推薦論文

OLAP データキューブにおける範囲合計配列の更新

沙 飛[†] 古川哲也[††]

オンライン分析処理（OLAP）では，指定される範囲でデータを集約する範囲合計問合せの処理が必要である．前置合計モデルは，データ構成法であるデータキューブに対してデータの範囲合計を記憶し，範囲合計問合せの処理を高速化するものであるが，データの更新が重要な問題となる．本論文では，前置合計モデルを拡張し，データの更新コストを削減するモデルについて議論する．データの更新情報を前置合計データキューブとは独立に記憶することによって，更新ごとにデータキューブを再構築する必要がなくなる．また，問合せを評価するコストを削減するため，更新情報の変更について議論する．

Updates of Range Sum Arrays in OLAP Data Cubes

Fei Sha[†] and Tetsuya Furukawa[††]

A range sum query applies an aggregation operation in OLAP data cubes. The Prefix Sum method provides a cube of the same dimension and size as the original data cube with a prefix range-sum stored in each cell. Using the Prefix Sum approach, any range sum query can be evaluated at a constant cost. However, this method is very costly to maintain. This paper introduces a new model with an update information cube independent of a prefix cube. In this model prefix cubes do not recomputed when the original data is updated. Update information can be modified so as to evaluate rang sum queries efficiently.

1. はじめに

大量のデータが企業の様々な意志決定に利用されるようになった．データはデータウェアハウスに蓄積され，様々な視点から分析される．これまでの事務処理などにおけるデータベース利用は，主にオンライントランザクション処理（OLTP）のためであったが，意志決定支援では，事象（たとえば個々の売り上げ）の数値データを分析するオンライン分析処理（OLAP）が必要となる[5]．そのためのデータの構成法や処理法など，データウェアハウジングや OLAP に関する技術が重要な研究テーマとなっている[3]．

データウェアハウス[11]を構成する方法として，多次元データベースがある[1),6)]．多次元データベースは，日時，商品の種類，顧客，支店などの属性を用いて多次元空間を構成し，そこに数値データを配置するもので，最も一般的な手法の 1 つである．多次元データベースによるデータの構成は，その形状から，データキューブとも呼ばれる[8),10)]．

データキューブは，それぞれの分析に対して，指定される範囲でデータを集約し，それを供給する．すなわち，任意の範囲に対する数値データの集約処理を行う必要がある．集約処理には最大，最小，平均，計数，合計などの操作がある．範囲に対する数値データの合計を求める操作を範囲合計問合せといい，処理効率における重要な要因となる．範囲合計問合せの処理を効率化するために，前置合計を用いたデータ構成手法が提案されている[9]．前置合計モデルでは，原点からの範囲合計をあらかじめ計算し，前置合計として記憶することによって，どのような範囲合計問合せに対しても，そこで用いられる次元数 d に対して前置合計配列の 2^d 個のセルにアクセスすることで答えを求める

† 九州大学大学院経済学府
Department of Economic Engineering, Kyushu University
†† 九州大学大学院経済学研究院
Department of Economic Engineering, Kyushu University

本論文の内容は 2001 年 3 月の九州支部火の国情報シンポジウム 2001 にて報告され，九州支部火の国情報シンポジウム 2001 プログラム委員長により情報処理学会論文誌への掲載が推薦された論文である．

ことができる．

しかしながら，元のデータキューブの要素に更新が発生した際，範囲合計を記憶するデータキューブを更新しなければならずその更新コストが重要な問題となる．データウェアハウスのデータに基づいて作られたデータキューブは，データウェアハウスでデータが追加されたことによって更新されなければならない．前置合計モデルでは，前置合計データキューブを更新するため，時には莫大なコストがかかる．合理的な更新コストで新しい情報を受け入れることができなければ，データキューブは有用な分析ツールとはいえない．

一方，蓄えるデータを詳細化するにつれ，データキューブの次元数が増加し，データキューブの大きさが急速に増大することが指摘されている．データウェアハウスではデータが追加されるたびに，データキューブを再計算するのと同じコストで前置合計データを更新することは現実的ではない．すなわち，一定の時間で範囲合計問合せを評価する方法が必要とされるが，そのために更新の際に前置合計データキューブを再構築することは望ましくない．

更新コストの問題を解決するため，相対的な前置合計アプローチ[7]や階層的なバンドキューブ[4]が提案されている．更新コストを削減するため，これらのアプローチでは元のデータキューブを小さな範囲に分け，それぞれで前置合計を計算し記憶する．しかし，データキューブを小さく分ければ分けるほど範囲合計問合せを評価するコストが増加する．

本論文では，前置合計モデルを拡張し，データキューブの更新を効率良く行うモデルについて議論する．データの更新情報を前置合計データキューブとは別に記憶することで，更新ごとに前置合計データキューブを再構築する必要をなくし，データの更新コストを削減するものである．範囲合計問合せの評価コストは，更新情報が少なければ従来の前置合計モデルとほぼ同じとなる．同様のモデルについて議論している論文 2) では，問合せ処理を効率化するための更新情報の構成法を提案しているが，本論文では，更新情報が増えた際，前置合計データキューブと更新情報を再構成することで範囲合計問合せの処理効率を上げる方法について議論する．

本論文は，以下のように構成される．2 章では，従来の前置合計モデルを概観する．3 章では，更新情報を分離したモデルを導入し，そこでの範囲合計問合せの処理について議論する．4 章では，範囲合計問合せの処理効率を良くするため，更新情報を移動する手法を検討する．5 章では，更新情報を与えられた範囲の外へ拡散する手法について議論する．6 章は，結論と今後の課題である．

A_1	0	1	2	3	4	5	6	7	8
0	3	5	1	2	2	4	6	3	3
1	7	4	2	6	8	7	1	2	4
2	2	4	2	3	3	3	4	5	7
3	3	2	1	5	3	5	2	8	2
4	4	2	1	3	3	4	7	1	3
5	2	3	3	6	1	8	5	1	1
6	4	5	2	7	1	9	3	3	4
7	2	4	2	2	3	1	9	1	3

図 1 2 次元配列 A_1
Fig. 1 2-dimensional array A_1.

2. 前置合計モデル

データキューブに蓄える対象データは数値データである．数値データは d 次元で構成されるデータキューブ空間に配置される．次元集合は属性に対応し，$D = \{1, 2, \cdots, d\}$ で表す．それぞれの次元のサイズは次元の値の数であり，n_j $(1 \geq j \geq d)$ とする．したがって，d 次元データキューブを 1 つの d 次元配列 A で表すことができる．配列のサイズは $n_1 \times n_2 \times \cdots \times n_d$ $(n_j \geq 2, j \in D)$ である．各次元での位置をデータキューブにおける座標という．座標は 0 から始まるものとする．また，配列の要素をセルと呼ぶ．

例 1 **図 1** は 2 次元のデータキューブを表す配列 A_1 である．次元 1 のサイズは 9，次元 2 のサイズは 8 で，配列のサイズは $9 \times 8 = 72$ である．

d 次元の配列 A において，座標 $[x_1, x_2, \cdots, x_d]$ にあるセルを $A[x_1, x_2, \cdots, x_d]$ で表す．座標 $[l_1, l_2, \cdots, l_d]$ と $[h_1, h_2, \cdots, h_d]$ $(l_j \leq h_j)$ に対し，$[l_1, l_2, \cdots, l_d] : [h_1, h_2, \cdots, h_d]$ で $[l_1, l_2, \cdots, l_d]$ から $[h_1, h_2, \cdots, h_d]$ までの範囲を表す．したがって，その範囲のセル $A[x_1, x_2, \cdots, x_d]$ の座標は，$l_j \leq x_j \leq h_j$ $(1 \leq j \leq d)$ となる．

データキューブに配置されている数値データは様々な分析に使われる．それぞれの分析はデータキューブの指定される範囲でデータを集約し，その集約値を用いて行われるので，任意の範囲に対する数値データの集約処理を行う必要がある．これらの集約処理の中で，範囲に対する数値データの合計を求める操作を範囲合計問合せという．

定義 1 配列 A の座標 $[l_1, l_2, \cdots, l_d]$ と $[h_1, h_2, \cdots, h_d]$ に対し，範囲 $[l_1, l_2, \cdots, l_d] : [h_1, h_2, \cdots, h_d]$ の

P_1	0	1	2	3	4	5	6	7	8
0	3	8	9	11	13	17	23	26	29
1	10	19	22	30	40	51	58	63	70
2	12	25	30	41	54	68	79	89	103
3	15	30	36	52	68	87	100	118	134
4	19	36	43	62	81	104	124	143	162
5	21	41	51	76	96	127	152	172	192
6	25	50	62	94	115	155	183	206	230
7	27	56	70	104	128	169	206	230	257

図 2 配列 A_1 の前置合計配列 P_1
Fig. 2 Prefix-sum array P_1 for A_1.

セルの合計を求める問合せを範囲合計問合せといい，$Sum([l_1, l_2, \cdots, l_d] : [h_1, h_2, \cdots, h_d])$ で表す．その値は，$\sum_{x_1=l_1}^{h_1} \sum_{x_2=l_2}^{h_2} \cdots \sum_{x_d=l_d}^{h_d} A[x_1, x_2, \cdots, x_d]$ である．

例 2 図 1 の配列 A_1 に対し，範囲 $[3,2] : [6,4]$ のセルの合計を求める範囲合計問合せは $Sum([3,2] : [6,4])$ であり，その値は $\sum_{x_1=3}^{6} \sum_{x_2=2}^{4} A_1[x_1, x_2] = 45$ である．

範囲合計問合せを効率良く評価するため，範囲合計をあらかじめ計算し，一定の時間で問合せを評価できる前置合計モデルが提案されている[9]．

定義 2 配列 A に対し，$Sum([0,0,\cdots,0] : [x_1, x_2, \cdots, x_d])$ を配列 A と同じ構成を持つ配列 P の座標 $[x_1, x_2, \cdots, x_d]$ に配置する．配列 P を前置合計配列という．

すなわち，前置合計モデルは，配列 A と同じ構成の配列 P を新たに作り，それぞれのセルに配列 A の座標 0 からそのセルの座標までの範囲合計を配置するものである．

例 3 図 1 の配列 A_1 に対する前置合計配列は**図 2** である．セル $P_1[4,0]$ の値は配列 A_1 のセル $A_1[0,0]$ から $A_1[4,0]$ までの合計 13 である．同様に，セル $P_1[2,1]$ は $A_1[0,0]$ から $A_1[2,1]$ までの合計 22 である．$P_1[8,7]$ は配列 A_1 のすべてのセルの合計である．

前置合計モデルでは，任意の範囲合計問合せに対し，前置合計配列 P の必要なセルにアクセスすることで計算できる．定理 1 は配列 P で最大 2^d 個のセルにアクセスするだけで，いかなる範囲合計問合せも評価できることを示している[9]．$s(j)$ は配列 P のセルの符号である．

定理 1 前置合計配列 P に対し，

$$Sum([l_1, l_2, \cdots, l_d] : [h_1, h_2, \cdots, h_d])$$
$$= \sum_{\forall x_j \in \{l_j - 1, h_j\}} \left\{ \left(\prod_{i=1}^{d} s(i) \right) * P[x_1, x_2, \cdots, x_d] \right\}.$$

ここで $\forall j \in D$ に対し，

$$s(j) = \begin{cases} 1, & if \quad x_j = h_j, \\ -1, & if \quad x_j = l_j - 1. \end{cases}$$

$x_j = -1$ の場合，$P[x_1, x_2, \cdots, x_d] = 0$ である．

例 4 配列が 2 次元の場合，前置合計配列を用いた範囲合計問合せの評価は

$$Sum([l_1, l_2] : [h_1, h_2])$$
$$= P[h_1, h_2] - P[h_1, l_2 - 1] - P[l_1 - 1, h_2] + P[l_1 - 1, l_2 - 1]$$

である．したがって，図 2 の前置合計配列 P_1 を用いた範囲 $[3,2] : [6,4]$ の範囲合計の計算は以下となる．

$$Sum([3,2] : [6:4])$$
$$= P_1[6,4] - P_1[6,1] - P_1[2,4] + P_1[2,1]$$
$$= 124 - 58 - 43 + 22$$
$$= 45.$$

しかしながら，このモデルではデータの更新が重要な問題となる．配列 A のセルが更新されると，その値が計算に使われた配列 P のセルはすべて更新されなければならない．前置合計モデルを更新するにはコストがかかる．

例 5 配列 A_1 のセル $A_1[3,2]$ が更新され，(-2) が加えられるとき，$A_1[3,2]$ は 3 から 1 に変わる．配列 P_1 ではセル $[3,2]$ から最後までのすべてのセルを更新しなければならない．すなわち，$x_1 \geq 3$，$x_2 \geq 2$ となるすべてのセル $P_1[x_1, x_2]$ は (-2) が加えられる．更新後の前置合計配列を P_1' とすると

$$P_1'[x_1, x_2] = P_1[x_1, x_2] + (-2) \ (x_1 \geq 3, x_2 \geq 2)$$

である．

更新コストの問題を解決するため，相対的な前置合計アプローチ[7]や階層的なバンドキューブ[4]が提案されている．相対的な前置合計アプローチでは配列 A を均等に小さく分け，それぞれの範囲に対する前置合計を記憶する．更新が発生したとき，更新されたセルを含む範囲だけを再計算すればよいので，分割された範囲が小さいほど更新効率が上がる．しかし，範囲合計問合せを評価するとき，それぞれの範囲で計算するため評価するコストが高くなる．階層的なバンドキューブは階層的な分け方で配列 A を分割しさらに更新効率を上げているが，相対的な前置合計アプローチと同じく範囲合計問合せを評価するコストが高くなる．

3. 更新情報を持つ前置合計モデル

前置合計モデルでは，範囲合計をあらかじめ計算し前置合計配列に記憶しておくことによって，範囲の大きさにかかわらず一定の時間で範囲合計問合せを評価することができる．しかし，配列 A でセルが更新されるとき，前置合計配列 P では関連するセルをすべて再計算しなければならない．本章では，前置合計モデルとほぼ同じコストで範囲合計問合せを評価することができ，データの更新を容易に行える方法について検討する．

前置合計配列 P での更新コストが高い原因は，次元のサイズが大きくなるにつれデータキューブも大きくなり，配列 A のセルが配列 P の多数のセルと関連するからである．配列 A のセルが更新されると，配列 P の関連するセルも更新されなければならない．更新コストは次元のサイズを n とすると $O(n^d)$ となり，範囲合計を記憶する前置合計配列を再構築するのと同等のコストになる．更新情報を独立に記憶することによって配列 A のセルの更新ごとに前置合計配列を再計算する必要がなくなる．このようなモデルを議論するために更新情報を明らかにする．

定義 3 配列 A が更新され，A' になったとき，$U = A' - A$ を配列 A の更新情報配列という．配列 U のセル $U[x_1, x_2, \cdots, x_d]$ を配列 A のセル $A[x_1, x_2, \cdots, x_d]$ の更新情報という．

セル $A[x_1, x_2, \cdots, x_d]$ が更新され，配列 U の更新情報 $U[x_1, x_2, \cdots, x_d]$ として記憶されるが，更新情報がないセルについては 0 となる．更新情報が存在するセルのみを記憶するなどして，更新情報配列 U を物理的に圧縮できる．ここで，更新情報配列 U のサイズは配列のサイズにかかわらず，更新情報の数に比例する．本論文で配列 A や配列 P と同様に議論するため更新情報配列として扱う．

例 6 2 次元の配列 A_1 ではセル $A[3,2]$ が 3 から 1 に更新されるとき，更新情報は (-2) となり，配列 A_1 に対応する更新情報配列 U_1 ではセル $U[3,2]$ として記憶される（**図 3**）．更新情報がないセルは値の 0 を省略している．

本論文では前置合計配列に更新情報配列を加えたモデルについて議論する．更新処理は更新情報配列に更新情報を加えるだけなので前置合計配列の再計算の必要がなくなり，更新コスト $O(n^d)$ は更新情報配列に更新情報を加えるコストのみとなる．範囲合計問合せの評価は，前置合計配列を用いた結果に更新情報配列による修正を加えなければならない．

U_1	0	1	2	3	4	5	6	7	8
0									
1									
2				-2					
3									
4									
5									
6									
7									

図 3 配列 A_1 の更新情報配列 U_1
Fig. 3 Update information array U_1 of A_1.

定理 2 前置合計配列 P，更新情報配列 U に対し，

$$Sum([l_1, l_2, \cdots, l_d] : [h_1, h_2, \cdots, h_d])$$
$$= \sum_{\forall x_j \in \{l_j - 1, h_j\}} \left\{ \left(\prod_{i=1}^{d} s(i) \right) * P[x_1, x_2, \cdots, x_d] \right\}.$$
$$+ \sum_{i_1=l_1}^{h_1} \sum_{i_2=l_2}^{h_2} \cdots \sum_{i_d=l_d}^{h_d} U[i_1, i_2, \cdots, i_d].$$

ここで $\forall j \in D$ に対し，

$$s(j) = \begin{cases} 1, & if \quad x_j = h_j, \\ -1, & if \quad x_j = l_j - 1. \end{cases}$$

$x_j = -1$ の場合，$P[x_1, x_2, \cdots, x_d] = 0$ である．

証明： 配列 A を前置合計配列 P が作られた配列，配列 A' を現在の配列とする．$U = A' - A$ なので，$A' = A + U$ である．したがって，配列 A' における範囲合計問合せの評価は配列 A における問合せの評価にその範囲の U の値を加えたものとなる．定理 1 より配列 A における評価は P で計算でき，その結果に U を加えたものが A' における範囲合計問合せの評価結果となる． □

例 7 配列 A_1 でセル $A_1[3,2]$ が 3 から 1 に更新されるとき，範囲 $[3,2] : [6,4]$ に対する範囲合計問合せは配列 P_1 で評価された結果に更新情報 $U_1[3,2]$ が加えられる．

$$Sum([3,2] : [6,4])$$
$$= P_1[6,4] - P_1[6,1] - P_1[2,4] + P_1[2,1] + U_1[3,2]$$
$$= 124 - 58 - 43 + 22 + (-2)$$
$$= 43$$

更新情報配列 U を持つモデルでは範囲合計問合せに対し，更新される前の前置合計配列 P で評価し，その結果に更新後の差分を配列 U で計算し加えることによって範囲合計問合せを評価することができる．このモデルは前置合計モデルより範囲合計問合せを評価するコストが更新情報を計算する分高くなる．更新情

報配列のサイズは更新情報の数に比例するので，更新情報が少ないときは，従来の前置合計モデルとほぼ同じコストで範囲合計問合せを評価できる．

更新情報が増えると更新情報配列による修正コストも高くなる．最も単純には，定期的に前置合計配列を再計算し，更新情報配列をクリアすることで対処できる．しかし，この方法では前置合計配列の再構築中には範囲合計問合せの処理ができない．配列のサイズが大きく再構築に時間を要する場合は，システム利用上重大な問題となる．本論文では前置合計配列の再構築よりも十分に短い時間，すなわちシステム利用のうえで問題とはならない程度の時間で前置合計配列と更新情報配列に変更を加え，範囲合計問合せの処理効率を上げる方法について検討する．

処理効率の向上のための前置合計配列と更新情報配列の変更に対しては次の方法を考える．

- 更新情報の部分的な反映
  更新情報を前置合計配列の一部に反映させる．前置合計配列の変更は部分的なので，再構築に比べ短時間となる．
- 更新情報の削減
  範囲合計問合せの処理に用いられる更新情報の数を減らす．
  - 更新情報配列の更新情報を削減する．
  - 範囲合計問合せによく用いられる範囲を想定し，その範囲内の更新情報を削減する．

これらの方法を具体化するうえで，前置合計配列と更新情報配列の変更の方針に注意しなければならない．方針には次のようなものがある．

- 最終的に更新情報をなくし，すべての更新情報を前置合計配列に反映することを目指す．
- 更新情報配列を範囲合計問合せの処理に用いることを前提とし，処理の効率化のみを目指す．

前置合計配列と更新情報配列の変更では，範囲合計問合せを正しく評価できることを保障しなければならない．そのために，等価の概念を導入する．

定義 4 前置合計配列と更新情報配列の組 (P,U), (P',U') は，任意の範囲合計問合せに対して，評価した結果が同じになるとき，等価であるといい，$(P,U) \equiv (P',U')$ で表す．

等価な前置合計配列と更新情報配列に変更すれば，範囲合計問合せを正しく評価することができる．以下の章では，与えられた前置合計配列と更新情報配列に対し，等価性を満足して範囲合計問合せの処理効率を向上させるための変更について議論する．

4. 更新情報の移動

本章では，更新情報配列の変更方法として更新情報の配列内での移動とそれにともなう前置合計配列の変更について議論し，移動を用いた更新情報の前置合計配列への部分的な反映について検討する．

定義 5 更新情報配列 U と U の座標 $[w_1, w_2, \cdots, w_d]$, $[w'_1, w'_2, \cdots, w'_d]$ で，セル $U[w_1, w_2, \cdots, w_d]$ の値をセル $U[w'_1, w'_2, \cdots, w'_d]$ に加え，セル $U[w_1, w_2, \cdots, w_d]$ を 0 にすることを配列 U における更新情報 $U[w_1, w_2, \cdots, w_d]$ の $[w_1, w_2, \cdots, w_d]$ から $[w'_1, w'_2, \cdots, w'_d]$ への移動といい，$U([w_1, w_2, \cdots, w_d] \rightarrow [w'_1, w'_2, \cdots, w'_d])$ で表す．

更新情報の移動によって更新情報配列 U が配列 U' に変更されるとき，前置合計配列 P と更新情報配列 U' の情報では正しい範囲合計を与えることはできない．データベースの情報を維持するため，$(P,U) \equiv (P',U')$ となるように範囲合計配列 P を更新情報配列の変更に従って P' に変更する．更新情報配列における更新情報の移動 $U([w_1, w_2, \cdots, w_d] \rightarrow [w'_1, w'_2, \cdots, w'_d])$ に対して，前置合計配列 P を以下のように修正する．座標が $(\forall x_j \geq w_j \text{ or } \forall x_j \geq w'_j)$ かつ $\exists x_j < max(w_j, w'_j)$ のセルに対し，

$$P'[x_1, x_2, \cdots, x_d] = P[x_1, x_2, \cdots, x_d] + Q$$

$$Q = \begin{cases} -U[w_1, w_2, \cdots, w_d], & if \ \ \forall x_j \geq w_j, \\ U[w_1, w_2, \cdots, w_d], & otherwise. \end{cases}$$

定理 3 組 (P,U) における $U([w_1, w_2, \cdots, w_d] \rightarrow [w'_1, w'_2, \cdots, w'_d])$ と移動の結果 U' および前置合計配列 P の修正結果 P' に対し，$(P,U) \equiv (P',U')$ である．

証明： 配列 A を配列 P が構成された配列，A_0 を $A_0 = A + U$ となる配列とする．更新情報 $U[w_1, w_2, \cdots, w_d]$ を前置合計配列 P に反映させると $x_j \geq w_j$ となる座標 $[x_1, x_2, \cdots, x_d]$ に $U[w_1, w_2, \cdots, w_d]$ が加えられる．変更された結果を P_1 とし，配列 A に更新情報 $U[w_1, w_2, \cdots, w_d]$ を反映した結果を A_1 とする．A_1 の座標 $[w'_1, w'_2, \cdots, w'_d]$ から $U[w_1, w_2, \cdots, w_d]$ を引いた配列を A' とすると，$A_0 = A' + U'$ である．A' の前置合計配列は P_1 に $A' - A_1$ の更新情報を反映したものとなるので，P_1 の $x_j \geq w'_j$ となる座標 $[x_1, x_2, \cdots, x_d]$ から $U[w_1, w_2, \cdots, w_d]$ を引いたものとなる．結果を P_2 とすると，P_2 は移動に対する P の修正結果 P' に一致する．すなわち，P', U' は A_0 に対する範囲合計問合せを評価するので，$(P,U) \equiv (P',U')$ である． □

U_1'	0	1	2	3	4	5	6	7	8
0									
1			-2						
2				(-2)					
3									

(a)

P_1'	0	1	2	3	4	5	6	7	8
0									
1			24	32	42	53	60	65	72
2			32						
3			38						
4			45						
5			53						
6			64						
7			72						

(b)

図 4 更新情報の移動
Fig. 4 Movement of update information.

例 8 範囲合計配列 P_1（図 2）と更新情報配列 U_1（図 3）に対し，$U([3,2] \rightarrow [2,1])$ の移動を行うと更新情報配列 U_1 は**図 4** (a) の U_1' となる．前置合計配列 P は，セル $P[x_1, x_2]$（$x_1 \geq 2$，$x_2 \geq 1$ かつ $\exists x_1 < 3$，$\exists x_2 < 2$）が影響されるので，$P'[x_1, x_2] = P[x_1, x_2] - (-2)$ のように更新される．図 4 (b) は更新後の配列 P_1' の更新されたセルを示している．

更新情報の移動を使って更新情報を部分的に前置合計配列に反映することができる．更新情報 $U[i_1, i_2, \cdots, i_d]$ を前置合計配列 P に反映するとき，前置合計配列 P ではセル $P[x_1, x_2, \cdots, x_d]$ $(x_j \geq i_j)$ をすべて変更しなければならない．更新情報 $U[i_1, i_2, \cdots, i_d]$ より座標が小さい更新情報を更新情報 $U[i_1, i_2, \cdots, i_d]$ の後に前置合計配列 P に反映すると，配列 P ではセル $P[x_1, x_2, \cdots, x_d]$ $(x_j \geq i_j)$ を再計算することになる．更新情報の移動を使って更新情報を部分的に前置合計配列に反映するとき，前置合計配列 P のセルの再計算を避けるためには，座標が小さい更新情報を配列 P ではこの更新情報だけが影響する範囲内で移動しなければならない．このような移動によって前置合計配列 P で移動される範囲に更新情報を部分的に反映することができ，全体の更新情報をなくすコストも小さくなる．

更新情報の移動では，更新情報が存在する座標に移動すれば更新情報の数が減る．また，複数の更新情報を同時に移動し 1 カ所に集めることができる．これによって更新情報を削減し，範囲合計問合せの処理効率を上げることができる．更新情報配列を範囲合計問合せの処理に用いることを前提とする場合，任意の範囲の更新情報を任意の場所にまとめてもよいが，1 個の更新情報を移動するときと同様な問題がある．すなわち，任意の範囲の更新情報を任意のセルにまとめるとほかの更新情報を移動するとき，前置合計配列で同じ範囲のセルを再計算する場合がある．最終的に更新情報をすべてなくす場合，前置合計配列 P での変更コストを削減するために，これらの更新情報を前置合計配列 P に部分的に反映させなければならない．これらの更新情報を配列 P でこれらの更新情報だけが影響する範囲内で座標が大きい場所に移動し配列 P に部分的に反映することができる．

よく使われる範囲を想定し，その範囲の更新情報を削減するためには，範囲内の更新情報を範囲外の 1 カ所の座標にまとめる方法がある．これによりその範囲だけではなく，更新情報配列の更新情報も削減できる．しかし，前置合計配列 P の変更コストを考慮すると，ある範囲内の更新情報を範囲外のそれぞれ一番近い場所に同時に移動したほうが前置合計配列 P の変更コストが小さい場合がある．更新情報をそれぞれの方向に同時に移動するとき，1 個の更新情報を移動するのと同じように，最終的にすべての更新情報をなくすことを考える場合，前置合計配列 P でこれらの更新情報だけが影響する範囲内で移動しなければならない．

5. 更新情報の拡散

4 章では更新情報の移動による範囲合計問合せ処理の効率化について議論した．しかし，更新情報の移動にともなって前置合計配列では多くのセルを修正しなければならない．本章では，更新情報配列である範囲から更新情報をなくした場合，前置合計配列で同じ範囲にあるセルだけを修正することによって，データベースの情報を維持する方法について検討する．

更新情報配列 U のある更新情報 $U[i_1, i_2, \cdots, i_d]$ を前置合計配列 P に反映させると，配列 P ではセル $P[x_1, x_2, \cdots, x_d]$ $(x_j \geq i_j)$ をすべて更新しなければならない．ここでは更新情報 $U[i_1, i_2, \cdots, i_d]$ に対して，ある範囲 $[i_1, i_2, \cdots, i_d] : [h_1, h_2, \cdots, h_d]$ だけを更新させることができる．このような更新に対し，更新情報配列 U では更新した範囲の更新情報をなくしたけれども，更新していない範囲に対して更新情報を記憶しなければならない．更新情報を対応するセルにそれぞれ記憶する．

定義 6 更新情報 $U[w_1, w_2, \cdots, w_d]$ と座標 $[h_1, h_2, \cdots, h_d]$ $(w_j < h_j)$ に対し，2^d 個の更新情報を次のよ

うに変更する操作を更新情報 $U[w_1, w_2, \cdots, w_d]$ の座標 $[h_1, h_2, \cdots, h_d]$ に対する拡散という．

$$U'[x_1, x_2, \cdots, x_d] = U[x_1, x_2, \cdots, x_d] - \prod_{i=1}^{d} s(i) * U[w_1, w_2, \cdots, w_d] \quad (\forall x_j \in \{w_j, h_j + 1\}).$$

ここで $\forall j \in D$ に対し，

$$s(j) = \begin{cases} 1, & if \quad x_j = w_j, \\ -1, & if \quad x_j = h_j + 1. \end{cases}$$

更新情報配列 U で更新情報 $U[w_1, w_2, \cdots, w_d]$ を座標 $[h_1, h_2, \cdots, h_d]$ に対して拡散した結果は，前置合計配列 P で更新情報 $U[w_1, w_2, \cdots, w_d]$ を範囲 $[w_1, w_2, \cdots, w_d] : [h_1, h_2, \cdots, h_d]$ に反映させるとき，配列 U の変化と同じである．したがって，$(P, U) \equiv (P', U')$ となるように配列 P を修正する．

$$P'[x_1, x_2, \cdots, x_d] = P[x_1, x_2, \cdots, x_d] + U[w_1, w_2, \cdots, w_d] \quad (w_j \le x_j \le h_j).$$

定理 4 前置合計配列と更新情報配列の組 (P, U) において，配列 U では更新情報 $U[w_1, w_2, \cdots, w_d]$ を座標 $[h_1, h_2, \cdots, h_d]$ $(w_j < h_j)$ に対し拡散した結果 U' および P を修正した結果 P' に対し，$(P, U) \equiv (P', U')$ である．

証明： 更新情報配列 U で更新情報 $U[w_1, w_2, \cdots, w_d]$ の座標 $[h_1, h_2, \cdots, h_d]$ $(w_j < h_j)$ に対する拡散で変更される座標に適当に番号をつけ，$Y_1, Y_2, \cdots, Y_{2^d}$ $(Y_i = [y_{i1}, y_{i2}, \cdots, y_{id}])$ とする．$U_0 = U$，$U_i = U_{i-1} + c_i$ $(1 \le i \le 2^d$，c_i は座標 Y_i に対する変更) とし，$P_0 = P$，P_i を P_{i-1} の $x_j \ge y_{ij}$ となる座標 $[x_1, x_2, \cdots, x_d]$ から c_i $(1 \le i \le 2^d)$ を引いた配列とする．明らかに $(P_i, U_i) \equiv (P_{i-1}, U_{i-1})$ なので $(P_0, U_0) \equiv (P_n, U_n)$ である．U_n は更新情報配列 U に対するすべての変更を行ったものなので U' に一致する．範囲合計配列では範囲 $[w_1, w_2, \cdots, w_d] : [h_1, h_2, \cdots, h_d]$ の外の座標で $\exists x_j < w_j$ となるセルは変更されないため $P_0[x_1, x_2, \cdots, x_d] = P_n[x_1, x_2, \cdots, x_d]$ である．範囲外の $x_j \ge w_j$ となる任意の座標 $[x_1, x_2, \cdots, x_d]$ に対して，座標が $y_{ij} \le x_j$ となる Y_i の数は 2^k 個となる．k は座標 $[x_1, x_2, \cdots, x_d]$ の中で h_j より大きい x_j の数である．Y_i の座標 $[y_{i1}, y_{i2}, \cdots, y_{id}]$ の中で $y_{ij} = h_j$ となる y_{ij} が偶数であれば c_i は負となり，y_{ij} が奇数であれば c_i は正となる．2^k 個の Y_i の中で偶数となる Y_i と奇数となる Y_i は同じ数なので，$P_n[x_1, x_2, \cdots, x_d]$ は $P_0[x_1, x_2, \cdots, x_d]$ と等しい．範囲内で Y_i は座標 $[w_1, w_2, \cdots, w_d]$ しかなく，任意の座標 $[x_1, x_2, \cdots, x_d]$ に対して $x_j \ge w_j$ なので，$P_n[x_1, x_2, \cdots, x_d]$ は $P_0[x_1, x_2, \cdots, x_d]$ に $U[w_1, w_2, \cdots, w_d]$ を加えたものである．したがって，P_n は拡散に対して P を変更した P' に一致する．$P_0 = P$，$U_0 = U$ なので $(P, U) \equiv (P', U')$ が成り立つ． □

U_2	0	1	2	3	4	5	6	7	8
0									
1		−3	−2	5					
2		1	1	2					
3		−4	3	−4					
4									

(a)

U_2'	0	1	2	3	4	5	6	7	8
0									
1					0				
2					4				
3					−5				
4		−6	2	3	1				

(b)

P_1''	0	1	2	3	4	5	6	7	8
0									
1		16	17	30					
2		26	27	45					
3		24	38	51					
4									
5									
6									
7									

(c)

図 5 更新情報の拡散
Fig. 5 Dispersion of update information.

更新情報配列 U では範囲 $[l_1, l_2, \cdots, l_d] : [h_1, h_2, \cdots, h_d]$ におけるすべての更新情報を同時に拡散することができる．これらの更新情報を前置合計配列 P に反映すると次のようになる．

$$P'[x_1, x_2, \cdots, x_d] = P[x_1, x_2, \cdots, x_d] + \sum_{i_1=l_1}^{x_1} \sum_{i_2=l_2}^{x_2} \cdots \sum_{i_d=l_d}^{x_d} U[i_1, i_2, \cdots, i_d] \quad (l_j \le x_j \le h_j)$$

例 9 更新情報配列 U_2（**図 5** (a)）と座標 $[1, 1]$,

[3,3] に対し，範囲 [1,1] : [3,3] における更新情報をすべて拡散する．その結果は図 5 (b) となる．前置合計配列 P_1 で変更されたセルは図 5 (c) となる．

更新情報の拡散を使って特定の範囲の更新情報をなくすことができる．また前置合計配列 P ではこの範囲のセルだけを変更すればよいので，移動より配列 P の変更コストが小さくなる．最終的に更新情報をなくし，すべての更新情報を前置合計配列に反映するコストを考えるとき，これらの更新情報を部分的に前置合計配列 P に反映しなければならない．任意の更新情報を拡散した後にそれより座標が小さい更新情報を同じ座標に対し拡散すると前置合計配列 P では同じ範囲のセルを再計算するので全体の更新情報をなくすコストが高くなる．拡散を使って更新情報を部分的に前置合計配列 P に反映するために座標が一番小さい更新情報を含む範囲の更新情報を同時に拡散しなければならない．

6. むすび

本論文では，OLAP データキューブにおける範囲合計問合せの処理を効率化するための前置合計モデルを拡張し，データの更新に対応できるモデルについて議論した．前置合計モデルでは更新のたびに前置合計配列を更新しなければならず，その更新コスト $O(n^d)$ は非実用的である．更新情報配列を持つモデルでは，範囲合計問合せを評価するコストをほとんど増やさず，更新コストを大幅に削減した．

データキューブのデータを更新するとき範囲合計問合せの処理ができないため，データの更新時間と更新範囲を制御することが必要となる．更新情報をデータや前置合計配列とは独立に記憶しているため，更新時間と更新範囲を制御することができる．更新時間と更新範囲の制御は更新情報の移動，拡散を使い分けることによって実現される．また，最終的にすべての更新情報をなくすか否かによってそれぞれの手法を実行する手順も違ってくる．

更新情報は，すべて前置合計配列に反映されているのが最も良い状態である．しかし，システムの稼働状況などから，即時には不可能な場合もある．今後は，これらの手法をいつ，どの部分に適用するかを判断する方法を検討する予定である．

参考文献

1) Agrawal, R., Gupta, A. and Sarawagi, S.: Modeling Multidimensional Databases, *Proc. IEEE 13th Intl. Conf. on Data Engineering*, pp.234–243 (1997).
2) Chun, S.J., Chung, C.W., Lee, J.H. and Lee, S.L.: Dynamic Update Cube for Range-Sum Queries, *Proc. 27th Intl. Conf. on Very Large Data Bases*, pp.521–530 (2001).
3) Chaudhuri, S. and Dayal, U.: An Overview of Data Warehousing and OLAP Technology, *ACM SIGMOD Records*, Vol.26, No.1, pp.65–74 (1997).
4) Chan, C.Y. and Ioannidis, Y.E.: Hierarchical Cubes for Range-Sum Queries, *Proc. 25th Intl. Conf. on Very Large Data Bases*, pp.675–686 (1999).
5) Codd, E.F.: Providing OLAP (On-line Analytical Processing) to User-analysts: An IT Mandate, Technical Report, E.F. Codd and Associates (1993).
6) Furukawa, T. and Sha, F.: Utilizing Functional Dependencies in Multidimensional Databases, *Intl. Conf. on Database Applications in Non-Traditional Environments*, pp.59–65, IEEE (1999).
7) Geffner, S., Agrawal, D., Abbadi, A. and Smith, T.: Relative Prefix Sums: An Efficient Approach for Querying Dynamic OLAP Data Cubes, *Proc. IEEE 15th Intl. Conf. on Data Engineering*, pp.328–335 (1999).
8) Gray, J., Bosworth, A., Layman, A. and Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-tabs and Subtotals, *Proc. IEEE 12th Intl. Conf. on Data Engineering*, pp.152–159 (1996).
9) Ho, C., Agrawal, R., Megiddo, N. and Srikant, R.: Range Queries in OLAP Data Cubes, *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, pp.73–88 (1997).
10) Harinarayan, V., Rajaraman, A. and Ullman, J.D.: Implementing Data Cubes Efficiently, *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, pp.205–216 (1996).
11) Inmon, W.H.: *Building the Data Warehouse*, 2nd edition, John Wiley & Sons (1996).

(平成 13 年 9 月 14 日受付)
(平成 14 年 7 月 2 日採録)

推薦文

従来「更新」がないものとして考えられた機構を，「更新」がある場合にも対処できるように新しい工夫を施した点を評価し，推薦に値すると認めた．

（九州支部火の国情報シンポジウム 2001
プログラム委員長　岡田 直之）

沙　飛

平成 10 年九州大学経済学部退学（飛び級のため）．平成 12 年同大学大学院経済研究科経済工学専攻修士課程修了．現在同大学院経済学府経済工学専攻博士課程在学中．データウェアハウス，多次元データベース，オンライン分析処理，問合せ処理の効率化等の研究に従事．電子情報通信学会学生会員．

古川　哲也（正会員）

昭和 58 年京都大学工学部卒業．昭和 60 年同大学大学院修士課程修了．昭和 63 年九州大学大学院博士後期課程修了．工学博士．九州大学工学部助手，大型計算機センター講師，同助教授，経済学部助教授を経て，現在同大学大学院経済学研究院助教授．この間，平成 7 年米国パデュ大学客員研究員．データベースの設計論・質問処理論，情報システムの研究に従事．電子情報通信学会，ACM，IEEE，日本 OR 学会等会員．