1N-06

属性の動的追加を考慮したタプルデータセットの実装方式

千葉陽介† 北嶋翔吾‡ 都司達夫† 樋口 健†
福井大学工学研究科† アートテクノロジー㈱‡

1．はじめに

近年，業務の拡大や多様化に応じて，システム運用時に新たに必要な属性が追加されるなどのデータベース定義の動的な変更(スキーマ進化)を必要とする機会が増えている．しかし，属性追加後のデータを効率よく扱うためには，一般にデータベースの再構築を必要とし，高いコストが伴う．本研究室では拡張可能配列の概念を用いて，経歴・パターン法という多次元データセットのエンコード方式およびその実装方式 HPMD (History Pattern implementation for Multidimensional Datasets)[1]を提案している．本研究では， 新たな属性の動的追加を考慮した多次元データを HPMD を用いて，効率的に扱えるような方法を提案し，その実装と評価を行う．

2. 経歴パターン法の概要

経歴・パターン法は拡張可能な論理配列空間の座標値をエンコードする方法であり，任意の次元の座標を経歴値とパターンの組で表現する(図 1)．論理配列の拡張では，拡張直前の配列と同じ大きさの部分配列が拡張次元方向に付加される．拡張した順序を部分配列の経歴値と呼ぶ．境界ベクトルは座標の各次元添字の表現に必要なビット数である．座標はそれが属する部分配列の経歴値 h と境界ベクトルを用いて，各次元添字をビット列として結合したビットパターン p の対<h，p>としてエンコードされる．p のビット長は h の値と等しくなるという性質がある．添字のビットサイズは固定長ではなく，配列拡張に応じて漸増する可変長であり，コンパクトに実装でき，また，値の照合はシフトとマスク命令のみで行われ，高速な検索を保証し得る[1].

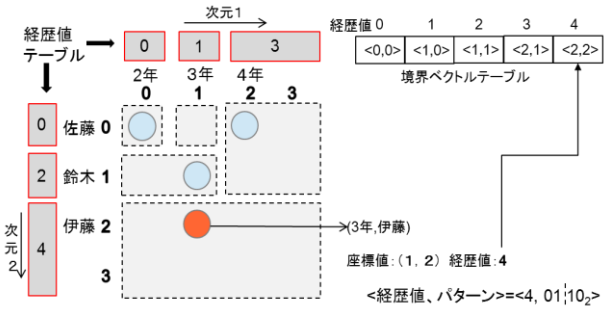

図 1. 経歴パターン法によるエンコード例

"An Implementation Scheme for Tuple Datasets Incorporating Dynamic Addition of Attributes",
†Chiba Yosuke, Tatsuo Tsuji, Ken Higuchi: Univ. of Fukui
‡Kitajima Shogo: ART Technology Co. Ltd.

3. 属性の動的追加の方法

経歴パターン法において新たな属性の追加は，拡張可能配列の次元を動的に 1 つ増加させることで対応できる．n 次元拡張可能配列では，新たな属性の追加要求後，実際に n+1 次元に新たな属性値が登録された際に次元拡張が起こる(図 2). このとき，次元拡張前の n 次元タプルの拡張次元 n+1 次元の属性値は NULL 値に対応させ，この NULL 値は n+1 次元添字 0 に対応させる．これにより，従来のエンコード/デコードのアルゴリズムは，ほぼそのまま利用できる

学年	名前	趣味
2	佐藤	
3	鈴木	
4	伊藤	
4	高橋	麻雀

座標に変換

次元1	次元2	次元3
0	0	0
1	1	0
2	2	0
2	3	1

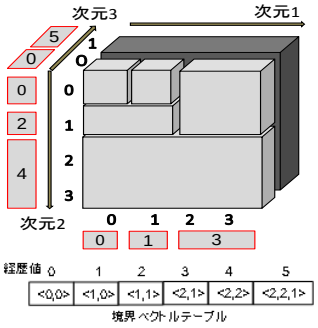

図 2. 図 1 の 3 次元方向への拡張

4. 属性の動的追加における問題点とその対策

3 節の方法を用いて，属性の動的追加を扱うとき，追加前のタプルが挿入された場合には，不利な状況が発生する(図 3 参照)．例えば経歴値 3 の時点で 2 次元目に新たな属性値を持つ 2 次元タプルが挿入され，2 次元方向の拡張がなされたとする．このときでも経歴パターン法にしたがって，3 次元配列全体が 2 次元方向への拡張を起こし，2 次元タプルであるにもかかわらず，経歴値 4 を使用するため，3 次元の座標添字のサイズをパターン中に占有してしまう．

経歴値	0	1	2	3	4
次元数	2	2	2	3	3
境界ベクトル	(0, 0)	(1,0)	(1,1)	(1, 1, 1)	(1,2,1)

図 3. 属性の動的追加前のタプル挿入が 2 次元方向の拡張を引き起こした場合の境界ベクトルテーブル

この問題に対して，タプルの次元数をタプル自身に含ませる，すなわち， k 次元タプル t (v_1, v_2, …, v_k)において， k をタプルの 1 次元目に挿入し k+1 次元のタプル t'(k, v_1, v_2, …, v_k)を構成する．その後，t'を境界ベクトルを用いて k+1 次元分エンコードし， パターン p を求め<経歴値 h, p>の組に変換する．このとき，k+1 次元を超える次元については p に含ませる必要がなく，したがって，p のビット数は経歴値 h 以下である．

図 3 に示す境界ベクトルテーブルでは次元数の情報が付加されている．

5. 属性の部分集合から構成されるタプルの挿入

4 節の方法では，最大次元数 n に対して，1≦k≦n なるすべての k について，属性値が定義されているとした．ここでは，属性の任意の部分集合について属性値が定義されている場合のタプルのエンコード法を示す．タプルにおいて，定義されている属性を表すためにビット列 B を用いる．ビット列 B のビット i (1≦i＜n)が 1 のとき n-i 次元目の属性値がタプルにおいて定義され，0 のときは未定義属性値である．ビット列 B を未定義パターンと呼ぶ．図 4 のタプルの場合，未定義パターンは 1101 となる．

学年	名前	e-mail 1	email 2
4	田中		abc@xyz,ac.jp

図 4. 未定義属性値を含むタプル

タプル t の 1 次元目に次元パターンを挿入した n+1 次元タプル t'をエンコードする．境界ベクトルを用いて t'内の未定義パターンを求め，未定義でない次元のみをエンコードし，<h,p> の対を得る．p には未定義属性値分のサイズは含まないので，p のビット長は実際には経歴値 h 以下である．図 5 にこの場合のエンコード法を示す．

未定義パターンは他の属性値と同様にエンコードされるので，その添字の最大は未定義パターンの種類の数 m であり，新たな未定義パターンの出現順に 1~m の値が割当てられるが p に占める未定義パターンの添字サイズは可変である．

未定義パターン	学年	名前	e-mail1	e-mail2
11	2	佐藤		
111	3	鈴木	suzu@fukui.ac.jp	
1101	4	佐藤		abc@xyz.ac.jp

(**1101**, 4, 佐藤 , NULL, abc@xyz.ac.jp)

→ 経歴値7,座標値 (**2**, 2, 0, -, 1)

↓経歴パターン法でエンコード

<経歴値,パターン> = <7, 10 10 0 1>

経歴値 0 … 7

…… <2,2,1,1,1>

境界ベクトル

図 5. 属性の部分集合からなるタプルのエンコード法

6. 評価

属性の動的追加に関して，DB 構築時間，構築サイズ，検索時間の測定を行った．測定データは 2000 万件の 7 属性数のテーブルである．比較対象として PostgreSQL を使用した．結果を図 6,7,8 に示す．測定では属性の動的追加を行いながらタプルを挿入した場合と動的追加を行わずに，最初から最大属性数に定義されたテーブルにタプルを挿入する場合の 2 種類の方法によった．図中の仕様 1 は 3 節をもとにした実装，仕様 2 は 4 節の次元数をもとにした実装，仕様 3 は 5 節の未定義パターンをもとにした実装を指す．

図 6 より，経歴パターン法を使用した場合は属性の動的追加ありの結果が無しの場合よりも若干早い．このことから，経歴パターン法では挿入するタプルの次元数によって生じる構築時間の差が属性の動的追加によるコストを十分保証していると考えられる．PostgreSQL では，動的追加有りがやや時間がかかっている． 図 7 より，いずれの実装でも属性の動的追加を行った場合でも構築サイズにほぼ変化はない．

検索時間に関しては仕様 1 に比べ仕様 2， 3 はデコードがやや複雑になったことから若干，遅くなっている．しかし，比較対象よりも 3~4.5 倍程度速い．これはおもに，経歴パターン法の高速なデコードと構築サイズが小さいことによる。PostgreSQL では動的追加有りの場合がやや遅くなっている．

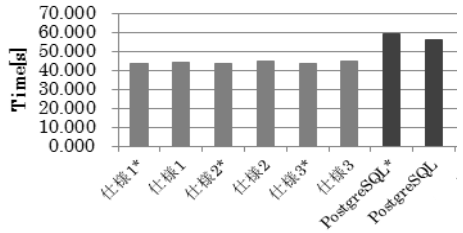

図 6. 構築時間(*は属性の動的追加ありを示す)

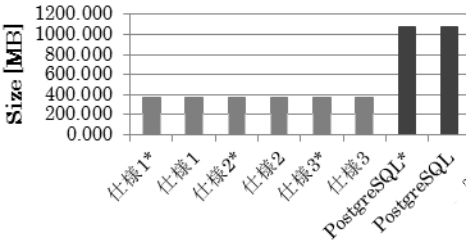

図 7. 構築サイズ(*は属性の動的追加ありを示す)

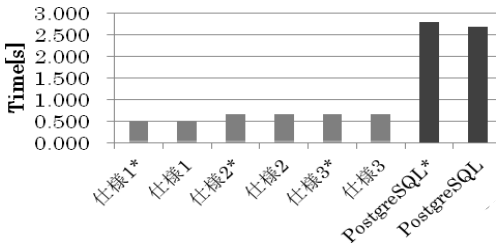

図 8. 検索時間(*は属性の動的追加ありを示す)

これらの測定結果から経歴パターン法を用いた属性の動的追加の実装方式は PostgreSQL と比較しても構築時間，検索時間の劣化が抑制されていることがわかる．

7. むすび

未定義属性値を含む場合の評価をその種々の分布について行う予定である．

参考文献

[1]前田，水野，都司，樋口，“多次元データのコンパクトな実装とその性能評価”，第 3 回データ工学と情報マネジメントに関するフォーラム，D6-2，2011