーブルに対して行う．これにより，更新の二次記憶への反映を保証する主記憶データベースに近い効果をねらっている．更新時は，両テーブルに対するアクセスになるが，これにはクラスタ内並列処理が有効に働く．

• 問合せ処理

Kappa-P は原始コマンドと *KQL* の二種類のコマンドを提供している．

原始コマンドは，非正規関係のためのプリミティブな操作を提供している．それは，レコード識別子によるポインタ操作を基本にした操作で，*KQL* に比べ低レベルではあるが効率的な操作が可能である．非正規関係は属性値に繰り返しが許されるため，このレコード識別子も単純な構造ではなく，このレコード識別子に対応するレコードを得るにはアンネスト / ネスト操作 [9] が必要になる．

KQL は，拡張関係代数からなる式の集まりからなり，問合せ中に一時的な操作を定義したり，推移閉包を処理するためのループを記述したりできる．この KQL は，一つのトランザクション単位でまとめて受けとられ，各種最適化され，拡張関係代数処理のための中間言語 [8] への変換され，通信量を考慮し各ローカル DBMS への部分問合せに分割され，各ローカル DBMS に送られ実行される．

• Kappa-II との互換性

PIM は，そのフロント エンド プロセッサとして PSI を使っている．その PSI 上の Kappa-II 用として開発されたプログラムを利用できるようにするため，Kappa-II と互換性をもつプログラムインタフェースも提供している．これは，並列処理向きのストリームを基本にしたインタフェースを，逐次言語向きにしたものである [10]．

4 まとめ

Kappa-P は現在初版が PIM/m 上で動作しており，蛋白質データベースを格納し，そのデータベース用に Kappa-II 上で開発されたアプリケーションが動作している．また，水平分割テーブルによる並列効果も得れら，十分な性能で動作することがわかった．今後，計測結果に基づく評価，システムの改良と安定化，複数 LDBMS を利用した問合せ処理実験などをおこなう予定である．

参考文献

[1] K. Yokota and H. Yasukawa, "Towards an Integrated Knowledge-Base Management System — Overview of R&D on Databases and Knowledge-Bases in the FGCS Project", *FGCS'92*, 1992.

[2] K. Yokota, M. Kawamura and A. Kanaegami, "Overview of the Knowledge Base Management System (Kappa)", *FGCS'88*, 1988.

[3] M. Kawamura, H. Sato, K. Naganuma, and K. Yokota, "Parallel Database Management System: Kappa-P", *FGCS'92*, 1992.

[4] "GenBank/HGIR Technical Manual", *LA-UR 88-3038*, Group T-10, MS-K710, Los Alamos National Laboratory, 1988.

[5] 永松哲か, "Kappa-P の並列問い合わせ処理", 第 45 回情処全国大会, 5R-04, 1992.

[6] 坂下哲か, "Kappa-P のネームサーバ機能", 第 45 回情処全国大会, 5R-05, 1992.

[7] 椿野哲か, "Kappa-P のトランザクション制御", 第 45 回情処全国大会, 5R-06, 1992.

[8] 藤部哲か, "Kappa-P の中間言語処理", 第 45 回情処全国大会, 5R-07, 1992.

[9] 川村哲か, "Kappa-P のアンネスト / ネスト処理", 第 45 回情処全国大会, 5R-08, 1992.

[10] 中嶋哲か, "Kappa-P の単一レコード・アクセス機能", 第 45 回情処全国大会, 5R-09, 1992.

Kappa-P の並列問い合わせ処理

永沼和智[1]，佐藤裕幸[1]，河村元夫[2]

1: 三菱電機(株) 情報電子研究所　2: (財) 新世代コンピュータ技術開発機構

1 はじめに

第5世代コンピュータプロジェクトでは、大規模知識処理を目的として、並列推論マシン PIM [1] 及びその上で動作する並列知識処理システムの研究開発を行っている。Kappa-P [2] は、これらの知識処理システムに大量の複雑なデータの効率的な処理を提供するために開発された、並列非正規関係データベース管理システムである。Kappa-P は、PIM の各クラスタに配置されたローカル DBMS と呼ばれる複数のデータベース管理システムから構成されており、クラスタ間での並列性を得るために、一つのテーブルを水平に分割し複数のローカル DBMS に配置することができる。そのテーブルへの問い合わせは、インタフェースプロセスにより構成テーブルへの並列な問い合わせに置き換えられ、各ローカル DBMS により並列に処理される。本論文では、この水平分割テーブル処理、また分割テーブルの欠点である通信コストの増大を防ぐための方法を、実測値と共に報告する。

2 問い合わせ処理概要

2.1 処理の流れ

Kappa-P では、原始コマンドと呼ばれるプログラムインタフェースを提供している。原始コマンドとは、一つのテーブルに対して読みや更新の処理を行うコマンドで、インタフェースプロセスがユーザからコマンドを受ける。インタフェースプロセスは、コマンドを受けるとサーバと通信し、指定されたテーブルがどのローカル DBMS に存在するかと言う情報を得る。サーバは、データベースのグローバル情報を管理しているプロセスである。そして、インタフェースプロセスはそのローカル DBMS 情報をコマンドに付加し、コーディネータにその情報を送る。その情報によりコーディネータは、各ローカル DBMS へコマンドを送り、各ローカル DBMS でコマンドが処理される。
(図 1参照)

2.2 水平分割テーブル

水平分割テーブルとは、1つのテーブルをその属性値などにより複数の構成テーブルにレコード単位で分割し、複数のローカル DBMS に配置したものである。この分割方法には、属性値の範囲による分割、属性値のハッシュ値による分割、ラウンドロビンの3種類がある。属性の範囲による分割はその属性を条件に指定された時に有効であるが、構成テーブルのレコード数がまちまちになってしまうと言う欠点があり、これに対しラウンドロビン

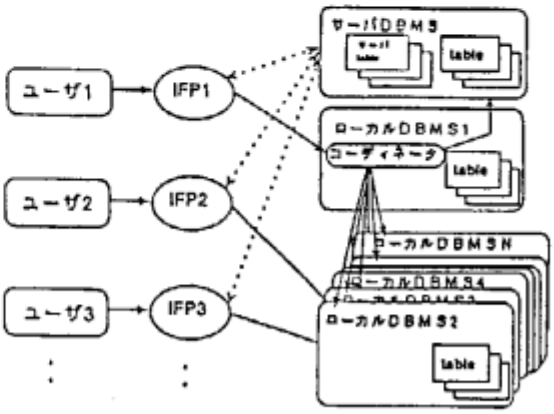

図 1: 問い合わせ処理

は構成テーブルの大きさが均一になるので、各テーブルに対する処理時間が均一になりやすい。これらは、そのデータに対する検索を考えて選択する必要がある。

水平分割テーブルに対する検索は、構成テーブルの存在する各ローカル DBMS で並列に処理されるが、ユーザは分割テーブルであることや構成テーブルの所在などを意識せずに一つのテーブルとしてに扱うことができる。ユーザから水平分割テーブルに対するコマンドが出されると、インタフェースプロセスはサーバとの通信により、そのテーブルの各構成テーブルの名前と存在するローカル DBMS を知る。そして、その情報によりコマンドは各ローカル DBMS に対する複数のコマンドに変換され、コーディネータ経由で各ローカル DBMS で並列に実行される。このときレコードの追加や削除のコマンドは、追加、削除するべき構成テーブルを分割情報から判断しその構成テーブルへのコマンドに変換される。

2.3 フィルタ付きレコード読み

テーブルを分割した場合にローカル DBMS 間の並列性は得られるが、その分ローカル DBMS 間の通信コストが増大すると言う欠点がある。特にユーザがデータを一度読み込み、そのデータに対して PIM の各クラスタを使って並列に処理を行いたい場合には、一箇所にデータが集められてから再び分散させ、そのデータに対する処理を行った結果をもう一度集めるために、かなりの通信コストがかかる。この通信コストの増大を防ぐために、Kappa-P ではフィルタ付きレコード読みの機能を提供している。このフィルタ付きレコード読みでは、ユーザがその検索の結果に行う自分で定義した処理(フィルタ)を指定することにより、その処理が分割された構成テーブルのあるクラスタで実行され、その処理の結果がユーザのいるクラスタに集められる。このため、一度データを

Parallel query processing on Kappa-P

Kazutomo NAGANUMA[1], Hiroyuki SATO[1], Moto KAWAMURA[2]
1:Mitsubishi Electric Corporation　2:Institute for New Generation Computer Technology.

集めてから分散させると言うことがなくなりまた、一回の通信データもフィルタを通った後のものになるので、通信コストはかなり軽減される。

3 評価

実際に一つのテーブルを分割した場合に、どれだけの速度向上が見られるのか、64台のプロセサを持つPIM/mを使って測定した。

3.1 測定方法

測定の対象としたデータは、公共のデータベースPIRの遺伝子データである。このテーブルのデータは約61メガあり、3万3千のレコードを持っている。このテーブルを16,32,64に分割し、それぞれ次の測定を行った。なお、ディスクの数が少ないため必要なデータはあらかじめ主記憶に読み込んでから測定を行ったが、分割しないテーブルは主記憶の容量が足りないために実行できなかった。

(1) 属性sequenceがCHという文字列を含むレコードの検索。
この検索では、約4119件のレコードが結果として得られる。

(2) フィルタ付きレコード読み。
指定したフィルタは、属性sequenceがある一定のパターンの文字列を含むレコードを検索するモチーフサーチである。この検索では、47件のレコードが結果として得られる。

(3) フィルタ付きレコード読みを使わずに、一度レコードを読み込んでからその結果に対して(2)と同じフィルタを並列に実行する。

3.2 測定結果 / 考察

測定結果を表1に示す。

(1)の測定結果では、32分割の実行時間が16分割のテーブルの2分の1に近い数値となっている。これは、テーブルの大きさが2分の1となりそれがそのまま実行時間に反映している。実際は通信時間、前処理、後処理の時間は2分の1とはならないので、厳密には2分の1よりも多少大きな数値となっている。

しかし64分割の場合には、32分割の2倍とはならずに約1.6倍となっている。これは、一つの分割テーブルに対する処理が非常に軽くなったため、通信にかかる時間が無視できなくなったと言うこと、各ローカルDBMSで行われる検索の前のジョブやトランザクションの生成、テーブルのロック、検索の後の終了処理などにかかる時間は分割テーブルであろうと普通のテーブルであろうと一定であるため、実際の検索時間が短くなるとこれらの時間の占める割合も無視できなくなったと考えられる。

(2)の測定では、32分割テーブルの実行時間は16分割の約2分の1となっており、64分割の実行時間は約4分の1となっている。これは、(1)のレコード検索と比べここで指定したモチーフサーチの負荷がかなり高く実行時間もかかっているため、先に述べたような通信、前処理、後処理にかかる時間が無視できる範囲であったと考えられ、分割の効果が十分表れていると言える。

(3)の測定結果を(2)と比べてみるとそれぞれ約1.5倍から、2倍近い時間がかかっている。フィルタ付きレコード読みを使わずに同じ処理をする場合には、フィルタの処理に入る前のレコード読みの結果が一度ユーザのいるクラスタに集められてから、再び分散することになる。この通信コストがそのまま数字に表れていると考えられ、フィルタ付きレコード読みの有効性が確認された。

テーブル分割数	16	32	64
(1) レコード検索	33,331	18,158	11,575
(2) フィルタ付き レコード検索	132,447	68,892	37,664
(3) レコード検索 + フィルタ	206,549	101,173	66,190

表1: 各テーブルの実行時間(msec)

この測定結果から、(1)の前方一致、後方一致と言うような検索に対しては、このテーブルを64以上に分割してもそれほどの効果が得られていない。これに対し(2)のモチーフ検索では、64分割でも十分な効果が得られており、更に分割した場合の効果も期待できる。このように、検索の負荷の大きさにより適した構成テーブルの大きさは異なり、そのデータの使用目的のより分割数を決定することが必要である。

4 おわりに

今回測定に使用した並列推論マシンPIMは64台のプロセサを持っているが、現在ICOTでは256台版のPIMが動作している。また、Kappa-Pは大量で複雑なデータを扱うことを目的としており、今後は今回扱った以上に大量で複雑のデータに対して負荷の大きい処理が要求されると考えられる。そのような要求に対して水平分割テーブルは非常に有効な手段であることが今回の測定により確認できた。ただ、データの複雑さ、データに対する処理の負荷の大きさにより、最適な分割テーブル数が異なるため、今後は分割をする際の目安になるような解析をする必要があると思われる。

参考文献

[1] 森: *Parallel Inference Machine PIM*, The International Conference on Fifth Generation Computer Systems'92,1992.

[2] 河村他: 並列データベース管理システムKappa-Pの概要, 第45回情処全国大会,5R-3,1992.10.

[3] 横田他: *Overview of the Knowledge Base Management System(KAPPA)*, The International Conference on Fifth Generation Computer Systems'88,1988.

Kappa-P のネームサーバ機能

坂下之子[1]， 合田光宏[1]， 佐藤裕幸[2]， 河村元夫[3]，

1: (株) アーティフィッシャル・インテリジェンス　　2: 三菱電機 (株) 情報電子研究所
3: (財) 新世代コンピュータ技術開発機構

1　はじめに

Kappa-P [1] は、 ICOT で開発された並列推論マシン PIM [2] 上で動作する並列データベース管理システムである。

Kappa-P は、複数のローカルデータベース管理システム (LDBMS) から構成される。 LDBMS は PIM の各クラスタに配置され、それ自体が完全なデータベース管理機能を持っている。また、データモデルとして非正規関係型モデルを採用し、複雑な知識データを格納、検索する機能がある。それらを複数、並列に動作させることにより、大量の知識データを効率的に処理することができるようになる。このためには、テーブル情報の管理が重要になる。

Kappa-P では、テーブル情報を一元管理するネームサーバ機能を実装した。また、ネームサーバへのアクセス集中を避けるため、ネームサーバの複製を可能にした。本論文では、この Kappa-P のネームサーバ機能の実装について、複製とトランザクション処理を中心に報告する。

2　ネームサーバの設計方針

Kappa-P のネームサーバ機能の設計方針は、以下の 2 点である。

- テーブルの位置をユーザに意識させない。
- アクセス集中を避けるためと耐故障性の向上のため、ネームサーバの複製を作る。

Kappa-P は、複数の LDBMS と水平分割テーブルの機能を提供している。これらはデータアクセスの並列化を目指したものである。しかし、ユーザの立場から考えると、 DB のアクセスは複雑にしたくない。そこで、存在位置に関係なく、DB 内ではテーブル名をユニークにつけることにし、テーブルの位置情報は、内部的にネームサーバから得ることにする。これにより、ユーザは、テーブルの位置情報を意識することなく、 DB にアクセスすることができる。ネームサーバ機能のある LDBMS をサーバ DBMS (SDBMS) と呼ぶ。

テーブル情報を一元管理しているネームサーバがただ一つのときは、ネームサーバにアクセスが集中し、それが全体の並列性を妨げるボトルネックになってしまう。またその場合、 SDBMS に障害が起きたときには、データベース全体がアクセス不可能になる。このような事態を避けるため、異なるクラスタにネームサーバの複製を置き、ある程度は、データベースの動作を保証することを考える。別クラスタで並列に動くのであれば、複製の増加がネームサーバへのアクセス実行に与える影響は小さいと考えられる。

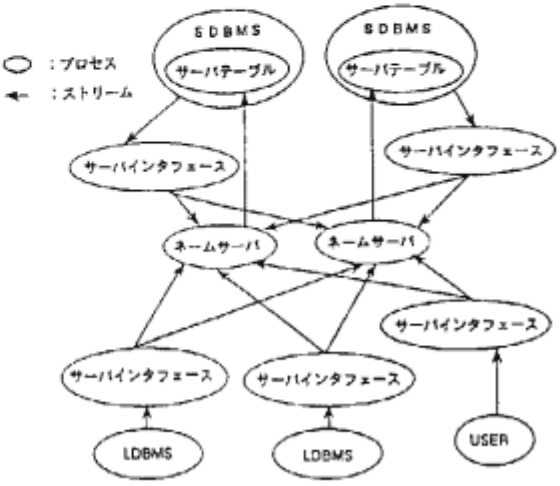

図 1: サーバの構成

そこで DB の複製方法として、 Special copy (primary copy 、 available copy)、 Voting などのアルゴリズムを検討したが、 Special copy による複製の場合、対応できる障害が限られていることが問題となった。また primary copy は読みだしの集中が問題となり、 available copy は実装が複雑になることが問題となった。その結果、対処できる障害に制限がなく、実装が比較的簡単な Voting [3] を採用することにした。

3　サーバの構成と複製方式

Voting のため、個々のネームサーバへの通信路を持ったサーバインタフェースをつくる (図 1)。テーブル情報の登録・更新、問い合わせなどのネームサーバ機能は全て、サーバインタフェースを通して行われる。これにより、ネームサーバが幾つ存在するかなどを意識することなしに、処理ができるようになる。

ネームサーバ内では、テーブル情報はテーブル名で管理され、バージョン番号が付けられる。一度ネームサーバに登録されたあとは、そのテーブル名の情報が更新される度にバージョン番号がインクリメントされる。サーバインタフェースは、複数のネームサーバに問い合わせた結果が異なっている場合、バージョン番号の最も新しいものを正しい情報と判断する。削除されたテーブルは削除情報で識別し、バージョン番号はインクリメントされて保存される。これは、最新情報としてテーブルが削除されていることを示す。

Name service function on Kappa-P
Yukiko SAKASHITA[1], Mitsuhiro GOUDA[1], Hiroyuki SATO[2], Moto KAWAMURA[3]
1:Artificial Inteligence Co.　2:Mitsubishi Electric Corporation
3:Institute for New Generation Computer Technology.

Voting のためのネームサーバの数は以下のように決定される。ネームサーバがN個あり、各ネームサーバに1票づつ割り当てる。読みに対して、R個のサーバの読みが必要であり、更新に対してW個のサーバの更新が必要である、とすると、R + W = N+1 の式を満たす数である。

サーバインタフェースからの Voting のアルゴリズムは、以下のようになる。

問い合わせ： サーバインタフェースは、各ネームサーバに対してテーブルの情報を尋ね、R個以上のネームサーバから返答が返ってきたとき、そのなかでバージョン番号の一番新しいものを最新情報として扱うことにする。

登録 / 更新： サーバインタフェースは、ネームサーバにテーブル情報を問い合わせ、その最新情報から登録 / 更新可能であると判断したら、最新のバージョン番号をインクリメントして更新情報を作り、各ネームサーバにメッセージを送る。W個以上のネームサーバから登録成功の答えが返ってきたら、その登録を成功とする。

これにより、最低R個の SDBMS が存在するかぎりは、ネームサーバからとりだした情報を信用することができる。

4 実装

実際のネームサーバは、 SDBMS のテーブルを利用して情報を保持、提供する。一つのネームサーバには、一つのサーバテーブルが対応している。DB がアクセス可能な状態の間は、ネームサーバ以外のユーザからアクセスできないように、これを排他ロックしている。サーバテーブルの1レコードには、1つのテーブル名についての情報が格納されている。

以下に、実装されたネームサーバのトランザクション管理と同時実行制御について報告し、問題になった点について述べる。

4.1 トランザクション管理

Kappa-P の LDBMS は、2相コミットによるトランザクション管理 [4] や、リカバリ処理を提供している。Voting で必要なトランザクション管理は、2相コミットと異なる。

一般の2相コミットでは、指定したテーブルのうち1つでも失敗すれば正常終了できない。しかし Voting 用のトランザクションでは、指定したテーブルのうち必要数のテーブルがロック / コミットできればよい。

このため Voting のトランザクションに2相コミットをそのまま使用すると、アルゴリズムとは異なり、本来書き込みが正常終了するものが、エラーになる場合がある。アルゴリズムでは、必要数のサーバテーブルに書き込めれば、サーバテーブルへの登録 / 更新は成功するのだが、2相コミットでは、ロックしたテーブル全ての書き込みが成功しなければ、トランザクションは失敗する。しかし、このような障害発生はまれなので、実用上問題ないため、現在の実装では2相コミットを少し修正して使っている。

一般のトランザクションからの変更は次の点である。

- 必要数のテーブルがロックできればトランザクションが開始できる。

トランザクション開始以後は、2相コミットプロトコルに従う。これによりロックできたテーブルが全部書き込めない限り、サーバテーブルへの登録 / 更新はできない。

読み出しについてはアルゴリズム通り、必要数のサーバテーブルが読めた時点で、 Voting を行なっている。

4.2 同時実行制御

現在の LDBMS ではレコード単位の同時実行制御は行っていない。上で述べたように、テーブル情報の更新の要請が来たとき、サーバインタフェースでは、 Voting に必要な数のサーバテーブルを、トランザクションで排他ロックしている。実際に排他制御が必要なのは、ユーザが指定したテーブル名の情報があるレコードだけであるが、現在の実装では、テーブル単位に排他制御されており、このことが並列処理の妨げになっている。

さらにもう一つの問題は、ロックするサーバテーブルの選び方である。実装は、全てのサーバテーブルをロックしてみて、成功したテーブルは全部使っているが、必要数より多くのテーブルに対してアクセスしていることもある。また、どのサーバテーブルを使うかという選択 (SDBMS の負荷の分散化など) も考慮する必要がある。

5 まとめ

以上のネームサーバの機能により、複数の LDBMS とそこに配置されたテーブルを、位置情報の指定なしに扱うことができるようになった。また、ネームサーバの複製により、アクセスの集中の緩和と安全性を実現することができた。が、幾つかの問題点も残っている。今後の課題としては、以下の点が挙げられる。

- Voting 向きトランザクションの検討
- ネームサーバの選択の最適化、分散化
- ロックするサーバテーブル数の最少化

これらを解決すれば、より効率的なネームサーバを作ることができる。

最後に、実装と本論文の作成にあたって、さまざまな助言、御指導をいただいた ICOT の方々、 Kappa-P の関係者の方々に深く感謝いたします。

参考文献

[1] 阿村他: 並列データベース管理システム Kappa-P の概要, 第45回情処全国大会, 5R-3, 1992-10.
[2] 編 : "Parallel Inference Machine PIM" The International Conference on Fifth Generation Computer Systems '92, 1992.
[3] S.H.Son, "Replicated Data Management in Distributed - Database Systems", SIGMOD RECORD, Vol.17, No.4, December 1988.
[4] 柳野他: 並列データベース管理システム Kappa-P のトランザクション制御, 第45回情処全国大会,5R-6,1992-10.

Kappa-P のトランザクション制御

椿野宣行[1]．　佐藤裕幸[2]．　河村元夫[3]

1: 三菱電機東部コンピュータシステム（株）　　2: 三菱電機(株) 情報電子研究所
3: (財) 新世代コンピュータ技術開発機構

1　はじめに

Kappa-P[1] は、ICOT で研究開発された並列推論マシン PIM[2] 上で動作する並列データベース管理システムである。Kappa-P の構成は、複数の独立したデータベース管理システム (ローカル DBMS) を一つのデータベース管理システムとするいわゆる分散データベース管理システムの構成となっている。このローカル DBMS は PIM の一つのクラスタに割り当てられている。

本システムは疎結合並列と密結合並列の両方を考慮している。この中のトランザクションは、ユーザから見た一つのトランザクションを複数のローカル DBMS のサブトランザクションの集まりとすることで疎結合並列処理を実現している。またローカル DBMS はクラスタに割り当てられるので一つのローカル DBMS 内の個々のトランザクションもまた並列に実行される。

本発表では、Kappa-P が用いた密結合並列を考慮した同時実行制御および一貫性の保証のためコミットメント制御について述べる。

2　トランザクション管理の概要

トランザクションとは一般に複数の I／O から構成される分割できないひとまとまりの処理であり、一貫性の保証のため同時実行制御とコミットメント制御が必要である。Kappa-P では、トランザクション管理として次のような同時実行制御とコミットメント制御をおこなっている。

2.1　同時実行制御

同時実行制御については、知識情報処理の環境ではタプル単位等の細かい制御の必要性があまりなかったこと、同じ資源 (テーブル) に対する使用要求の数はそれほど多くないことなどの理由により、テーブル単位の排他制御のみおこなうこととした。すなわち、トランザクション開始時に、read_only か exclusive でテーブルをロックし、トランザクション終了時にアンロックするというものである。この時問題となるのは、テーブルの施錠の管理方法である。Kappa-P の構成だと全体もしくはローカル DBMS 単位での集中管理が考えられるが、その場合実行時にボトルネックになり並列性が阻害される可能性がある。このため各テーブル自身が管理を行なう分散ロック方式を用いている。

2.2　コミットメント制御

トランザクションのコミットメント制御は、1 トランザクションが複数ローカル DBMS のサブトランザクションの集まりからなるため、分散コミットメントを行う必要がある。分散コミットメントには、2 相コミット [3] やブロック状態を減らすための 3 相コミット [3] などが知られている。対象である PIM は、分散システムではなく並列マシン (しかもプロトタイプ) であり、3 相コミットによりブロック状態を減らす必要性は大きくなかった。このため分散システムとしての一貫性の保証に重点をおき、分散コミットメントを最低限保証するプロトコルとして、2 相コミットプロトコルの中で最も単純なものを採用した。

3　分散ロック

テーブルの施錠管理を各テーブル自身がおこなう分散ロック方式では、ボトルネックが解消できる反面デッドロックの検出とその解消が問題となってくる。このデッドロックの問題を Kappa-P では、時間監視により検出しリトライを行なうことで解消することとしている。Kappa-P でのトランザクションでは分散ロックを次のように行なっている。

3.1　テーブルのロック

ロック要求は各サブトランザクションが受け付け、各テーブルにロック要求を出す。各テーブルはキューを持ちロック要求を受け付ける。すなわちアンロック状態になったらキューの先頭の要求がロックできたこととなる。しかしある時間待って全てのテーブルがロックできない場合は、ロック要求を取り消しリトライ (3.2) をおこなう。トランザクションは全てのサブトランザクションでのロックが完了してから開始可能となる。またトランザクション開始後は新たにロックできない。

水平分割テーブルに関しては、構成テーブルを一度に全てロックしようとするとデッドロックになる可能性が高いので、代表テーブルを決めて最初はそれのみをロックし、その後で残りの全テーブルをまとめてロックするようにしている。

Transaction Management on Kappa-P
Noriyuki TSUBAKINO[1], Hiroyuki SATO[2], Moto KAWAMURA[3]
1:Mitsubishi Electric Computer System(Tokyo) Corporation　2:Mitsubishi Electric Corporation　3:Institute for New Generation Computer Technology.

3.2 リトライ

サブトランザクションの中で一つでも開始できないサブトランザクションがあった場合は、開始できた全てのサブトランザクションを無効にした後ランダムな時間待ってリトライを行なう。ランダムな時間にすることで同一テーブルをロックしようとした他のトランザクションとのデッドロックを避けている。

またリトライする場合は、デッドロック監視時間を少し増やすことにより、多くのテーブルを使用するトランザクションが不利にならないようにしている。

4 2相コミットプロトコル

一貫性の保証とできるだけ並列性を阻害しないためテーブルをロックしている時間をできるだけ短くすることを考慮している。トランザクションは以下の順で実行される。

(1) テーブルの分散ロック
サブトランザクションを生成しテーブルのロックを行なう。

(2) トランザクションの開始
複数のサブトランザクションを取りまとめるためのサブトランザクション（コーディネータ）を一つ決定する。これがトランザクションの開始宣言でもある。コーディネータは、全サブトランザクションの監視、各サブトランザクションへのコマンドの振り分けなどを行なう。

(3) コマンドの実行
テーブルをアクセスするコマンドには検索系、読み系、更新系などがあり全てコーディネータが一括して受け付ける。受け付けたコマンドは各サブトランザクションに渡され実行される。コマンドの形式は以下の通りである。

コマンド：{サブコマンド, サブコマンド, ...}
サブコマンド：各サブトランザクションに対するコマンドである。

(4) トランザクションの終了
コーディネータは、2相コミットプロトコルに従って全サブトランザクションをコミットする。

4.1 コマンドのシリアライズ

コーディネータは各サブトランザクションに対するコマンドを一括して受けとり各サブトランザクションに渡している。ここで問題になるのがコマンドの追い越しである。例えば検索コマンドの実行後更新コマンドを実行する場合などである。このためにコマンドのシリアライズを行なう必要がある。これはコーディネータで行なえば簡単であるが、コーディネータが受けとった一つのコマンドの中で最も処理時間の長いサブコマンドの実行が終るまで待つのは効率が悪いので各サブトランザクションでおこなっている。

4.2 エラー処理

エラー監視はコーディネータと各サブトランザクションがそれぞれおこなっている。コーディネータは受け付けた全コマンドのステータスを監視する。各サブトランザクションは自身で受け付けたコマンドのステータスを監視する。

サブトランザクションのなかで一つでもエラーが発生すれば、それはコーディネータが発見する。コーディネータは全サブトランザクションにアボート通知を出し全ての処理を中断させた後全トランザクションのアボートを行なう。エラーが発生したトランザクションはそれ自身でアボートを行なう。

4.3 障害回復処理

1相目のコミット後コーディネータダウンもしくは通信エラーなどによりサブトランザクション自身コミットかアボートかが不明の場合、他の全サブトランザクションにどうしていいか尋ねる。その結果の一つでもコミットかアボートかの返事があればそれを行ないサブトランザクションを終了する。これによってコーディネータが立ち上がらなくてもコミットかアボートかが判断できる場合は、資源の解放(テーブルのアンロック)が可能となる。それ以外の場合はコーディネータの指示を待つ。

コミットかアボートかが不明のままシステムが落された場合は、システム立ち上げ時に障害が発生した時と同じ状態を再現し上記と同じ処理を行なう。立ち上げ処理と障害回復処理は並列に実行されるため、障害回復処理の終了を待たずに立ち上がることができる。

5 まとめ

疎結合並列マシン PIM/m のプロセッサ数64で水平分割テーブルを用いた評価をおこなった。この結果トランザクションへ発行された実行コマンドやコミット処理での制御コマンドは、全サブトランザクションにほとんど同時に到達し各サブトランザクションの処理が並行同時に開始されていることが確認された。これは分散データベースの手法が有効であったと言える。

今後は密結合並列マシン PIM/p での評価をおこない密結合並列での有効性を検証したい。

参考文献

[1] 河村哲か: 並列データベース管理システム Kappa-P の概要, 第45回情処全国大会, 5R-3, 1992-10.

[2] 著: "Parallel Inference Machine PIM" The International Conference on Fifth Generation Computer Systems '92, 1992.

[3] P.A.Bernstein, V.Hadzilacos, N.Goodman, "Concurrency Controland Recovery in Database Systems", Addison-wesley, 1987.

Kappa-P の中間言語処理部

澤部直太[1]、 西山聡[1]、 富樫泰子[1]、 河村元夫[2]

1: (株) 三菱総合研究所 情報科学部　　2: (財) 新世代コンピュータ技術開発機構

1 はじめに

第5世代コンピュータプロジェクトでは、大規模知識処理を目的として、並列推論マシン PIM[1] 及び その上で動作する並列知識処理システムの研究開発を行なっている。これらの知識処理システムでは、複雑でかつ大量の知識データを効率的に処理できるデータベース管理システム (DBMS) が必要とされる。Kappa-P[2] は、この目的のために研究開発された並列データベース管理システムであり、現在、その最初の版が PIM 上で動作している。Kappa-P 中間言語処理部は、関係代数演算処理を並列に実行する機能を提供するものである。中間言語処理部では、関係代数演算処理の対象とするデータ構造として、テーブル、集合、ストリームの3種類を用意しており、並列処理の処理効率を考慮してデータ型の選択が行なえるようになっている。本稿では、Kappa-P の中間言語処理部の設計方針ならびに提供機能について報告する。

2 Kappa-P の言語階層

Kappa-P はユーザからの問い合わせ処理を行なうために5段階の言語階層をもっており、中間言語はその4番目のレベルに相当する (図 1)。以下に、各レベルの特徴を説明する。

ユーザ記述言語： ユーザが問い合わせを記述するための言語で、拡張関係代数に基づく問い合わせ言語である。

関係代数列： ユーザ記述言語で書かれた問い合わせから抽出した関係代数列である。ユーザ記述言語の記述内容と性質一対一に対応する。

部分関係代数列： 関係代数列の内容に対して水平分割情報の内容を考慮して最適化を行なうことによって得られる、ローカル DBMS レベルで閉じた関係代数列である。

中間言語： 部分関係代数列に対してローカル DBMS 内での最適化を施した結果から生成される。中間言語を生成する上で行なわれる主な最適化は、検索条件を変形することによってデータベースに対する読み出しの回数を減少させるという静的なものである。

原始コマンド： レコード ID 情報によるポインタ操作に基づく演算であり、非正規関係に対する効率的な操作を提供している。

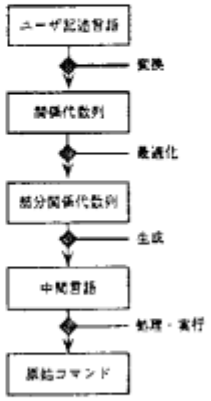

図 1: Kappa-P の言語階層

3 中間言語処理部における実行制御

Kappa-P の言語階層で特徴的なところは、ユーザは処理の並列性を考慮せずに問い合わせを記述するのに対して、部分関係代数列から中間言語を生成する際には、関係代数演算の流れの中である程度並列性を考慮した中間言語列の生成を行う必要がある点である。

中間言語処理部における基本的な実行制御方式は、処理対象レコードをパイプラインで処理プロセス (個々の関係演算処理、ならびにデータベースに対するアクセス処理) を流すことによって行なっている。しかしながら、いくつかの処理プロセスに関しては、パイプライン処理によって制御することができないため、これらのものについては、処理の正常終了を表すステータスを受け渡すことにより、実行制御を行なうこととした。ステータスの受け渡しによって実行制御を行なうものとして、以下のものがある。

関係の生成処理： 生成した新しい関係がデータベース上に正常に定義される前に、その関係に対してレコードの追加処理を行なってもその結果は保証されない。したがって、新たに作られた関係に対してレコードの追加処理は、生成処理の実行結果ステータスの到着を待って実行されなければならない。

関係の更新処理： 関係に対するレコードの追加、および削除といった更新処理を並列に実行することはデータベースの整合性を損なう原因となるため、逐次処理を行なう必要がある。したがって、ある関係に対して更新処理を行なう場合は、その関係に対する全ての更新前の処理に関する実行結果ステータスの到

Internal Language Processor for Relational Algebra on Kappa-P
Naota SAWABE[1], Satoshi NISHIYAMA[1], Yasuko TOGASHI[1], Moto KAWAMURA[2]
1:Mitsubishi Research Institute, Inc.　2:Institute for New Generation Computer Technology.

着を待って更新処理を実施しなければならない。また、更新した関係を対象とした処理についても、更新処理に関する実行結果ステータスの到着を待って実行されなければならない。

4 中間言語の構造

4.1 中間言語が提供するオブジェクト

中間言語処理部では、その処理対象として次の5種類のオブジェクトを用意している。

テーブル： 「関係」に相当するオブジェクトで、レコードの読み出し、ならびに書き込みを行なうことができる。

集合： テーブルから抽出したレコード群を、対応するレコードID情報列の圧縮形式で表現したものである。集合指定によるレコード群の読み出しや、集合同士による和・差・積演算処理を行なうこともできる。

整列集合： 集合に対して属性名をキーとして整列化を行なったものである。

ストリーム： レコード、またはレコードとそのレコードID情報の対のリストである。ただし、ストリームの個々の要素を複数個のレコード(たとえば、100レコード)を格納できるバッファとし、このバッファを要求駆動形式で受け渡すことにより、実行時のレコードの読み出しのオーバー・ヘッドを小さくするとともに、主記憶の効率的な利用を実現する。

ハッシュ表： テーブルに対して作られるハッシュ表(テーブル・ハッシュ)と、ストリームに対して作られる一時的なハッシュ表(ストリーム・ハッシュ)とがある。テーブル・ハッシュには、ハッシュ・キーとレコードID情報の対が蓄えられ、ストリーム・ハッシュにはキーとレコードの対を蓄えらることができる。

4.2 中間言語が提供するコマンド

中間言語処理部が提供するコマンドは大きく分けると、次の5つのグループに分けることができ、関係代数演算に関しては各オブジェクト(とその組合せ)に対応するものが用意されている。

関係代数演算： 選択処理、射影処理、和演算処理、積演算処理、差演算処理、結合演算処理、直積演算処理、整列処理

オブジェクト・タイプの変換： ハッシュ表の生成、ストリームからテーブルへの変換、集合からテーブルへの変換

関係の更新処理： レコードの追加、レコードの更新、レコードの削除

関係の構造操作： 中間言語レベルでのネスト／アンネスト処理用コマンド

外部プロセスとのインタフェース用コマンド： スキーマの変更コマンド、テーブル・プロセスの生成コマンド

その他： 関係に対するレコードの読み出しコマンド、リモート・アクセス・コマンド

4.3 中間言語の生成に関する基本方針

中間言語が提供する各オブジェクトの特徴から、部分関係代数列から実行効率の良い中間言語列を生成するための基本方針として以下のものを考えている。

- 関係演算のうち、和演算や直積演算は実行中に与えられた関係の全てのレコードを必要とするが、差演算、積演算、および結合演算はそうではない。特に、後者の演算については、オブジェクトの特性を活かすことにより、実行効率の良い中間言語列を生成することができる。
- ストリーム同士の演算の場合は、処理対象として与えられた全てのストリームを同一の属性で整列することにより、マージ処理として扱うことができる。
- ストリーム、集合、ならびにテーブルが混在している演算の場合は、以下の観点から、レコードをテーブルから読み出す回数を可能な限り少なくするようにする。
  - テーブル(集合と対になっているテーブルを含む)に索引属性がある場合は、ストリームを流れてくるレコード中の索引属性値を利用して、テーブルの索引読みを行なう。
  - テーブルにハッシュ表が作られている場合は、まずストリームを流れてくるレコード中の索引属性値を使ってハッシュ表からレコードID情報を取り出した上で、そのレコードID情報で指定されたレコードをテーブルから直接取り出す。
  - テーブルが索引属性とハッシュ属性のどちらも持たない場合は、そのテーブルにハッシュ表を作成し、上述の方法を適用する。

5 おわりに

現在、Kappa-Pの中間言語処理部は、個々のコマンドに関してPIM上での実装が終了している。今後は、複数LDBMSに渡る処理に対する評価、LDBMS間のデータの受け渡しに関する評価、ならびにこれらの評価結果に基づく中間言語コマンドの見直しを順次行なっていく予定である。

参考文献

[1] 瀧： "Parallel Inference Machine PIM" The International Conference on Fifth Generation Computer Systems '92, 1992.

[2] 河村哲か: 並列データベース管理システム Kappa-P の概要, 第45回情処全国大会, 5R-3, 1992-10.

Kappa-P のアンネスト / ネスト処理

川村達[1]，佐藤裕幸[2]，河村元夫[3]

1: 三菱電機東部コンピュータシステム (株)　　2: 三菱電機 (株) 情報電子研究所
3: (財) 新世代コンピュータ技術開発機構

1　はじめに

Kappa-P は並列推論マシン PIM で動作する DBMS であり、PSI 上の逐次版 DBMS の Kappa-II を発展させたものである。Kappa の最大の特徴の 1 つは、データモデルとして非正規関係モデル (レコードに階層と繰返しを許す) を採用していることである。非正規関係の意味論としてはいろいろ考えられるが、Kappa では、非正規関係の意味はそれをアンネストしたフラットなタプルの集まりと同じである [1,2]。よって Kappa では内部処理としてアンネスト / ネスト処理が必要となり、またそれが効率上の鍵を握る。

この論文では、Kappa でこの処理をどのように効率的に行っているかについて述べる。

2　アンネスト / ネスト処理

Kappa の原始コマンドの選択演算は、集合を用いて次の様に行われる。まず、選択条件の中の各単一条件を満たす集合間の集合演算の結果として、選択条件を満たす集合が得られる。次にその集合とテーブルから結果のレコードが得られる。この後者の処理の中でアンネスト / ネスト処理が必要となり、集合からいかに効率的に結果のレコードを得るかが問題となる。そこで、まず Kappa の集合について簡単に説明したのちに、集合から結果レコードを得るための処理方式について説明する。

2.1　集合

集合は選択演算を効率的に行うためのものであり、テーブル内の非正規レコードを特定するレコード ID(RID) とそのレコード内における各属性のオカレンスの情報の組を要素とする集まりである。オカレンスの情報は、SubRID(各属性の選択オカレンスを and 関係で並べたもので、肯定部と否定部から成る) と呼ばれるものの or 関係の並びから成っている。以下に定義を記述する。

```
Set      =  [Element, ···]
Element  =  {RID,[SubRID, ···]}   % SubRID and ···
         |  RID
SubRID   =  {Posi,Nega}           % Posi and not(Nega)
Posi     =  [Path, ···]           % Path or ···
Nega     =  [Path, ···]           % Path and ···
Path     =  {AID,OccInfo}         %
AID      =  Integer               % 属性 ID
OccInfo  =  [Occ, ···]            % オカレンス情報
Occ      =  Integer               % オカレンス
```

例として、次のような非正規テーブルを考えてみる (レコードは 1 件、* は繰返し属性を示す)。

社員名	家族 *		
	名前	趣味 *	資格 *
中村	道夫	ゴルフ	教員免許
		スキー	英検 1 級
	良夫	スキー	運転免許
		読書	

上記のテーブルで、例えば「趣味がスキーであるか、または教員免許の資格を持つ」条件を満たす集合は次の様に求まる。まず、「趣味がスキーである集合」と「資格が教員免許である集合」がインデックス検索によって直ちに得られ、次に集合演算で 2 つの集合の和がとられて以下の集合が得られる。

```
Set = [{1,[{[{<趣味>,[1,2]}],[]},
          {[{<趣味>,[2,1]}],[]},
          {[{<資格>,[1,1]}],[]}]}]
```

上記の集合は、レコード ID が 1 のレコードが選択され、かつそれが 3 つの SubRID を持つことを示している。そして各 SubRID は肯定部のみから成り、例えば 1 番目の SubRID は属性 "趣味" のオカレンス情報 [1,2]("家族" の 1 番目のオカレンスの中の "趣味" の 2 番目のオカレンス) のオカレンスが選択されていることを示している。

2.2　処理方式

集合から結果レコードを求める場合には、意味的には非正規レコードを一度フラットなタプルの集まり (下記参照) に変え (アンネスト) てから、条件に合うタプルのみを再び非正規レコードに戻す (ネスト) 操作が必要である。

	社員名	家族 *		
		名前	趣味 *	資格 *
O	中村	道夫	ゴルフ	教員免許
	中村	道夫	ゴルフ	英検 1 級
O	中村	道夫	スキー	教員免許
O	中村	道夫	スキー	英検 1 級
O	中村	良夫	スキー	運転免許
	中村	良夫	読書	運転免許

しかし、結果の非正規レコードを得るのに、実際にすべてのフラットなタプルを生成していたのでは効率が非常に悪いのは明らかである。そこで Kappa では基本的に実際の非正規レコードは用いずに、集合要素の中の属性のオカレンスの情報である SubRID を用いてアンネスト / ネスト処理を行っている。この場合、ネストの結

Unnest/nest of Kappa-P

Toru KAWAMURA[1], Hiroyuki SATO[2], Moto KAWAMURA[3]
1:Mitsubishi Electric Computer Systems(Tokyo).　2:Mitsubishi Electric Corporation.　3:Institute for New Generation Computer Technology.

果がそのまま非正規レコードになる訳ではなく、元レコードにおけるオカレンス選択情報が得られ、それを元レコードと対応させることで、結果の非正規レコードが得られる。以下に、SubRIDを用いたアンネスト/ネスト処理の流れを前述の例とともに示す。

(1) Nega部のPosi部への変換
集合の中の各SubRIDの否定部を肯定部に変換する(簡約処理)。例の場合には各SubRIDには肯定部しかないので、集合は変化しない。

(2) 属性組の構造の決定
各SubRIDに少なくとも一度は現れたすべての属性の並びをもって、属性組の構造とする。これは、フラットなタプルの属性の並びに相当するもので、この中に現れない属性はオカレンスがすべて選ばれているため、省略されている。例では、属性組の構造は{家族(趣味,資格)}である。

(3) 属性組リストの作成
各SubRIDを属性組に変換し、属性組のリストを得る。このとき、属性組の構造にあってSubRIDにない属性はオカレンスがすべて選ばれているので、各属性のオカレンスの組合せを取った1つ1つを属性組として属性組リストに加える。ここまでが、アンネスト処理に相当する。例では、属性組リストは以下の様に4つの属性組から成る。

家族*	
趣味*	資格*
[1,2]	[1,1]
[1,2]	[1,2]
[1,1]	[1,1]
[2,1]	[2,1]

(4) オカレンス選択情報の生成(ネスト処理)
上記の属性組のリストをネストシーケンスに従ってネストし、オカレンスの選択情報を得る。例えば、上記の属性組リストを[趣味,資格,家族]の順序でネストすると、次のオカレンス選択情報が得られる。

```
オカレンス選択情報 =
{{<家族>,[{1,{{<趣味>,[1,2]},{<資格>,[1]}}},
          {1,{{<趣味>,[2]},{<資格>,[2]}}},
          {2,{{<趣味>,[1]},{<資格>,[1]}}}]}}
```

上記は、属性"家族"にオカレンスが3つあり、それぞれ元レコードの1番目、1番目、2番目のオカレンスが選択されていることを示している。また更に"家族"の中の属性のオカレンスも制限されていて、例えば2番目のオカレンスの属性"趣味"では元レコードの2番目のオカレンスが選択されている。

(5) 結果のレコードの獲得
オカレンス選択情報を元の非正規レコードと対応させることで、結果としての非正規レコードが得られる。

社員名	家族*		
	名前	趣味*	資格*
中村	道夫	ゴルフ	教員免許
		スキー	
	道夫	スキー	英検1級
	良夫	スキー	運転免許

この方式は以下の利点により、ネストの速度向上、メモリ使用量の点で有利である。

- Integerの比較で済む(フラットなタプルだと比較対象が長い)。
- 属性組の数は、一般にすべてのフラットなタプルの数より少なくなる。
- 属性組を構成する属性数も一般にフラットなタプルにおける属性数より少なくなる。

3 Kappa-IIからの改良点

Kappa-Pのアンネスト/ネスト処理はKappa-IIを基にし、性能改善のために幾つか改良した。以下に、そのうちの主なものについて説明する。

3.1 属性組のグループ化

ネスト処理では、属性組の数が少なければ少ないほど効率が良い。そこで、属性組のオカレンスの表現に"すべて選択されている"という情報を用意した。これによって、SubRIDを属性組に変換する処理で、1つのSubRIDは1つの属性組になるだけなので、属性組の数は大幅に減る。

3.2 否定部の扱い

前述の属性組のグループ化と関連して、属性組のオカレンスの表現として、"…以外"を設けた。Kappa-IIでは否定部をあらかじめ肯定表現に変換(簡約処理)してから処理をおこなっているが、これを否定表現のままネスト処理をおこなうようにした。これによって、ネストすべき属性組の数が減る。

4 むすび

前述のKappa-Pでの改良によって、Kappa-IIでは時間がかかりすぎて動作しなかったものが、動作するようになったことが確認されている。一般に各属性のオカレンスの数が多くなればなる程、この改良がきくようである。今後、詳細な評価を行うとともに、アルゴリズムを再度見直して、更に高速化を図る予定である。

参考文献

[1] 河村ほか, 並列データベース管理システムKappa-Pの概要, 第45回情処全国大会,5R-3, 1992-10.

[2] 横田ほか, 総合知識ベース管理システム, 第五世代コンピュータの研究開発成果,1992.

Kappa-P の単一レコード・アクセス機能

中嶋かおり[1]，佐藤裕幸[1]，河村元夫[2]

1: 三菱電機 (株) 情報電子研究所　　2: (財) 新世代コンピュータ技術開発機構

1　はじめに

第5世代コンピュータプロジェクトでは、大規模知識処理を目的として、並列推論マシン PIM[1] 及び その上で動作する並列知識処理システムの研究開発を行なっている。これらの知識処理システムでは、複雑でかつ大量の知識データを効率的に処理できるデータベース管理システム (DBMS) が必要とされる。Kappa-P[2] は、この目的のために研究開発された並列データベース管理システムであり、現在、その最初の版が PIM 上で動作している。

Kappa-P のインタフェースとして、プログラムから直接使うものと、端末から使うものが考えられるが、現在、Kappa-P には端末から DBMS を操作する機能が実現されていない。そこで逐次データベース管理システム Kappa-II[3] で既に実現されている端末インタフェース機能を流用することを目的として、Kappa-P 上に、Kappa-II のコマンドを受け付ける機能を実現した。Kappa-P と Kappa-II の主な仕様の違いとして、DBMS に格納されているレコードのアクセス方法がある。Kappa-P では、並列性を出すため、一度にテーブル全体のレコードをアクセスすることを基本としているのに対し、Kappa-II では、1 レコードずつアクセスする (= 単一レコードアクセス) 仕様となっている。また、端末にレコードの値を表示するなどを考えると、全てのレコードを一度に表示することは不可能であり、常に全てのレコードの値が必要ということはない。したがって、Kappa-II の端末インタフェースを流用しないとしても、単一レコードアクセス機能は、端末にレコード値を表示する場合には必要である。

本論文では、この Kappa-P における単一レコードアクセス機能を中心に報告する。

2　機能概要 及び 構成

Kappa-II のコマンドを受け付ける機能を実現するため、大きく分けて Kapppa-II エミュレータ部と、タップ付きコマンドインタフェース部の二つの部分から構成されている (図 1)。

以下に、Kapppa-II エミュレータ部と、タップ付きコマンドインタフェース部の各々について説明する。

Kapppa-II エミュレータ部: Kappa-P と仕様が異なる Kapppa-II コマンドを、Kappa-P の仕様に合わせるための変換を行なう。但し、単一レコードアクセス機能など、一部の機能を実現するための変換は、タップ付きコマンドインタフェース部で行なっている。ここでいう仕様の違いには、レコード読み方法の指定形式、レコード読み順の指定形式、スキーマ (テーブルの定義) の形式、データの検索条件の形式、レコードの形式などがある。

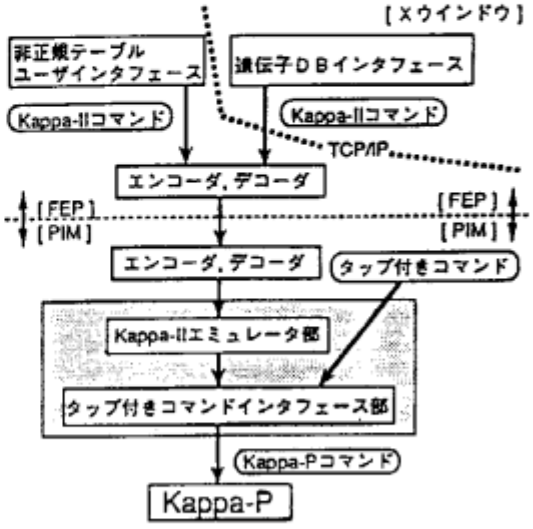

図 1: 構成

タップ付きコマンドインタフェース部: 主に、単一レコードアクセス機能 (1 レコードずつ読み込む機能) を提供するための、コマンド変換を行なっている。タップとは、レコードの位置情報を指すポインタ・オブジェクトである。その他の提供機能として、集合 (選択されたレコード群を指すもの) とその識別子の管理表による、集合の管理機能があるが、詳細は省略する。

端末インタフェース (図 1の非正規テーブルユーザインタフェース) から入力された Kappa-II コマンドは、FEP と PIM 間の通信のため FEP 及び PIM のエンコーダ, デコーダ を通る。次に Kapppa-II エミュレータ部、タップ付きコマンドインタフェース部でそれぞれ変換され、Kapppa-P コマンドとして Kappa-P 本体に送られる。

Sequential record access for terminal interface on Kappa-P

Kaori NAKAJIMA[1], Hiroyuki SATO[1], Moto KAWAMURA[2]

1:Mitsubishi Electric Corporation　2:Institute for New Generation Computer Technology.

Kapppa-II エミュレータ部 とタップ付きコマンドインタフェース部の二つの部分に分けたことにより、タップ付きコマンドインタフェース部に、直接タップ付きコマンドを送って、Kappa-P を使うことができる。また、端末インタフェースとして、FEP 上のインタフェース以外に、TCP/IP を介して、X ウインドウ上の端末インタフェースからの操作も可能である。

3 Kappa-P のレコード読み込み機能

単一レコードアクセス機能を説明する前に、まず、Kappa-P 本体のレコード読み込み機能について述べる。Kappa-P のレコード読み込みは、原則としてテーブル全体のレコードを一度に読み込む。これは、1 レコードずつの読み込みが処理単位として小さ過ぎ、並列性がないためである。ただし、二次記憶上に存在するデータを一度に全て主記憶に読み込んでしまうと主記憶の容量がたりなくなってしまう。そこで、Kappa-P のレコード読み込みは、要求駆動の形態をとっている。

この要求駆動と言うのは、レコード値を入れるバッファを Kappa-P に渡すと、ある決まった個数 (例えば、100 個) のレコードが読み込まれるものである。ユーザは、このバッファに入れられたレコードを処理し終ったら、次のバッファを Kappa-P に渡して、次のレコード群を読み込む。従って、二次記憶から主記憶に一度に読み込まれるのは、この決まった数だけであり、主記憶があふれることはない。また、一度に複数のバッファを渡すことにより、ダブル・バッファリングが行なえ、効率良くレコードの読み込みが行なえる。

4 単一レコード・アクセス機能

4.1 タップ

単一レコード単位でアクセスするためには、タップというものを用いる。タップとは、レコードや集合をアクセスする時に必要な情報を保持するものである。タップを生成する際に、ユーザはアクセスしたいレコードが存在するテーブル名と、アクセスの順番、アクセス方法を指定する。アクセスの順番としては、レコードの格納順、昇順、降順などが指定できる。アクセス方法としては、アクセスする対象を、一つのある属性とするか、あるいは複数の属性とするか、などが指定できる。タップは、これらの情報を保持しておく。

4.2 実現方式

次に、上記のタップを用いた、単一レコードアクセスの実現方法について述べる。

- 最初にユーザからレコード読みが要求されたら、タップが保持しているテーブル名、アクセスの順番、アクセス方法などの情報を用いて、Kappa-P に対して 1 バッファ分のレコード読みを要求し、読み込まれたバッファの先頭の 1 レコードを取りだしてユーザに結果として返す。

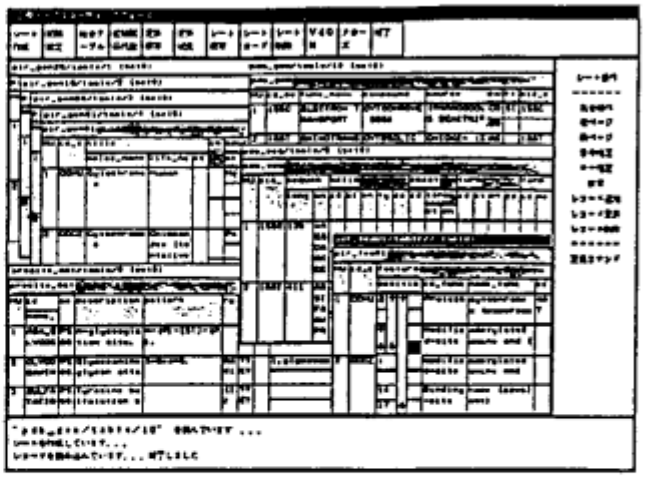

図 2: 非正規テーブルユーザインタフェース

- 次にユーザからレコード読みが要求されたら、既に読み込まれているバッファの、先頭の 1 レコードを取り出して返す。
- 以下、レコード読みが要求される毎に、バッファから 1 レコードずつ取り出していく。
- バッファが空になったら、再び Kappa-P にレコード読みを要求して、以下同様の操作を繰り返す。

5 おわりに

以上の機能により、非正規テーブル・ユーザインタフェース [4] から、Kappa-P を使うことが可能になった。非正規テーブル・ユーザインタフェースとは、Kappa-II で実現されている端末インタフェースであり、非正規関係に対する操作を提供している (図 2)。

また、単一レコード・アクセス機能を提供することにより、FEP (端末インタフェース) に必要な情報だけを送るため、FEP と PIM との間の通信量が少なくなり、通信コストが削減されている。

今後の課題としては、

- 現状は主要なコマンドのみなので、非正規テーブル・ユーザインタフェースの全てのコマンドをサポートする。
- 現在、エンコーダ、デコーダがネックになっているので、その部分の高速化を行なう。

の二点があり、これらについて改良、評価していく予定である。

参考文献

[1] 齋： "Parallel Inference Machine PIM" The International Conference on Fifth Generation Computer Systems '92, 1992.

[2] 河村ほか: 並列データベース管理システム Kappa-P の概要, 第 45 回情処全国大会, 5R-3, 1992-10.

[3] 横田ほか: "Overview of the Knowledge Base Management System(KAPPA)", The International Conference on Fifth Generation Computer Systems '88, 1988.

[4] 小池ほか: 知識ベース管理システム Kappa の非正規関係ユーザインタフェース, 第 37 回情処全国大会, 2Q-7, 1988-10.