TM-1246

並列データベース管理システム Kappa-P

椿野　宣行、川村　達、佐藤　裕幸（三菱）

March, 1993

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

並列データベース管理システムKappa−P

基本ソフトウェア開発第一部　・椿野宣行　・川村　進
三菱電機（株）情報電子研究所　・佐藤裕幸

1. はじめに

通産省第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開発が行なわれている。これらの知識処理システムでは，複雑かつ大量なデータを扱うため，大量の知識データを効率よく処理できる知識ベース管理システムが必要とされている。Kappa−P（Knowledge APPlication oriented Advanced database management system Parallel version）[2] は，そのために開発された並列データベース管理システムであり，現在初版が動作している。本稿では，Kappa−Pの全体概要及び当社が開発に携わった非正規モデルにおけるレコード処理及びトランザクション制御などについて述べる。

2. Kappa−Pの概要

2.1 動作環境

Kappa−P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上で動作する並列データベース管理システムである。PIMはMIMD型の疎結合と密結合が混合されたハイブリット並列マシンである。8台程度の要素プロセッサ（PE）が，共有バス／共有メモリにより結合されて一つのクラスタになり，クラスタ間は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。

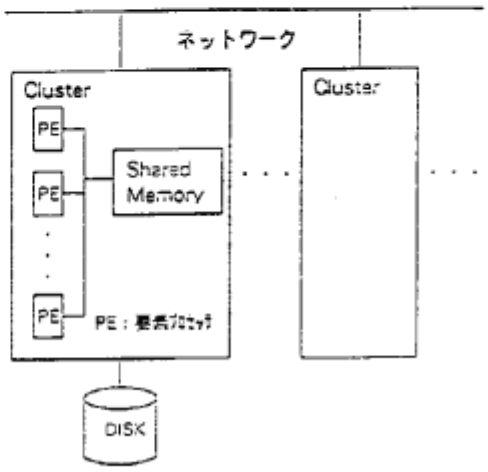

図1. PIMの構成

2.2 全体構成

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をとる。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。

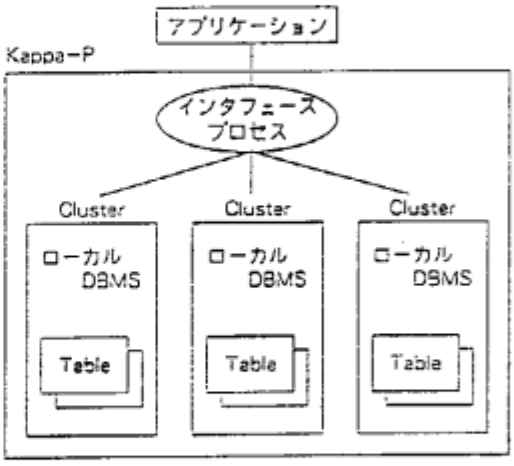

図2. Kappa−P全体構成

2.3 非正規関係モデル

Kappa−Pは，知識処理で扱う複雑な構造データを効率よく処理するため，関係モデルを拡張した非正規関係モデルを採用している。このモデルは，レコードに階層と繰返しを許すものである。これにより関係モデルで扱うことが困難であった複雑な構造をしたデータを扱うことができる。またデータの冗長部分を省略できるため記憶容量の節約にもなる。

従来の関係モデル

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

Kappa−Pでの非正規関係モデル

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

図3. 関係モデルと非正規関係モデル

2.4 主記憶テーブル

PIMのクラスタは，数百メガバイトの大容量主記憶をもっている。Kappa−Pは，この大容量主記憶を利用した主記憶テーブルを提供している。主記憶テーブルは二次記憶に反映されない一時テーブルとして実装している。

2.5 並列処理

Kappa−Pは，分散DBMSの構成となっており各クラスタにローカルDBMSを対応させているため水平分割テーブルなどローカルDBMS間の並列処理ができる。また各ローカルDBMSは一つのクラスタに配置されているためローカルDBMS内の内部処理で密結合向き並列処理ができる。

3. 非正規モデルにおけるレコード処理

3.1 Kappa−Pの非正規関係

Kappa−P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−P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としている [3]。そのため，選択演算で正しい値を返すためにはレコードを一度フラットなタプルにしてから，条件に合うタプルのみを再びネストするという操作が必要である。

次の非正規テーブル（レコードは1件，*は繰返し属性を示す）を用いて，選択演算で検索結果のレコードが返される例を示す。

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

図4. 非正規テーブルの例

検索条件：家族の趣味がスキーであるか，または家族の資格が教員免許である。

まず，上記のレコードは検索条件に合う部分を含んでいるので，このレコードをアンネストして以下のフラットなタプルの集まりにする。

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

図5. フラットなタプルの集まり

上記のフラットなタプルのうち，条件に合っているもの（○が付いたもの）のみをネストすると次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 晃	道夫	ゴルフ	教員免許
		スキー	
	道夫	スキー	英検1級
	良子	スキー	運転免許

図5. 選択演算結果のレコード

上記のレコードが，選択演算の結果として返されるレコードである。このレコードの形はネストする属性の順番（ネストシーケンス）に依存している。上記の例では［趣味，資格，家族］の順にネストされて一つのレコードになっているが，一般的には複数の非正規レコードになる。

3.2 Kappa-Pの選択演算

Kappa-Pの選択演算は集合を用いて行われる。集合は選択演算を効率良く行うために導入されたものである。そして選択演算の結果のレコードは集合の各集合要素をもとに作り出す必要がある。集合の例を以下に示す。これは，前述のテーブルと検索条件から得られる集合である。

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

集合要素が表しているのは，該当レコードとその中の各属性の該当値（オカレンス）の選択情報である。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（SubRID内はand関係。SubRID間はor関係。）を持つことを示している。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番目のオカレンスであるスキー）のオカレンスが選択されていることを示している。

集合要素から対応するレコードを得るためには，前述した様にKappa-Pの非正規関係の意味に従って，レコードをフラットなタプルの集まりにアンネストし，各属性の該当値（オカレンス）の選択情報に合うタプルのみを再びネストすれば良いのだが，Kappa-Pでは次に述べる様にもっと効率の良い方式を採用している。

3.3 Kappa-Pのアンネスト／ネスト処理

kappa-Pは以下の方法で集合要素からレコードを取り出している。

- 集合要素の示す各属性の該当値（オカレンス）の選択情報そのものをタプルとみなし，それらをネストして，元レコードのオカレンスの選択情報を生成する。これは，ネストした結果のレコードが元レコードのどのオカレンスが選択されたものかを示すものである。
- 元レコードにオカレンスの選択情報を対応させることによって，結果のレコードが求まる。

次にこれを前述のテーブルと検索条件を用いた例で示す。

(1)タプルの構造の決定

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，タプルの構造とする。例では，タプルの構造は｛家族（趣味，資格）｝である。

(2)タプルの作成

各SubRIDをタプルに変換し，タプルのリストを得る。このとき，タプルの構造にあって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つをタプルとしてタプルリストに加える。ここまでが，アンネスト処理に相当する。例では，タプルのリストは以下の様に4つのタプルから成る。

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

図7. タプルリスト

(3)オカレンス選択情報の生成（ネスト処理）

上記のタプル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，上記のタプルリストを［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]},    %オカレンス1
                {<資格>,[1]}]},
```

```
{1,[{<趣味>,[2]},      %オカレンス2
    {<資格>,[2]}]},
{2,[{<趣味>,[1]},      %オカレンス3
    {<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。

(4)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 晃	道夫	ゴルフ	教員免許
		スキー	
	道夫	スキー	英検1級
	良子	スキー	運転免許

図8. 選択演算の結果のレコード

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）。
- タプルの数は，一般にすべてのフラットなタプルの数より少なくなる。
- タプル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。

4. トランザクション制御

トランザクションとは一般に複数のI/O（read，write）を含む処理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並列性を考慮した同時実行制御とコミットメント制御をおこなっている。

4.1 トランザクションの並列性

トランザクションは分散トランザクションとして実現している。すなわち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ローカルDBMS間の並列処理を実現している。またローカルDBMSは一つのクラスタに割り当てられるのでローカルDBMS内の個々のサブトランザクションも並列に処理される。

4.2 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行うこととした。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。

4.3 分散ロック方式

分散ロック方式は，テーブルの施錠管理を各テーブル自身がキューを持ちテーブル自身が管理をおこなうものである。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困難となってくる。このデッドロックの問題をKappa-Pでは，時間監視とリトライを行なうことで対処している。

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。

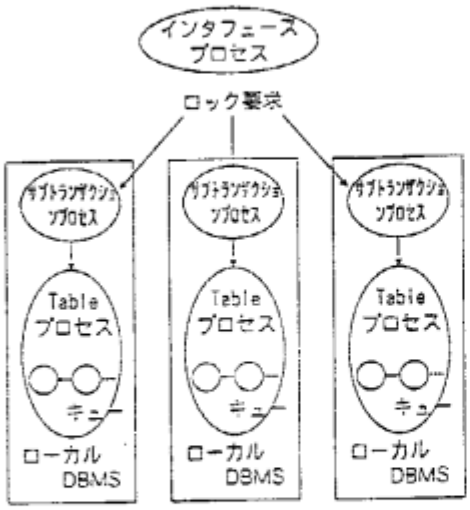

図9. 分散ロック

(1)ロック方法

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する。この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ランダムな時間待ったあとでリトライをおこなう。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。

(2)複数テーブルに対する考慮

リトライを行う場合は，デットロック監視時間を増加させることにより多くのテーブルをロックするトランザクションを不利にさせないようにしている。

(3)水平分割テーブルに対する考慮

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トロックになる可能性が高いので，最初は代表テーブル（スキーマで決定）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。

4.4 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 [4] やブロック状態を減らすための3相コミット [4] 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）である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。

またKappa-Pでは，大容量主記憶を利用した主記憶テーブルを提供しており主記憶テーブルは二次記憶に反映されない一時テーブルとして実装しているため主記憶テーブルと二次記憶テーブルに関しては，2種類のコミットメント制御が必要となる。Kappa-Pのコミットメント制御は以下のように行なっている。

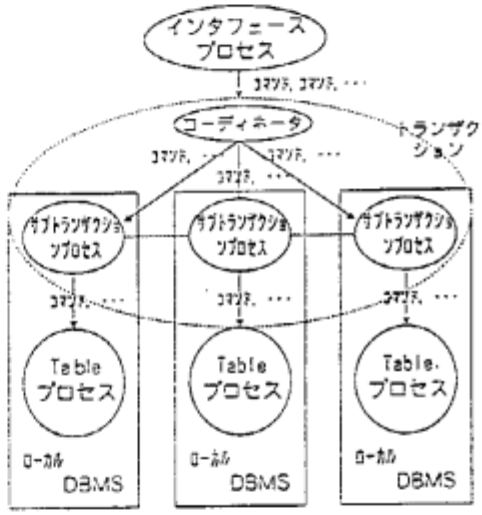

図10. トランザクション構成

(1)コミット処理

全体処理：

コーディネータが2相コミットに従って複数のLDBMSのトランザクションをコミットする。

LDBMSでの処理：

二次記憶テーブルの更新は主記憶上のバッファのみを更新し，コミット処理でログ経由で二次記憶に反映する。主記憶テーブルの更新は主記憶を直接更新しコマンドログを採取し，コミット処理では更新時に採取したコマンドログの消去をおこなう。

(2)アボート処理

全体処理：

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。エラーを発見した

場合は全サブトランザクション処理を中断後アボートする。

LDBMSでの処理：

二次記憶テーブルは，主記憶上の更新情報（バッファ）を消去する。主記憶テーブルは，逆コマンドの発行をおこなう。

(3)障害回復処理

コーディネータがダウン又は通信エラーの場合：

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処理を続行する。この場合は二次記憶テーブル，主記憶テーブルとも同一処理を行う。

立ち上げ時の回復処理：

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などは障害発生時と同一環境を再現することにより障害回復を行う。この場合は二次記憶テーブルのみ回復する。

5. 終りに

Kappa−Pは現在初版が動作しており水平分割テーブル（16分割）での簡単な動作確認を行なった。この結果では並列効果も得られ十分な性能で動作することが確認された。今後本格的な測定・評価を行ないシステムの安定化及び改良を行なっていく予定である。

6. 参考文献

[1] 溜："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：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

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

執筆者紹介

椿野　宣行

1983年4月入社
島根大学理学部卒業
(社)情報処理学会会員
逐次型推論マシンPSI上のデータベース管理システムKappa−I,Kappa−II開発を経て,現在は並列推論マシンPIM上での並列データベース管理システムKappa−Pの開発を担当。

川村　達

1984年4月入社
弘前大学理学部卒業
(社)情報処理学会会員
逐次型推論マシンPSI上のデータベース管理システムKappa−I,Kappa−II開発を経て,現在は並列推論マシンPIM上での並列データベース管理システムKappa−Pの開発を担当。

佐藤　裕幸

1982年4月三菱電機（株）入社
筑波大学第三学群情報学類卒業
入社以来,論理Prologの開発,第五世代コンピュータプロジェクトにおいて逐次型推論マシンのOS(SIMPOS),並列推論マシンのOS(PIMOS),並列データベース管理システム(Kappa−P)の開発に従事。昭和60年6月より4年間新世代コンピュータ技術開発機構（ICOT）に出向。