DEWS2007 E7-6

多構造データベース演算を用いたログデータ分析の試み

涌波　信弥† 　大森　匡† 　星　守†

† 電気通信大学大学院情報システム学研究科　〒 182-8585 東京都調布市調布ヶ丘 1-5-1
E-mail: †{wakunami,omori}@hol.is.uec.ac.jp

あらまし　計算機システムのログデータ系列や Web サイトの利用者ログを集め，それを分析して何らかの有意な情報を取り出したい，という要求は依然として大きい．このために，著者らは，データキューブモデルの枠組みの下でログデータの多次元分析を行う機構「アイテムセットキューブ」を提案している．一方，近年，多構造データベース (MSDB) と呼ばれる数値データキューブの一種が提案され，有意な情報抽出のための演算が提案されている．本稿では，MSDB の演算をログデータの多次元分析に適用し，その適用方法と試行結果を報告する．
キーワード　データウェアハウス, OLAP, ログデータ分析, 多構造データベース

A Study of Log-Data Analysis by using Multi-Structural Databases

Shin-ya WAKUNAMI†, Tadashi OHMORI†, and Mamoru HOSHI†

† Graduate School of Information Systems, The University of Electro-Communications
1-5-1 Chohugaoka, Chohu, Tokyo, 182-8585 Japan
E-mail: †{wakunami,omori}@hol.is.uec.ac.jp

Abstract　A today's hot problem is to find meaningful information from massive amounts of log-data sequences, which are generated by computer systems or Web site. To alleviate this problem, the authors proposed a new data-mining tool termed an **itemset cube** system. However, the itemset cube system still depends on human knowledge to judge meaningful sequences of log-data. To solve this matter, this paper applies an idea of a recently-proposed data-cube tool named a *Multi-Structural Data Base* (MSDB). A method to apply operations of the MSDB to log-data analysis and its effects in the environment of the itemset cube system are described.
Key words　data mining, OLAP, log-data analysis, sequential data, Web and Internet.

1. 研究の背景と目的

計算機システムのログデータ系列や Web サイトの利用者ログを集め，それを分析して何らかの有意な情報を取り出したい，という要求は依然として大きい．この要求のための技法は，大別して次の 3 つがある：

- 1. データキューブを使った多次元データ分析に基づくもの．
- 2. 単位時間あたりに生じたイベントの組合せを元にした系列データマイニングを行うもの．(例えば，計算機システムのメッセージ解析において，5 分単位で生じたメッセージの頻出組合せの時間変化を求めて，計算機内の状態を推定する，など).
- 3. ベイズモデルやマルコフ過程モデルを使ってログの生成モデルを学習する技法.

このうち，著者らは，項目 1 と 2 を統合して扱えるデータキューブシステムとして，アイテムセットキューブとよぶ機構を提案している．

アイテムセットキューブの特徴は，数値データキューブを構成する多次元分析空間に沿った頻出アイテムセット系列の生成を行う点である．文献 [1] では，Web サイトのアクセスログを集積しアイテムセットキューブによる分析を行って，アクセスログから，分析者の指定した多次元空間上での集計値 (ヒット数) の系列，および，それに応じた分析空間内での高頻度アイテムセット (つまり，特定の制約条件下で頻繁にアクセスされたページの組合わせ) の列，の 2 つの情報系列を生成できることを示してきた．

例えば，図 1 は Web サイトアクセスログ解析の結果である．この図は，ページ種類別に，その種類のページを訪れるユーザの数，及び，そこで発生したアクセスの頻出ページの組を示している．このような情報を，多次元データキューブの枠組の中で計算する．しかし，アイテムセットの系列の価値判断には，最終的には専門的知見が必要である．例えば，図 1 では，10-12 月付近の講演会ページへのアクセスの内容に特徴的な動作が見られることがわかっている．しかし，この事実は自動的に検出

できていない．すなわち，ログデータ系列の分析においては，多次元分析空間内での数値集計列の生成やアイテムセットの系列の生成，だけでなく，もう 1 つ，アイテムセット系列のうち内容が特徴的な部分を自動的に判定する機能も必要である．

従来，数値列や記号列からの重要部分の検出には，学習モデル技法を使うことが多い．例えば，NEC の WebSAM [6] と言う統合システム運用管理製品にある ASManager と言う自律運用管理ソフトウェアの機能がある（図 2）．図では，単純に単位時間あたりの計算機メッセージ数の時系列を追いかけ，その数値時系列の学習モデルに基づいて異常状態を検出する (文献 [6]).

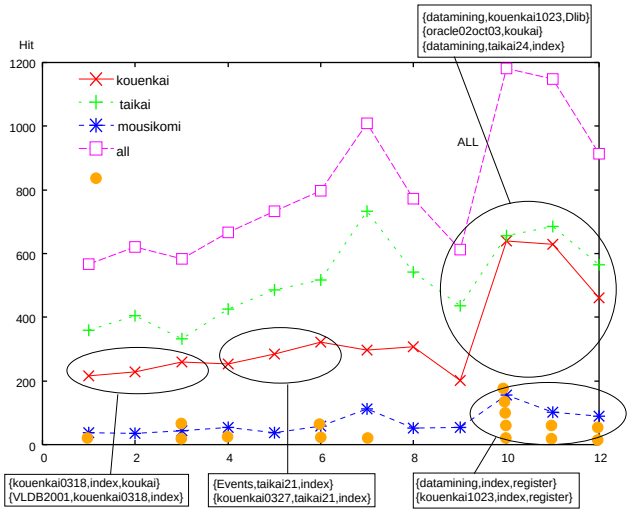

図 1　各イベント種類の月別ヒット数

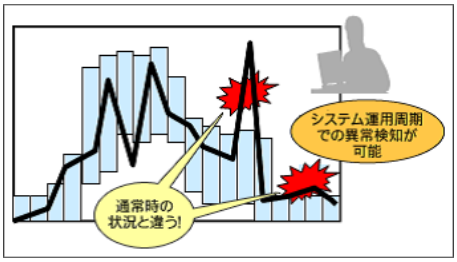

図 2　WebSAM ASManager の機能 ([6] より)

一方，アイテムセットキューブを使った多次元ログデータ分析では，数値の系列だけでなく，多次元的なアイテムセットなどの新しい情報の系列が生成される．このような系列から有用な部分を検出することが本研究の目的である．上記の動機にたって，本稿では，IBM から最近提案された多構造データベース (Multi-Structural Databases, MSDB) [4], [5] と呼ばれるデータキューブの考え方をアイテムセットキューブに適用する．MSDB は，与えられた評価関数の下でデータキューブから「意味のある」部分領域を求める体系である．本稿では，1-アイテムセット，及び，2-アイテムセットに基づいた計算機ログメッセージ系列の分析に MSDB を適用した結果を報告する．

以下，2 節で MSDB の概要を述べ，3 節で MSDB の演算とログ分析へ適用するために我々が補正した方法を述べる．4 節で計算機ログデータ分析への適用方法と結果を述べ，あわせて，

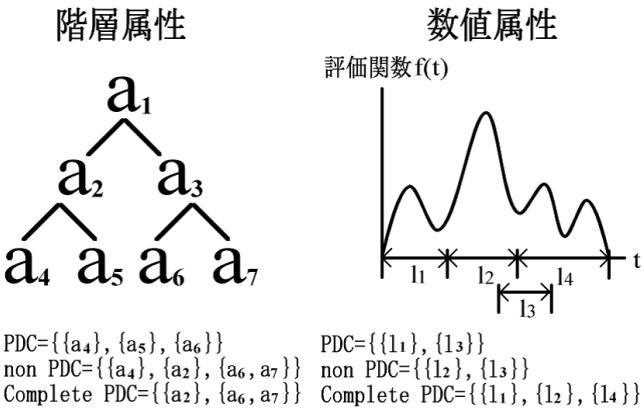

図 3　PDC の構造例

アイテムセット系列への適用方法を検討する．最後に 5 節で本稿をまとめる．

2. MSDB の概要

2.1 MSDB の考え方

MSDB は数値データキューブの一種であり，概念階層のある属性 (場所や話題など), 及び, 階層を持たない数値属性（時間軸など）からなる多次元データキューブである．データキューブの各キュボイド（最小立方格子）は，制約を満たすデータ数を格納する．文献 [5] では，Web データ集合を話題, 場所, 時間といった多次元分析空間内で自動分類する枠組として MSDB を提案している．

MSDB の典型的な問い合わせは,「今年，Web 空間上でどのような話題がどの地域で偏って人気があったか．その＜話題, 地域＞の組のうちできるだけまとまりのあって重なりのないものを k 個，与えた概念階層上で求めよ」というものである．実現方法は，対応する多次元データキューブに対して，適切な制約条件下で，PDC(または，問い合わせの種類によっては完全 PDC) と呼ぶ最適な立方格子分割を計算する．

ここで，PDC(Pairwise Disjoint Collection) とは，要素の集合 S が与えられたとき，S の部分集合 l_i の集まり $\{l_1, l_2, ..., l_n\}$ であって，かつ，全ての i, j に対して $l_i \wedge l_j = \phi\ (i \neq j)$ となるものである．典型的には，適当な評価関数の下で，データキューブの各属性ごとに最適な PDC を計算する．例として，図 3 左に，概念階層のある属性 1 つについての PDC を示す．また，同図右に，数値属性 1 つの場合の PDC を示す (この図は，時間軸 t にそって関数 $f(t)$ が与えられた場合に適当な t 軸上の PDC を求める場合である). PDC が元の集合 S の直和になっているとき，その PDC を完全 PDC (Complete PDC) とよぶ．

MSDB は，上記の枠組に沿って，与えられた評価関数を最適にするような PDC を見つける技法である．原論文 [4] [5] では，評価関数を適当に与えて，

＊「どの PDC がもっともよく均等に多次元空間を分割するか」(Divide 演算),

＊「データ集合 A と B を与えたとき，どの PDC がもっともよく A と B を識別する分割になっているか」 (Differentiate 演算),

*「適当な尺度 M を決めたとき，どの PDC が M の下でもっとも良く次元 X について目立つ分割になっているか.」(Discover 演算),

を提案している．

2.2 準備: 数値属性の場合の PDC 計算アルゴリズム

ここで，図 3 右のような数値属性 1 つの場合の PDC 計算アルゴリズムとして，k-最大部分区間問題を述べておく．

今，数値データ列を $x_1, ..., x_n$ とする．添字 i は，数値属性を単位要素の系列とみなしたときの i 番目の要素であり，x_i はこの要素が持つ評価関数の値である．(例えば，x_i は，i 番目の時刻に生じたエラー回数)．以下，この系列を変換し，奇数項は正の値，偶数項は負の値を持つと仮定する．変換は，元の系列で同じ符号同士が隣り合っているときにそれらを足すだけである．変換後の系列も $x_1, ..., x_n$ と表記する．

数値属性は添字の列 $\{1, 2, \ldots, n\}$ で与えられている．今，この列の中から，1 つの連続した部分区間 l_i を取り出したとき，l_i に属す要素 x_p $(p \in l_i)$ の値の合計値を $agg(l_i)$ とする．

k-最大部分区間問題とは，数値属性が持つ区間 $[1, n]$ の中から k 個の（重ならない）部分区間 $l_1, l_2, \ldots, l_k$ を取り出すとき，$agg(l_i)$ の和が最大になるよう $\{l_i\}$ を選ぶ問題である．すなわち，$X = \{x_1, x_2, ..., x_n\}$ のデータに対して，$PDC = \{l_1, l_2, ...l_k\}$ を選ぶとき，

$$\sum_{i=1}^{k} agg(l_i) \rightarrow \max$$

としたい．

$P([1, i], k)$ を，区間 $[1, i]$（つまり，要素列 $x_1, ..., x_i$）から最適な k 個の部分区間を見つける場合の上の式の値とする．$P([1, i], k)$ は下記によって計算される:

$$P([1,i],k) = \max_j \{P([1,j-1],k-1) + P([j,i],1)\}, (k \leqq j \leqq i).$$

$P([1, n], k)$ は動的計画法により $O(n^2 k)$ で計算できる．

例：系列 {5, -1, 2, -7, 3} から $k = 2$ で計算すると，区間 [1,3]（つまり，要素列 {5,-1,2}）と区間 [5,5](つまり要素列 {3}）が選ばれる．

3. MSDB の演算

本節では，MSDB の演算のうち，4 節で用いる Discover 演算を説明する．その後，ログデータ分析への適用方法を，簡単な事例を使って述べる．以下，3.1 で数値属性用の演算定義を原論文 [5] に従って述べる．その後，3.2 でログデータに適用するために我々がとった手法を示す．最後に，3.3 で，階層属性 1 個の場合の PDC 計算アルゴリズムとして，原著では不明確なものを我々が補正した算法を述べる．

3.1 Discover 演算 (数値属性)

Discover は，調べたいデータ集合 X の中で M と言う尺度次元から見て目立っている領域を k 個見つける問い合わせである．今，X を，データ集合 X_i $(i = 1, 2, \ldots, n)$ の和集合とする．例えば，X_i は時刻 i において観測されたメッセージの集合を表す．以下，X_i を単位として考え，M から見て「目立つ」ような i 上の連続区間を k 個発見したい．

今，X_i の添字 i の区間 $[1, n]$ を対象に，その部分区間を l とおく．このとき，l に属すデータ $X|l$ が M について目立つ度合いを表す評価関数 η を次式 $f_D(X|l)$ で与える：

$$f_D(X|l) = \frac{\sum_{x \in X|_l, y \in X \setminus (X|_l)} d_M(x,y)}{\sharp(X|_l)\sharp(X \setminus (X|_l))} - \gamma \frac{\sum_{x,y \in X|_l} d_M(x,y)}{\sharp(X|_l)^2}.$$

ここで，$d_M(x, y)$ は要素 x と y の次元 M における違いを表す尺度関数である．ただし，要素 x や y は，X_i に含まれる原子的なデータであり，上の例では個々のメッセージに相当する．

上式の前項を Separation，後項を Cohesion という．Separation は，$X|l$ に含まれる任意の 1 要素 x と $(X - X|l)$ に含まれる任意の 1 要素 y からなる対 (x, y) あたりの $d_M(x, y)$ の平均値を表す．Cohesion は $(X|l)$ 内の $d_M()$ の群内平均値である．γ の値は 1〜2 が良いとされる．

以上の準備の下で，Discover という問い合わせは，X の全区間 $[1, n]$ を PDC $\{l_1, l_2, ..., l_k\}$ に分けたとき，

$$\sum_{i=1}^{k} f_D(X \mid l_i) \rightarrow \max$$

とすることである．ただし，原論文では明記されていないが，実際には，最初に与えた各 X_i について $\eta = f_D(X_i)$ を計算しておき，部分区間 l の評価値 $f_D(X|l)$ は，l に含まれる X_i の持つ η 値の和で与える．解の計算は 2.2 の算法が用いられる．以上が原論文 [5] で述べられている技法である．この文献では Web データの自動分類に Discover を適用しており，X_i はある話題に分類される Web データの集合である．

3.2 ログデータ分析への適用方法

次に，例を使って，数値属性上の Discover 演算をログ分析に適用する場合を説明する．図 4 のデータを用意した．図は，集合 X_i $(i = 1, \ldots, 7)$ が，属性 Y 上の整数値 {1, 2, 3, ..., 10} のうちどの値をいくつ要素として持つか，を示している．X_i が時刻 i におけるメッセージ集合，Y の値 (整数) がメッセージ ID と見なせばよい．

図中，X_i を示す長方形に $a \times b$ と表記してあるとき，X_i は Y 軸上の幅 b 個の区間に入る Y の値 (整数値) を，a 個ずつ持っていることを示す．例えば，X_5 は，Y の値 1 から 10 までを各々20 個ずつ持つ集合である．(つまり，X_5 は，時刻 $t = 5$ において，メッセージ ID ＝ 1 から 10 までが各々20 個生成された，という意味になる)．X_2 は，Y の値 1 から 4 までを各々50 個ずつ持つ集合，である．

今，X_i を添字 i の順に並べて区間 $[1, n]$ を考える．これを，時間を表す属性 t とおいて，t 軸の中から Y 軸上で偏りのある値をとっている部分区間を $k = 2$ 個求めたい，とする．

ここで，X_i は，メッセージ ID $= y_{i1}, y_{i2}, ...$ を各々$n_{i1}, n_{i2}, ...$ 個ずつ持っている．今，メッセージ ID が x, y としたとき，$d_M(x, y) = |x - y|$ とする．(つまり，$x = 3, y = 5$ なら $d_M(x, y) = |3 - 5| = 2$). 次に，η の第一項となる Separation, つまり，X_i と $(X - X_i)$ との間の要素対 1 つあたりの平均距離は，メッセージ x, y を $x \in X_i$ と $y \in (X - X_i)$ としたとき，対 (x, y) の出現回数 $(n_x \times n_y)$ の全メッセージ対についての合計を分母,

特定のメッセージ対 (x_0, y_0) の出現回数 $occurrence(x_0, y_0)$ を係数とした値 $occurrence(x_0, y_0) \times d_M(x_0, y_0)$ の全メッセージ対についての合計を分子として求めた．η の第 2 項，Cohesion も同じ考えで計算する．

以上の定義の下で，t 軸に沿って各 X_i の Separation, Cohesion, η の値を求めたものが図 5 である ($\gamma = 2$)．Discover 演算 ($k = 2$) は，この図で t 軸に沿って $k = 2$ で k-最大部分区間問題を実行すれば良い．

図 5 上で t 軸について k-最大部分区間問題を計算すると，評価式を最大とする部分区間は，区間 $[2, 3]$(=X_2, X_3) と 区間 [6,7](=X_6, X_7) となる．従って，図 4 で言うと，偏ったメッセージを生成している 2 つの最大連続時区間は [2,3] と [6,7]，という結果になる．

このように，数値属性上の Discover 演算は，与えたメッセージ階層属性において偏ったメッセージ群を生成させているような連続時区間を k 個求めたいときに用いることができる．無論，メッセージ ID 間の距離は単純な ID の差では表せないので，概念階層下の適当な尺度関数を用いる必要がある．

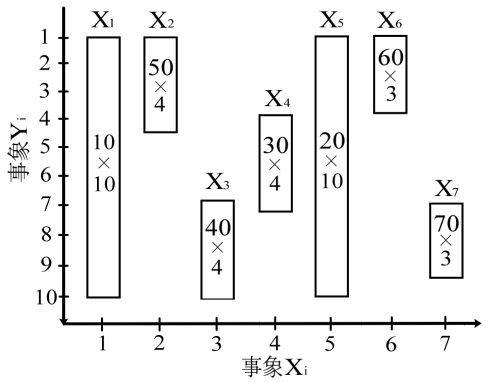

図 4　Discover 数値属性用元データ

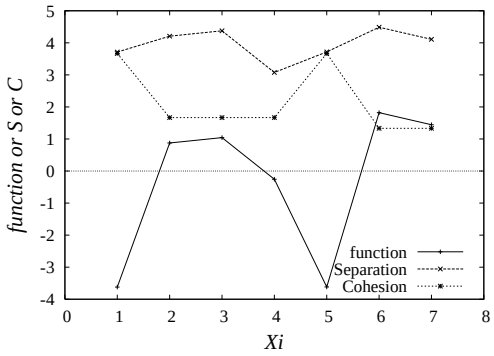

図 5　Discover 数値属性の解 ($\gamma = 2$)

3.3　階層属性用の Discover

階層属性における Discover の漸化式は数値属性用のものとは異なる． 原文献 [5] では記述が簡略化されており，例外的な状況が扱えない．以下，我々が補正して 4 節で用いる計算式を述べる．

今，概念階層が 2 分木構造で与えられたとする．ある親ノードを a，a の左側の子ノードを a_l，右側の子ノードを a_r とする (図 6 左). 各ノード a は，a を根とする部分木 l_i に該当する全データ $X|l_i$ について前節と同じ評価値 $\eta = f_D(X|l_i)$ を与えられている．この値も a と表記する．図 6 はそのような木の例である．

今，ノード a を根とした木から，大きさ k の PDC $\{l_1, l_2, \ldots, l_k\}$ を

$$\sum_{i=1}^{k} f_D(X \mid l_i) \to \max,$$

となるように選びたい．このときの最大値を求める漸化式 BestSplit($BS(k, a)$) は下記になる：

$(a \neq leaf)$ の場合，
　もし $k = 1$ なら，
　　$BS(k, a) = \max\{a, BS(1, a_l), BS(1, a_r), 0\}$
　もし $k \geqq 2$ なら，
　　$BS(k, a) = \max_{k'=0}^{k}\{BS(k', a_l) + BS(k - k', a_r), a, 0\}$
　それ以外の場合 $(k = 0)$ なら，
　　$BS(k, a) = 0$
$(a = leaf)$ の場合，
　もし $k \geqq 1$ なら，
　　$BS(k, a) = \max\{0, a\}$
　それ以外の場合 $(k = 0)$ なら，
　　$BS(k, a) = 0$

となる．

上式では，k 分割する時，各ノードにおける最適値がマイナスの値となってしまう場合は，その値を 0 として分割を無視している．また，$a \neq leaf$ かつ $k \geqq 2$ において，$k \geqq 2$ における解より，親ノード a 自体の値の方が大きい場合は，$k = 1$ として a を返す．この 2 点が我々が補正した部分である．

$k = 2$ における簡単な事例を図 6 右に示す．計算は上の式に沿った動的計画法で行われ，木のノード数を n，与えられた分割数を k として $O(k^2 n)$ で計算できる．一般には，概念階層は二分木ではない．この場合は，親ノード a を根ノードとおき，a の元々の子ノード m 個を葉ノードとするように二分木を挿入していく．追加した一時的ノードのコストは Differentiate, Discover の演算では $-\infty$ とする．

階層属性上の PDC 計算は，例えば，観察期間のうち偏った時間帯に生成されているメッセージ階層を求めたいときに用いることができる．

4.　計算機ログデータへの適用

今回，著者の一人が使用している個人用ノート PC のアプリケーションログに対して階層属性に変換し，Discover をかけてみた．なお，本節で述べるシステムは，アプリケーションログ系列を MS ACCESS 上に格納し，SQL プログラムによるデータ変形処理と Visual C による Discover 演算，演算結果を受けての元のイベントログデータ列との結合処理，が記述・実行できる実装になっている．

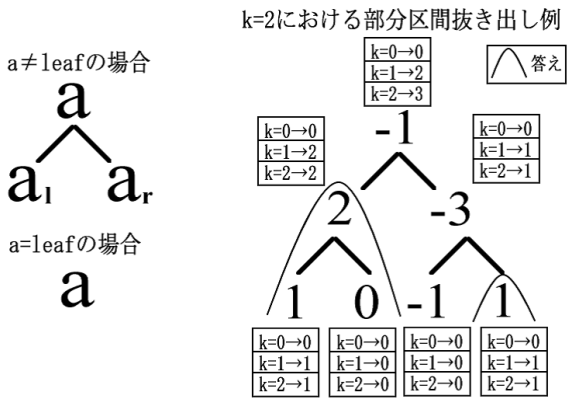

図 6　Discover 階層属性

4.1　ログデータの説明

2006/9/20〜2006/11/20 までの 62 日間に WindowsXP で発生したアプリケーションイベントログ 3112 件である．概念階層木は，全体を根ノードとし，イベントのソース名（全 24 種類）を子ノードとして，その下にイベント ID（全 47 種類）を子ノードとして作成した（図 7）．表 1 にその内訳を示す.（表 1 の内部 ID は，木構造の結果を表示する為に内部的に割り当てた番号である.）また，図 8 は発生したイベント数を 3 次元にグラフ化したものである．x 軸，y 軸をそれぞれ日付とイベントログの＜ソース名＋イベント ID ＞とし，z 軸をイベントが発生した件数を示している．

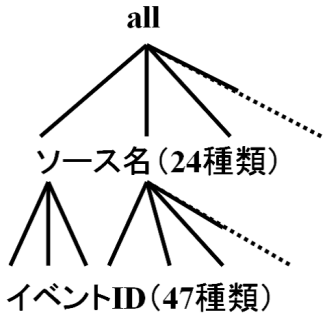

図 7　アプリケーションログ概念階層木

4.2　メッセージログの発生回数

Discover 演算をかける前に，メッセージログの発生件数を 1 時間間隔の単位区分に分け，これに対して k-最大部分区間問題をかけた（$k = 5$）．具体的な問い合わせ内容は以下のように記述する．

Q1:「メッセージログ発生回数の中で通常より多く発生している時間帯は？」.

その結果が図 9 である．これはメッセージログ発生回数の中で通常より多く発生している時間帯を調べる問い合わせであり，図 9 を見る限り，目視で確認できる内容も自動的に判別可能であることが分かる．

4.3　メッセージ階層を尺度次元にした場合

次に，3.2 の方式に従ってメッセージの階層木を尺度次元 (M) とし，メッセージ階層木の中で偏ったメッセージ集合を生成しているような k 個の時間帯を求めた．具体的な問い合わせ内容は以下のように記述する．

Q2:「メッセージの階層木を尺度次元 M とし，メッセージ階層木の中で偏ったメッセージ集合を生成しているような時間帯は？」.

表 1　アプリケーションイベントの内訳

内部 ID	イベント ID	ソース名 (内部 ID)
0	1000	ApplicationError
1	1002	ApplicationHang
2	1	ccEvtMgr(102)
3	26	ccEvtMgr(102)
4	27	ccEvtMgr(102)
5	1	ccSetMgr(103)
6	26	ccSetMgr(103)
7	2	crypt32(104)
8	7	crypt32(104)
9	100	ESENT(105)
10	101	ESENT(105)
11	102	ESENT(105)
12	103	ESENT(105)
13	300	ESENT(105)
14	301	ESENT(105)
15	302	ESENT(105)
16	0	EvtEng
17	1904	HHCTRL
18	251	InterBaseGuardian
19	1000	Kraidsvc(110)
20	1001	Kraidsvc(110)
21	1000	LoadPerf(111)
22	1001	LoadPerf(111)
23	4100	MDM
24	2444	MSDTC
25	1005	MsiInstaller(113)
26	1015	MsiInstaller(113)
27	1022	MsiInstaller(113)
28	1025	MsiInstaller(113)
29	11707	MsiInstaller(113)
30	11724	MsiInstaller(113)
31	11728	MsiInstaller(113)
32	1	NPFMntor(114)
33	26	NPFMntor(114)
34	1	nview_info
35	0	RegSrvc
36	1800	SecurityCenter(117)
37	1801	SecurityCenter(117)
38	1	SNDSrvc(118)
39	26	SNDSrvc(118)
40	0	SPBBCSvc
41	1517	Userenv(120)
42	1524	Userenv(120)
43	100	VisualStudioAnalyzerRPCbridge
44	1001	Winlogon(122)
45	1002	Winlogon(122)
46	5603	WinMgmt

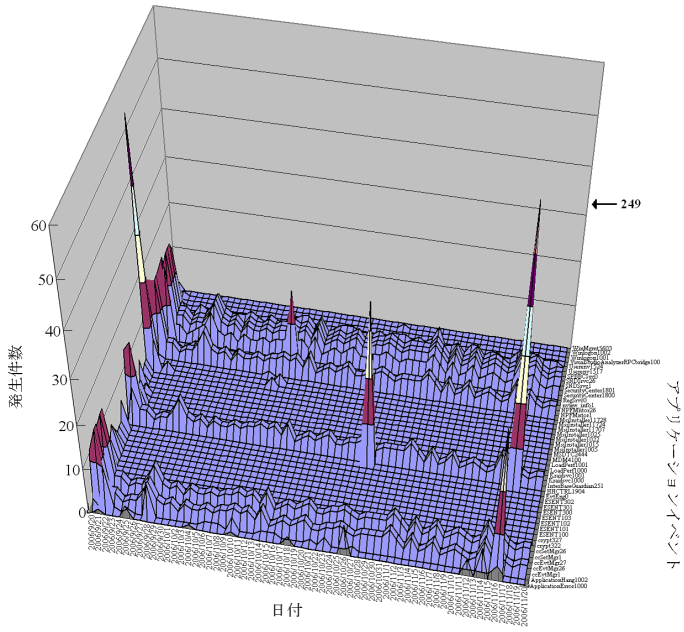

図 8　アプリケーションログ発生件数

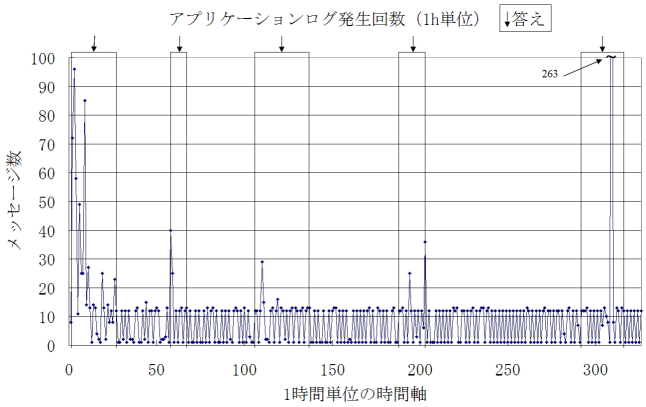

図 9　イベント発生件数（1 時間間隔）と Q1 の結果

ここで使用する Discover の評価関数内の $d_M(x, y)$ は，概念階層の木構造におけるメッセージ x と y 間の hop 数である．時間軸は連続しているので数値属性として計算している．その結果が図 10（1 時間間隔，分割数 $k = 5$）である．図 10 において縦線で囲まれている領域が今回見つかった範囲である．

図 10 では，1 時間間隔として計算した中で目立っているとして選ばれた $k = 5$ 区間は，

2006/9/28 1:00〜2006/9/28 4:00

2006/10/14 6:00〜2006/10/16 7:00

2006/10/18 15:00〜2006/10/18 19:00

2006/10/26 16:00〜2006/10/27 4:00

2006/11/16 8:00〜2006/11/16 18:00

である．図 9 と見比べると，Q2 で異常と判断された区間は Q1 と重なることもあるが，図 9 からでは分からない日付の範囲もいることも分かる．

図 10 において赤点線で囲まれた部分の実際のログデータを示したものが，図 11 である．図中の A では，η の値が高い範囲を選択しており，つまり，選ばなかった範囲である B とは

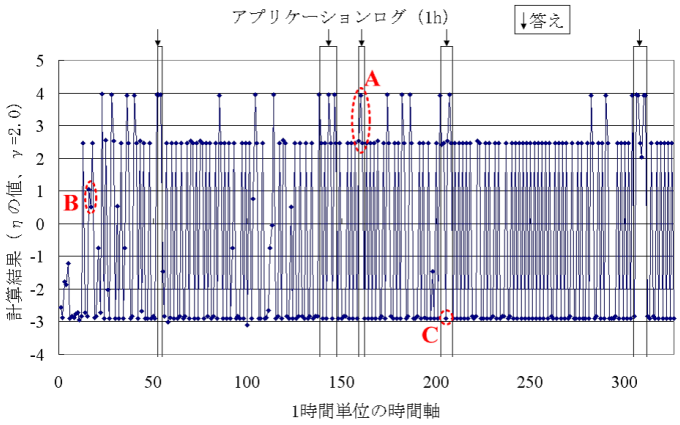

図 10　1 時間間隔の結果（Q2 の結果）

違い，珍しいメッセージが多く発生した時区間を検出していることが分かる．また，η の値が低い C のメッセージ列を調べると，Windows の起動メッセージ列であった．つまり，よく発生するメッセージを捉えていることが分かる．

A

日付	ソース	num
2006/10/18/15	nview_info1	1
2006/10/18/18	ApplicationError1000	2
2006/10/18/19	Userenv1517	1

B

日付	ソース	num
2006/09/21/18	MsiInstaller11707	3
2006/09/21/18	MsiInstaller11728	1
2006/09/21/19	SecurityCenter1800	1
2006/09/21/19	SecurityCenter1801	1

C

日付	ソース	num
2006/10/26/23	ccEvtMgr1	1
2006/10/26/23	ccEvtMgr26	1
2006/10/26/23	ccSetMgr1	1
2006/10/26/23	ccSetMgr26	1
2006/10/26/23	InterBaseGuardian251	1
2006/10/26/23	Kraidsvc1000	1
2006/10/26/23	NPFMntor1	1
2006/10/26/23	NPFMntor26	1
2006/10/26/23	SecurityCenter1800	1
2006/10/26/23	SNDSrvc1	1
2006/10/26/23	SNDSrvc26	1
2006/10/26/23	SPBBCSvc0	1

図 11　図 10 におけるログデータの中身

図 10 の η の値に応じたログ系列の内訳を調べてみた．

$\eta = 4$ では，ApplicationError(1000)，ApplicationHang(1002)，と言ったイベントが単体で出現．プログラミングの実行時によるエラーであることが分かる．

$\eta = 2.5$ では，nview_info(1) か Userenv(1517) と言うイベントが単体で出現．主に前者は Oracle のクライアントが Back ground で動作した時，後者は Windows の終了時に発生している．

$\eta = 1 \sim -1$ では，nview_info(1)，ApplicationError(1000)，ApplicationHang(1002)，Userenv(1517)，MsiInstaller(11707,11728)，SecurityCenter(1800,1801) のイベントのうち 2〜3 個セットとなった列が出現．η が高いときに発生しているイベント以外にインストールや Update 関係のイベントが発生している．

$\eta = -3$ では，ccEvtMgr(1,26)，ccSetMgr(1,26)，InterBase-

Guardian(251)，Kraidsvc(1000)，NPFMntor(1,26)，SecurityCenter(1800)，SNDSrvc(1,26)，SPBBCSvc(0) の列が発生している．これは Windows の起動時に毎回ロードされているイベントである．このように η の値によってログ系列の内容が分かれている点は良かったと言える．

従って，選ばれた 5 区間は ApplicationError(1000)，ApplicationHang(1002) が集中して作業をしている時区間であったと言える．

4.4 時間軸を尺度次元にした場合

次に日付を尺度次元 (M) として，メッセージの概念階層上で特定の時間帯に偏って生じている階層を見つけるために Discover を実行した．具体的な問い合わせ内容は以下のように記述する．Q3:「日付を尺度次元 M とし，メッセージの概念階層上で特定の時間帯に偏って生じているメッセージ階層は？」．

ログは 1 日単位のレコードに変換している．ここで使用する Discover の評価関数内の $d_M(x, y)$ は，メッセージ x と y における日付の差で計算している．これに Discover をかけて，分割数 $k = 10$ とし，階層属性として 2006/9/20〜2006/11/20 の間で計算した ($\gamma = 1.5$)．その結果が図 12 である．図 12 中の木構造のノードは { 内部 ID:η の値 } として表記しており，山で囲まれた部分が Discover によって見つかった領域である．η の値の値が高いイベント ID だけで構成されているソース名は，結果としてその山をまとめて捕らえているのが分かる．

この結果より，アプリケーションログの全体の中において目立っている領域を調べると，各アプリケーションメッセージの詳細は，

1．Windows の Update 情報（内部 ID…104）
2．データベースエンジンに関する情報（内部 ID…105）
3．TOSHIBA RAID サービスの停止（内部 ID…20）
4．Adobe Reader のインストール情報（内部 ID…25〜28）
5．Windows のワイヤレス LAN に関する情報（内部 ID…16,35）
6．Visual Studio SP2 サービスの情報（内部 ID…43）

であることが分かった．これらのメッセージを図 8 上に対応させると，主に PC の使用を開始した 2006/9/20〜2006/9/21 付近に発生したアプリケーションイベントを捉えていた．

4.5 1-item,2-itemset 混合列への適用

4.5.1 ログデータの変形方法

次に，メッセージ系列において単位時間あたりに生じた目立つ 1-item (個々のメッセージ ID) と 2-itemset(一定頻度以上の確率で同時に起きたメッセージ 2 つの組合せ) の集まりを 1 レコードとおいて，そのレコードの時系列に Discover を適用する．変形の手順は次の通り：

(i) 5 分間に発生した各アイテム (＝メッセージ ID) について組み [アイテム ID, 発生数] をつくり，そこから，5 分間に生じた 2-アイテムセットとして [アイテム 1 &アイテム 2, 発生数] をつくる. (2-アイテムセットの発生数は元の 1-アイテムの発生数の積). 次に，このレコードを 1 時間単位で 2-アイテムセットごとに集計し，1 時間単位の 2-アイテムセット (と発生数) のレコード列をつくる．

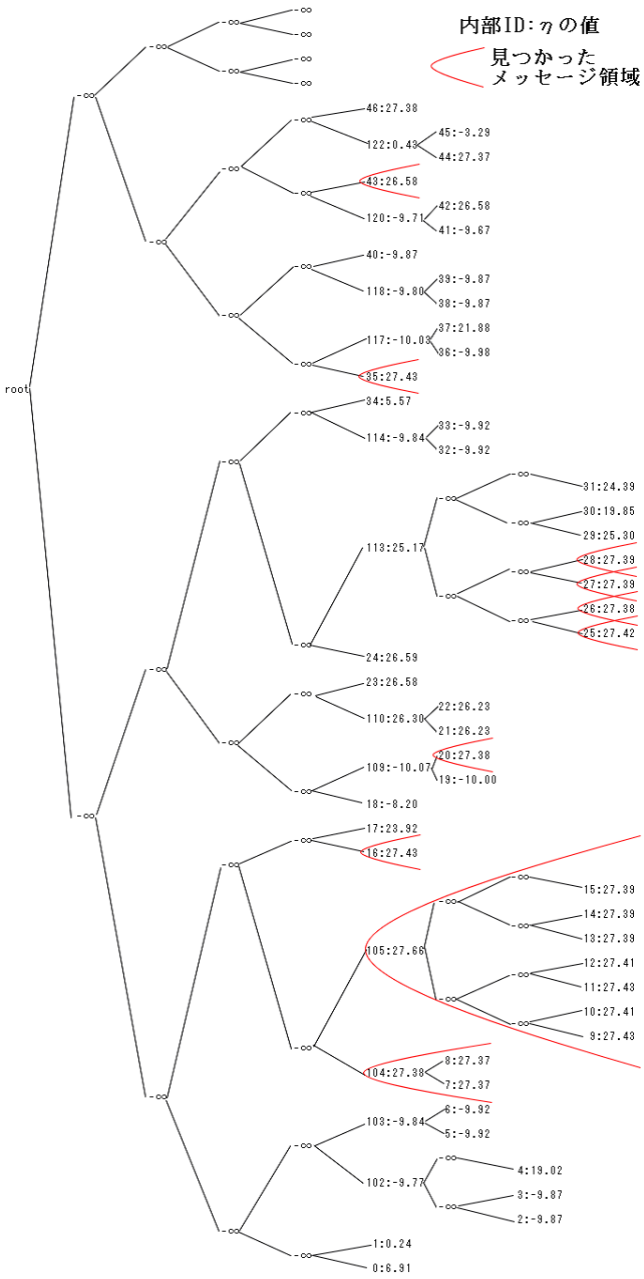

図 12　アプリケーションログ木構造 (Q3 の結果)

(ii) 全時区間にわたって 2-アイテムセット全ての発生数の総和を求め，その数に対して全時区間に渡って 0.06%以上発生した 2-アイテムセットのみを (i) に残す．(今回は, 13646 回中, 9 回以上).

(iii) 4.3 節の実験で，Q2 の Discover 演算を行った結果が $\eta = 2.0$ 以上となる部分列に含まれる 1-アイテム (11 種類) を選ぶ．4.3 節で用いた 1 時間単位の 1-アイテム (と発生数) のレコード列から，この 11 種類のアイテムに関するレコードを抜きだし，(i) の結果に加える．

以上の変形処理は Access 上の SQL プログラムで作成した．

用いる概念階層木を図 13 に示す．この図は，4.3 節の Q2 の実験結果 (図 10) に基づき，メッセージ系列の中身を，ブートの場合，アプリケーション関連エラーの場合，などの事象に応じて分類したものに相当する．図中の η は 4.3 節の Q2 で分類し

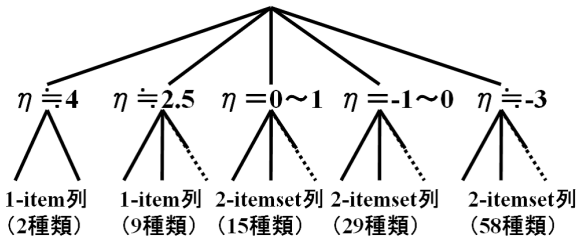

図 13　1-item,2-itemset 混合列概念階層木

たとき（図 10）の η である．階層木で $\eta \geqq 2$ となる中間ノードの下には，図 10 で $\eta \geqq 2$ となったメッセージ列に現われる 1-アイテムを配置した．階層木の $\eta < 2$ となる中間ノードの下には，図 10 で該当する η 値をとるメッセージ列に出てくる 2-アイテムセットを配置した．

図 13 の $\eta \simeq 4$ の葉ノードに追加した 1-item は ApplicationHang1002,ApplicationError1000 であり，$\eta \simeq 2.5$ の葉ノードに追加した 1-item 列は MsiInstaller1025,MSDTC2444,MsiInstaller11724, nview_info1,MsiInstaller11707,HHCTRL1904,InterBaseGuardian251, Userenv1517,SecurityCenter1800 である．2-itemset 列と混合にするので，組み合わせのもう片方には”Nodata”を入れている．以降，このログデータに対して Discover 演算をかけた結果を示す．

4.5.2　メッセージ階層を尺度次元にした場合

まず，メッセージの階層木を尺度次元 (M) とし，メッセージ階層木の中で偏ったメッセージ集合を生成しているような k 個の時間帯を問い合わせ Q4 として求めた ($k = 5$)．図 14 において縦線で囲まれている領域が今回見つかった範囲である．

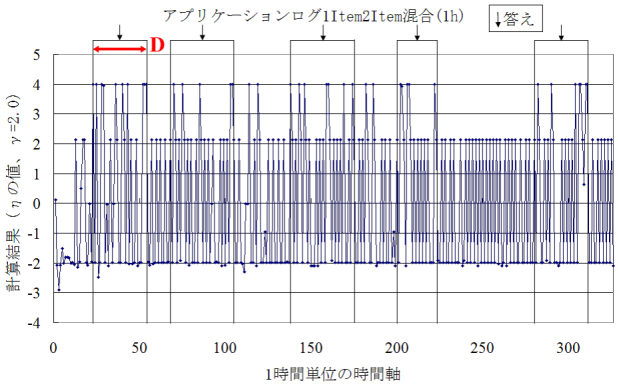

図 14　1-item,2-itemset 混合列の結果（Q4 の結果）

図 14 の中で目立っているとして選ばれた $k = 5$ 区間は，

2006/9/22 7:00～2006/9/28 4:00

2006/9/30 16:00～2006/10/8 3:00

2006/10/14 6:00～2006/10/23 18:00

2006/10/26 13:00～2006/10/30 10:00

2006/11/11 18:00～2006/11/16 18:00

である．図 10 と見比べると，特に最初の 2 区間が異なった範囲を取っていることが分かる．そこで，図 10 では取らなかった範囲のログデータを調べてみた．

図 14 において矢印で示した部分 D の実際のログデータを示したのが，図 15 である．図中 D の下線部のメッセージ列は Q2 において $\eta = 4$ 付近では見られなかったメッセージを新たに確認することが出来ている．

日付	Item	num
2006/09/22/07	ApplicationHang1002,NoData	8
2006/09/22/09	ccEvtMgr1,ccEvtMgr26	1
2006/09/22/09	ccEvtMgr1,ccSetMgr1	1
：		
2006/09/22/12	nview_info1,NoData	8
2006/09/22/13	ApplicationHang1002,NoData	1
2006/09/22/13	ApplicationHang1002,nview_info1	9
：		
2006/09/23/03	MsiInstaller11707,NoData	1
：		
2006/09/28/01	ApplicationHang1002,NoData	1
2006/09/28/02	ApplicationHang1002,NoData	2
2006/09/28/04	ApplicationHang1002,NoData	2

D

図 15　図 14 におけるログデータの中身

5.　まとめと現在の課題

本稿では，ログデータ系列の分析に多構造データベース (MSDB) 演算を適用する方法とその試行結果を述べた．

本稿では，まず，MSDB の概要を説明し，次に，単位時間あたり複数のメッセージが生成される状況の下で MSDB 演算を適用する方法を述べた．その後，Windows XP のアプリケーションイベントログ系列に MSDB の Discover 演算を適用した．その結果，与えたメッセージ階層の中で偏った部分階層のメッセージ列を生成している時間帯を自動的に識別することができた．特に，Discover 演算を適用する評価指標である η の値に応じて，意味のある特定のイベントログ系列が生成されていることを確認した．一方，特定の時間帯に偏って生じるメッセージの部分階層を求める場合には，数値集計データから人目で判定できる内容とほぼ同じ結果に限られた．この点は，メッセージ階層をより複雑にする，などを試す必要がある．

また，長さ 1, 2 のアイテムセットの混合系列に対しても Discover 演算を適用する方法も試行し，メッセージ系列の内容に応じて有意な時区間の判定ができることを示した．

文　　献

[1] 成瀬 正英, 大森 匡, 星 守,“ 多次元的なログデータマイニングを実現するデータキューブ機構の提案と評価 ”, DEWS2005 3C-i10, 電子情報通信学会 2005.

[2] 大森 匡, 李 光浩, 七海 嘉仁, 星 守, “ N-OPS:系列データベースにおける連続系列パターンの探索算法 ”, 電子情報通信学会和論文誌, J89-D (No.7), pp.1465-1480, 2006 年 7 月.

[3] 山平 耕作, 小寺 考, 土田 正士“ SQL スーパーテキスト ”, 技術評論社.

[4] R.Fagin Ph.Kolaitis R.Kumar J.Novak D.Sivakumar A.Tomkins, “ Efficient Implementation of Large-Scale Multi-Structural Databases ”, Proceedings of the 31st VLDB Conference, Trondheim, Norway, pages 958-969 2005.

[5] Ronald Fagin R.Guha Ravi Kumar Jasmine Novak D.Sivakumar Andrew Tomkins, “ Multi-Structural Databases ”, 24th ACM Symposium on Principles of Database Systems , pages 184-195 2005.

[6] 日本電気株式会社 , “ 統合システム運用管理 WebSAM ”, http://www.sw.nec.co.jp/middle/WebSAM/.

[7] L.Harada, Y.Hotta, N.Akaboshi, K.Kubota, T.Ohmori, R.Take, “ Event Analyzer: A Tool for Sequential Data Processing ”, ACM CIKM 2003, pp.172-174, 2003.