D-021

タプルデータセットの一エンコード方式

An Encoding Scheme for Tuple Datasets

菅安 唯伽† 都司 達夫† 樋口 健†

Yuika Sugayasu Tatsuo Tsuji Ken Higuchi

1. はじめに

複数の属性値からなるタプルデータのエンコード方式やその実装方式について，現在まで，多くの研究が行われている．我々が提案している，経歴・パターン法[1]と呼ぶ方式は，動的に増大し得るタプルデータセットに対して，優れた記憶コストと検索コストを提供することができる方式である．しかし，エンコード結果は，必ずしも最小記憶コストを保証するとは言えない．本稿では，エンコード結果の記憶コストの視点から，経歴・パターン法と競合し得る新しいエンコード方式を提案し，検索時間をはじめ他の性能について，経歴・パターン法と定性評価する．

2. 経歴・パターン法

経歴・パターン法は拡張可能な論理配列空間における n 次元座標のエンコード方式であり，任意の座標を経歴値とパターンの組で表現できる．論理配列空間は必要に応じて新たな論理部分配列を付加することにより拡張可能であるために，各次元の属性のカーディナリティがシステム運用時に増加する動的な多次元データに対応できる(図1)．経歴値は部分配列の付加順序を表し，パターンは論理配列の各次元の添字のビットパターンの連結である．また，パターンにおける添字の境界は境界ベクトルとして記憶され，各次元の添字を何ビットで表現するかを表す．経歴値，部分配列，境界ベクトルはそれぞれ，1 対 1 対応しており，経歴値を添字とする境界ベクトルテーブルが存在する．論理配列は，拡張の際に拡張前の空間と同じ大きさの拡張が行われることで，拡張後の論理配列空間は拡張前の 2 倍となる．これにより，経歴値は座標パターンのビットサイズと等しくなり，エンコード結果の<経歴値，座標パターン>対を可変長データとして逐次，出力ファイルに格納できる．また，各配列添字から座標パターンへの変換はレジスタ命令であるビットシフト，ビットマスクにより高速にエンコードが行える．

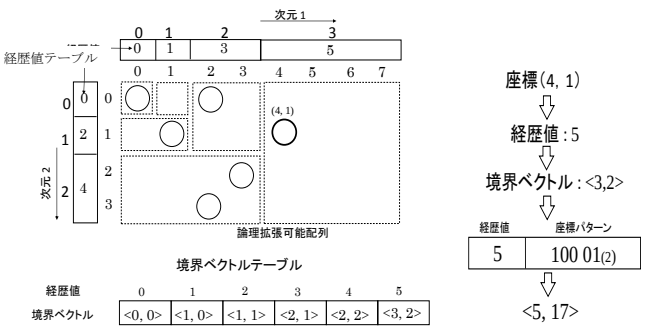


図 1 経歴・パターン法によるタプルエンコード

†福井大学 工学研究科, Graduate School of Eng., Univ. of Fukui.

任意のデータ型を有する属性値のタプルに対して，経歴・パターン法を適用するためには，属性値を添字に変換するためのデータ構造(例えば B+木)および，添字から対応する属性値への変換データ構造(1 次元配列)を次元毎に必要とし，この 2 種類の変換データ構造を加えた，多次元データセットの実現方式を HPMD(History-Pattern Implementation for Multidimensional Datasets)と呼ぶ．

3. 区画エンコード法

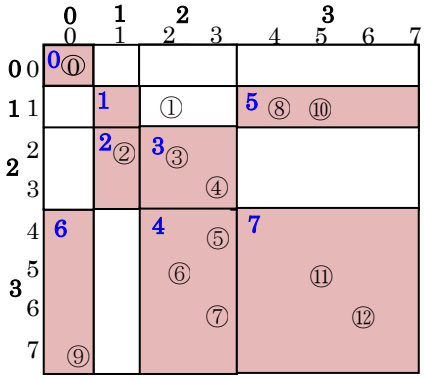
n 個の属性値よりなる n 次元タプル $t = (v_1, v_2, \ldots, v_n)$ の各属性値をその出現順に論理配列空間の添字に対応づけて変換した n 次元座標を $(i_1, i_2, \ldots, i_n)$ とし，各添字 i_k $(1 \leqq k \leqq n)$ のビットパターンサイズのベクトル $<b_1, b_2, \ldots, b_n>$ を t の境界ベクトルとよぶ．例えば，n 次元座標が(8, 6, 20, … ,15)のとき，その境界ベクトルは<4,3,5, …, 4> となる．この境界ベクトルは同時に，タプル t を含む区画を表す．I を 0 以上の整数集合とするとき n 次元座標空間

$$C_n = \{(i_1, i_2, \ldots, i_n) \mid i_k \in I, k=1,2,\ldots,n\}$$

に対して，C_n 上の関係≡を任意の $p, q \in Cn$ に対して，

“$p \equiv q \Leftrightarrow p$ と q のビットパターンサイズベクトルが等しい”

と定義すれば，≡は同値関係であり，C_n は上記の区画の集合に類別できる．図2は C_2 の区画集合の一部を示す．



<0,0	<1,1	<1,2	<2,2	<2,3	<3,1	<0,3	<3,3
1	1	1	2	3	2	1	2

図 2 区画エンコード法による 2 次元空間の区画成長

区画エンコード法では，タプルのエンコードは<区画ID, パターン>の対で表される．境界ベクトルテーブルは保有するが，経歴値の考え方はなく，経歴値テーブルは保有しない．すなわち，このエンコード方式は経歴・パターン法におけるような拡張可能配列に基づく方式ではない．

区画に座標が存在する場合，その区画を活性区画という．区画IDは区画の活性化順に0,1,2,・・・の整数値が割り当てられる．座標の挿入により区画がはじめて活性化した場合，境界ベクトルテーブルにその区画IDを添字として，新たなエントリーが割り当てられる．このエントリ

ーには当該区画の境界ベクトルと区画に存在する座標の数が記録される．図2は，以下のような2次元座標が順次，挿入された場合，区画の成長の様子を示している．以下の上段は座標の挿入順序，中段は挿入座標，下段は，座標の挿入にしたがって活性化される区画の区画IDを表す．

⓪ ① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫

(0,0) (1,1) (1,2) (2,2) (3,3) (3,4) (2,5) (3,6) (4,1) (0,7) (5,1) (5,5) (6,6)

0 1 2 3 4 5 6 7

境界ベクトルテーブル以外に，経歴パターン法と同様の属性値と添字の相互変換のための 2 種類の変換データ構造を加えた多次元データセットの区画エンコード法による実現方式を SPMD(Section-Pattern Implementation for Multidimensional Datasets)と呼ぶ．

4. 経歴・パターン法におけるパターン冗長性

例えば，図 1 の例において，座標(4,1)は<5,100.01$_{(2)}$>とエンコードされ 2 次元の添字 1 は本来 1 ビットで済むところを 01$_{(2)}$と 2 ビットを要している．これは座標(4,1)を表現するのに必要以上に大きい部分配列においてエンコードされるためである．経歴・パターン法では，このようなパターンの冗長性を許容することにより部分配列を指定するための経歴値のビットサイズを低く抑えているといえる．下記の図 3 は図 2 と同じ順序で各座標が挿入された場合に経歴パターン法による 2 次元配列の成長を示している．太線で示される部分配列が配列拡張時に割り当てられる．図 3 に見るように区画エンコードにおける各区画は経歴パターン法における部分配列のいずれか一つに真に含まれる．また，経歴・パターン法における上述のパターン冗長性は図 3 の(A)の区画を除く白い区画(区画エンコード法における非活性区画)に含まれるタプルのエンコード結果に含まれる．区画エンコード法によれば，すべての活性区画に含まれるタプルのエンコード結果には，パターン冗長性は含まれない．

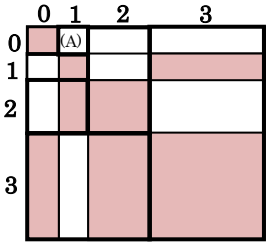


図 3 経歴・パターン法における部分配列（太線）と区画エンコード法における区画

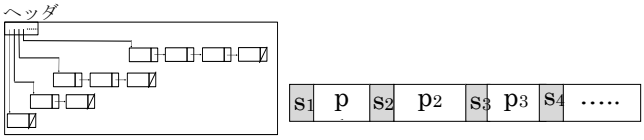
5. エンコード用データ構造

区画エンコード法では図 3 に見るように区画の数が経歴・パターン法の部分配列の数に比べて一般には，多くなる．このため，活性区画を順に登録する境界ベクトルテーブル(図 2）が大きくなり，与えられた座標に対してそれを含む区画の決定にやや時間を要し，エンコード時間が増加する．ここでは，エンコード時のメモリ量はやや増加するが，エンコード時間を抑制するデータ構造として，ハッシュにより境界ベクトルテーブルを実装する．ハッシュキーはタプルを表す座標が含まれる区画の境界ベクトルであり，ハッシュデータは区画 ID である．

6. 非シリアル実装とシリアル実装

通常の関係テーブルにおけるように多次元データセットはタプルの集合であり，タプルの格納順序には意味がないとしたとき，エンコード結果を格納する出力ファイルの方式として，区画 ID をヘッダ item として，同一区画 ID を持つタプルのパターンのみをノードブロックリストに格納する方式が考えられる(非シリアル実装・図 4(a))．これは記憶コストと検索コストの両面で有利な方式である．

一方，データの並び順や発生順を保存する必要があるアプリケーション，例えば，受信時間順を保存する必要があるセンサーデータの解析や文字列の出現順を保存する必要がある文書処理対しては，上記の非シリアルな実装方式は採用できない．<区画 ID,パターン>のエンコード結果を発生順に逐次格納する方式(シリアル実装・図 4(b))が必要である．



(a) 非シリアル実装　　(b) シリアル実装

図 4 非シリアル実装とシリアル実装

7. 経歴・パターン法との比較.

区画エンコード法では，パターン部の記憶コストは常に最小であることが保証されるが，経歴パターン法では図 1 のエンコード例 の 2 次元目の 01 のように，冗長な 0 が前置される場合があり，常に最小ではない．一方，区画エンコード法の区画の数と，経歴・パターン法の部分配列の数は図 3 に見るように一般に後者の方が少ない．そのため，境界ベクトルテーブルのサイズも経歴・パターン法の方が小さくなる．一方，単一値検索の検索候補となるタプルの数は非シリアル実装では，区画エンコードの方が小さいために検索コストの点では区画エンコード法の方が，有利であると考えられる．表 1 に両者のエンコード法の定性比較を示す．

表 1. 経歴・パターン法と区画エンコード法の比較

	HPMD	SPMD
エンコード/デコード用データ構造サイズ	小	中
経歴値または区画 ID の記憶コスト	小	大
パターンの記憶コスト	中	小
エンコード時間	小	中
デコード時間	小	小
DB 構築時間	中	中
検索量（非シリアル実装の場合）	大	小
検索時間（非シリアル実装の場合）	中	小
検索量の分散（非シリアル実装の場合）	中	小

8. むすび

タプルデータセットの一エンコード方式として，区画エンコード方式を提案した．今後は，区画エンコード法の解析と実装・評価を行うとともに，7 節で述べた，区画エンコード法におけるシリアル実装の際に問題となる区画数の増加に伴う性能劣化に対する対策を検討する予定である．

参考文献

[1] M.Makino, T.Tsuji, H.Higuchi, ” History-Pattern Implementation for Large-Scale Dynamic Multidimensional Datasets and Its Evaluations”, Proc. of DASFAA (2), 275-291(2015).