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

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

涌波 信弥[1]　大森 匡[2]　星 守[3]

Shin-ya WAKUNAMI　Tadashi OHMORI
Mamoru HOSHI

計算機システムのログデータ系列や Web サイトのアクセスログから有意な情報を取り出したいという要求は大きい．このために，著者らは，アイテムセットキューブと呼ぶ多次元的なログデータ分析機構を提案している．本稿では，Fagin らの多構造データベース (MSDB) という情報抽出用データキューブモデルを本機構のログデータ分析出力に適用することを試み，その方法とログ分析の効果を報告する．

A today's hot problem is to find meaningful information from massive amounts of log-data sequences. The authors previously proposed a multi-dimensional data-mining tool named *itemset cube* for this purpose. This paper applies an idea of another data cube Multi-Structural Data Base (MSDB) to the outputs of an itemset cube, and describes its ability to find meaningful log-sequences.

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 サイトのアクセスログを分析して有用な情報を取り出したいと言う要求は依然として大きい [2][3]．ログデータ系列の分析とは，一般に，ログデータの系列を数値の時系列 (例えば，単位時間あたりのメッセージ数 vs. 時刻) やメッセージの時系列 (単位時間あたり生じたメッセージ集合 vs. 時刻)，あるいは，同時に発生したメッセージの組合せや相関の時系列，などへ変換し，そこから有用な情報を取りだすことである [3]. 有用な情報とは，例えば，「メッセージ数やメッセージの内容，あるいはメッセージの組合せが通常のメッセージ生成列から比べて特徴的な時間帯はどれか」という問いがある．これら多様なログデータ分析作業を支援する機構として，著者らは，文献 [1] で，データキューブモデルに基づいたログデータマイニング機構「アイテムセットキューブ」を提案してきた．ここで，アイテムセットは，｛メッセージ A，メッセージ B，…｝と言う同時に一定頻度以上起きた事象の組み合わせの系列（1-item 列…k-itemset 列）を表す．アイテムセットキューブは，データキューブモデルに沿った多次元制約条件の下で，上述した数値列やアイテムセット系列といった時系列を高速に計算する機構である．文献 [1] で，著者らは，Web サイトのアクセスログ系列の分析を本機構により行い，アイテムセット (つまり，1 ユーザあたりの頻出する Web アクセスのパターンを表す情報) の系列を多様な条件下で生成し，OLAP の枠組の中で，全体から見て特徴的な行動をとっているユーザグループと時期の組合せを識別した．しかし，アイテムセット系列の中で特徴的な情報を生じている部分系列の判定は自動的にできていない．一方，近年，多構造データベース (MSDB)[4] という技法が Fagin らにより提案され，データキューブから特徴的な部分立方体領域を検出する機構として有効とされている．そこで，本稿では，MSDB の考えをアイテムセットキューブの出力に適用し，数値列や長さ 1 または 2 のアイテムセット系列から「意味のある」部分領域を自動的に見つける方法とその試行結果を述べる．具体的には，概念階層を持った計算機ログメッセージの生成時系列を対象に，1-item 列や 2-itemset 列の内容が特徴的な時間帯を求める例を扱う．

以下，2 節で MSDB の概要を，3 節で MSDB の Discover 演算と計算機ログデータ系列への適用方法を述べ，適用結果を 4 節，まとめを 5 節で述べる．

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，概念階層を持つ属性 (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 [4]．データキューブのキュボイド（最小立方格子）は，制約を満たすデータ数を格納する．文献 [4] では，Web 上のニュース文献の集合を話題や地域の階層属性と時間軸 (数値属性) から成る多次元分析空間内で自動分類している．具体的には，「今年 1 年間のニュース文献集合のうち，特定の時間帯に偏って出現した話題 $Topic$ を k 個知りたい．$Topic$ の概念階層上で重なりのない部分階層のうち，この条件を満たす最適な k 個を選べ」とか，「話題 $Topic$ の概念階層のうち，どこか特定の部分階層の話題が集中して出現しているような時間帯を k 個知りたい．この条件を満たす時区間のうち，重なりのない k 個として最適なものを求めよ」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の下で PDC(Pairwise Disjoint Collection) と呼ぶ最適な立方格子分割を計算する．PDC 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 $\{l_1, l_2, ..., l_k\}$ 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．(PDC が元の集合 S の直和なら，完全 PDC(Complete PDC) とよぶ).

例として，図 1(a) に，概念階層のある属性 1 つについて，$k = 3$ のときの PDC を示す．図中，$a1$ が根ノード (つまり，全体 (universe))，$a2, a3$ が $a1$ の子ノード，$a4, a5$(または $a6, a7$) が $a2$(または $a3$) の子ノードである．$\{\{a4\}, \{a6\}, \{a7\}\}$ は，互いに疎な要素集合 3 つから成るから PDC である．$\{\{a4\}, \{a2\}, \{a6\}\}$ は，$a2$ が $a4$ の上位概念であるから排他的でなく，PDC ではない．同図 (b) は数値属性 1 つの場合の PDC である．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．

MSDB は，典型的には，適当な評価関数 f の下で PDC のサイズ k を与えたときに，データキューブの各属性ごとに大きさ k の最適な PDC を計算する技法である．具体的には，問い合わせクラスごとに適当な評価関数を与えて，「データ集合 A を最適に自動分類する PDC の発見 (Divide)」，「データ集合 A と B の違いを最大にする PDC の発見 (Differentiate)」，「データ集合 A について適当な尺度 M の下で次元 X に沿ってもっとも目立つような PDC を発見する演算 (Discover)」がある．本稿では，Discover 演算を用いる．

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 1(b) 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

[1]学生会員，電気通信大学，wakunami@hol.is.uec.ac.jp
[2]正会員，電気通信大学，omori@hol.is.uec.ac.jp
[3]非会員，電気通信大学

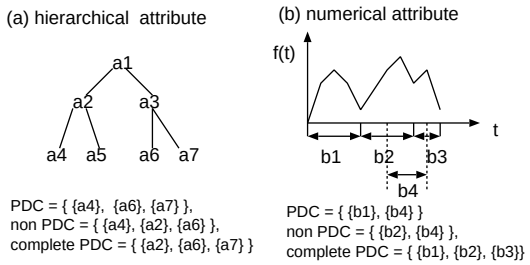


図 1: MSDB における PDC の例
Fig.1: examples of PDC

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:

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

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

例：系列 S={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．S の各要素 5 や-1 を, 単位時刻あたりのメッセージ集合が持つ「目立つ度合い」とすれば, これで, 目立つ度合いの和が最大になる 2 個の部分区間を検出したことになる．

3. MSDB の演算

3.1 Discover 演算 (数値属性)

Discover は，データ集合 X の中で M と言う尺度次元 (尺度が与えられた次元) から見て目立つ領域を 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：

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

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．(本稿では $\gamma = 2$ に固定した).

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．

3.2 ログデータ分析への適用方法

ログデータ系列分析の場合, 時刻 i のメッセージ集合 X_i は，メッセージ ID $= y_{i1}, y_{i2}, ...$ を各々 $n_{i1}, n_{i2}, ...$ 個持つ．今，メッセージ ID x, y について，$d_M(x, y)$ を, メッセージの概念階層上で適当に定義した x, y 間の「遠さ」を表す尺度とする．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値を分母，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値を分子として求めた．η の第 2 項，Cohesion の計算式も同様にして与える．

4. 計算機ログデータへの適用

4.1 ログデータの説明

MSDB の諸演算とアイテムセット系列の生成機能を MS ACCESS 上の SQL プログラムと Visual C で試作し, 実ログデータ系列へ Discover 演算を適用した. 対象としたログは, 2006/9/20〜2006/11/20 までの 62 日間に WindowsXP(本システム開発用の個人ノート PC1 台) で発生したアプリケーションイベントログ 3112 件である．ログレコード 1 つは, 時刻とソース名, イベント ID を持つ. 概念階層木は，全体 (universe) を根ノード，ログのソース名 (事象を生成した主体の種類を表す ID, 全 24 種類) をその子ノードとし, この子ノードの下に, 対応するイベント ID(全部で 47 種類) を葉ノードとして作成した．(各ログレコードは葉ノードに該当). 以下, ログの種類を, ソース名 (イベント ID) と表記し, Discover 計算用のログレコード x, y 間の尺度 $d_M(x, y)$ は，この木構造上の x と y 間の最短 hop 数とする．

4.2 メッセージログの発生回数

まず, メッセージログの発生件数を 1 時間単位の数値列に直し, これに k-最大部分区間問題を実行した $(k = 5)$. すなわち, 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を k 個求めよ.」という問い合わせである. 実行結果を図 2 に示す. 同図より, 目視で確認できる内容も自動的に判別可能である.

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 Discover 演算により求めた．すなわち, 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

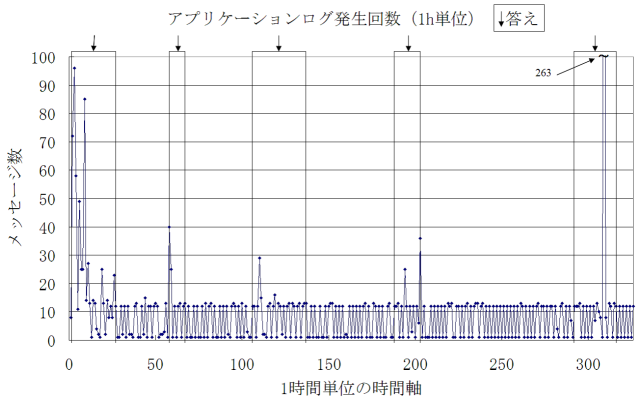


図 2: イベント発生件数（1 時間間隔）と Q1 の結果
Fig.2: Num. of events vs. time (hours)

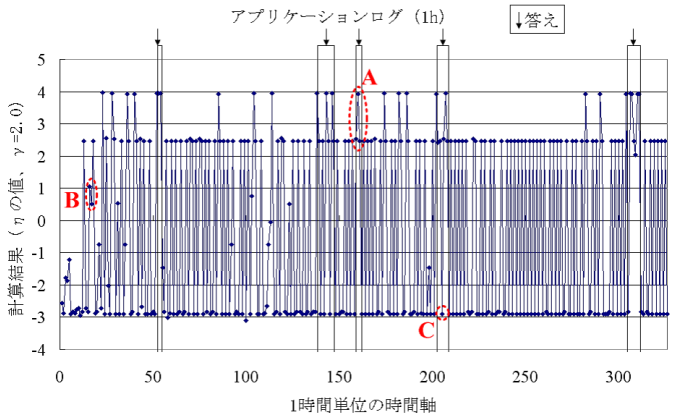


図 3: 1 時間間隔の η 値列と Q2 の結果
Fig.3: η vs. time (hours), with Q2's result

間帯を求めよ.」である. ログ x, y 間の尺度 $d_M(x, y)$ は 4.1 のそれに従い, 時間軸は数値属性として扱う. ($\gamma = 2$).

Q2 の実行結果を図 3（1 時間単位の η 値の系列，分割数 $k = 5$）に示す．図 3 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 3 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 2 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 2 からでは分からない日付の範囲もいることも分かる．

図 3 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 4 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 4: 図 3 におけるログデータの中身 (図中のソースはソース名＋イベント ID を示す)
Fig.4: logs of the periods A,B,C in Fig.3

図 3 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 Background で動作した時，後者は Windows の終了時に発生．
$\eta = 1 \sim -1$ 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)，InterBaseGuardian(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 1-item,2-itemset 混合列への適用

4.4.1 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．

(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).

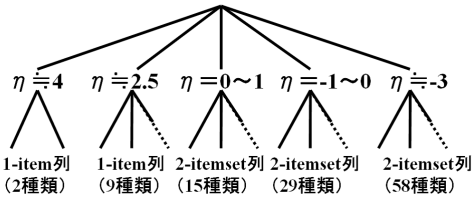
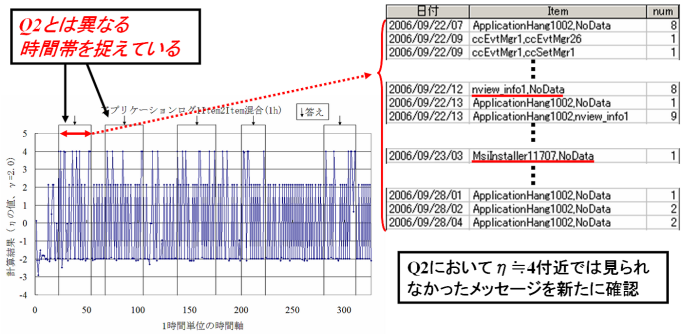


図 5: 1-item,2-itemset 混合列概念階層木
Fig.5: conceptual hierarchy for 1-/2-itemsets



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

図 6: 1-item,2-itemset 混合列の結果（Q4 の結果）
Fig.6: η vs. time (hours). (Result of Q4)

(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．

用いる概念階層木を図 5 に示す．この図は，4.3 節の Q2 の実験結果 (図 3) 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 で分類したとき（図 3）の η である．階層木で $\eta \geq 2$ となる中間ノードの下には，図 3 で $\eta \geq 2$ 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 3 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 5 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”を入れた．

4.4.2 メッセージ階層を尺度次元にした場合

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 6 が, 該当する η 値の 1 時間単位列と見つかった時区間 (縦線で囲まれた領域) である. 図 6 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 3 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 3 では取らなかった範囲のログデータを調べてみた．

図 6 において矢印で示した部分の実際のログデータを図 6 右に示す．図中右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．

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，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．次に，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．また，長さ 1, 2 のアイテムセットの混合系列への Discover 演算の適用方法も述べ, 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．ただし, 2-アイテムセット混合列では適当な概念階層木の設定は自明ではなく, 文献 [2] のように 3 層サーバーのログなどで試験する必要がある. 特定の時間帯に偏って生じる特徴的なメッセージ階層を求める場合については文献 [5] を参照されたい．

[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] 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/.

[3] E.Hoke, J.Sun, C.Faloutsos, “InteMon: Intelligent System Monitoring on Large Clusters,” VLDB ’06, pp.1239-1242, 2006.

[4] R.Fagin, R.Guha, R.Kumar, J.Novak, D.Sivakumar, A.Tomkins, “Multi-Structural Databases”, 24th ACM PODS, pp.184-195, 2005.

[5] 涌波信弥, 大森匡, 星守, “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,” DEWS’07, E7-6, 2007.

涌波信弥 Shin-ya WAKUNAMI
2007 電気通信大学大学院修士課程了, 工修. 現在, (株) 日本電気. SQL システム自動管理等に関心を持つ. DBSJ 学生会員.

大森 匡 Tadashi OHMORI
1990 東京大学大学院博士課程了, 工博. 1994 より電気通信大学大学院助教授. 高性能 DB 等を研究. DBSJ 正会員.

星 守 Mamoru HOSHI
東京大学大学院修士課程了, 工博. 1992 より電気通信大学大学院教授. データ構造等を研究. ACM, IEEE 等, 各会員.