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]

5R−8

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＊
○	中村	道夫	ゴルフ	教員免許
	中村	道夫	ゴルフ	英検1級
○	中村	道夫	スキー	教員免許
○	中村	道夫	スキー	英検1級
○	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

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.

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.