テキスト分析のためのOLAPシステム

猪口明博[†] 武田浩一[††]

今日の計算機環境の整備，インターネットの普及によりテキスト形式で記述された文書データが多く蓄積されるようになった．大量の文書から有用な知見が発見できれば非常に有益であるが，構造化されていない文書データから目的の文書を取捨，選択し，様々な角度から分析することは構造化されたデータに比べ容易ではない．一方，オンライン分析処理（OLAP）は各次元に沿って構造化されたデータを探索的に分析し，非定型の問合せに対して高速に分析結果を提供することができる．このような従来の多次元データベースを用いて文書データを分析する際，次元や次元値の定義，高速な応答時間を実現することは容易ではない．本稿では，大量の文書を分析するために，OLAP とオントロジを統合するためのデータモデルとデータ操作を提案する．

An OLAP Systems for Text Analytics

Akihiro Inokuchi[†] and Koichi Takeda[††]

Textual information has been very important for providing context to subject domains or entities described by conventional structured information, but it was primarily intended for human users and suffered from low utilization in analytical applications. The amount of accessible text (unstructured) data is increasing ever more rapidly. Such data potentially contains a great wealth of knowledge. However, analyzing huge amounts of text requires a tremendous amount of work for reading all of the text and organizing it. Online analytical processing (OLAP) systems provide fast answers for queries to perform trend, comparative, and time-based analysis by enabling exploration of structured data along multiple dimensions. When analyzing text data with the systems, it is difficult to store the text data in traditional multidimensional databases, to integrate ontologies with them, and to achieve fast computation. In this paper, we propose a data model, and a relational algebra to integrate ontologies with OLAP systems to analyze a huge set of text documents. The proposed method is implemented with a persistent store using preorders and postorders in a hierarchy.

1. はじめに

近年，電子化された文書は増加の一途をたどっており，インターネットの普及と相まって，これらの文書を手軽に入手することも可能となってきた．しかしながら，膨大な文書の中から所望の情報を探したい，あるいは文書の集合を分析して傾向をつかみたいといった，利用者の様々な要求に十分に対応できる情報アクセス手段はまだ乏しいのが現実である[5]．

多次元データベースは意思決定支援のため大量のデータを対話的に分析する技術である[15]．多次元データベースは計数と次元値を持ったファクトの集合としてデータを扱う．たとえば，小売のデータでは各購入履歴がファクトであり，量や価格が計数となる．商品種別，購入時間，購入場所などが次元値となる．オンライン分析処理（Online Analytical Processing: OLAP）では，データの傾向を分析するために，たとえば，ある特定の商品の月ごとの合計売上金額などを集計する．多次元データベースの次元は階層構造を持ち，詳細な次元値よりも利用者の望むデータの粒度でデータを選択・集計することが可能である．既存の多次元データベースの次元階層はバランス木で定義されることが多い．

多次元データベースの形態の 1 つであるスタースキーマやスノーフレークスキーマはファクト表と次元表にデータを保持する．ファクト表の各行は各ファクトに相当し，各列には外部キーによって次元表を参照する次元値を保持する．またファクト表には計数に対する列が 1 つ以上存在する．

このような多次元データベースで大量の文書データを分析し，文書集合の傾向を引き出す際は，**図 1** に

† 大阪大学産業科学研究所
The Institute of Scientific and Industrial Research, Osaka University
†† 日本アイ・ビー・エム株式会社東京基礎研究所
Tokyo Research Laboratory, IBM Japan

示すように文書をファクトとして扱い，語のカテゴリ（たとえば，生物医療文献では，蛋白質名，遺伝子名，疾患名，出版日）を分析の軸として用いることが考えられる．各セルには文書数が集約されている．従来の OLAP と同様に drill down，roll up，slice，dice，drill through [15] などの処理が使われる．文書データを従来の多次元データベースで分析する場合，多様な語☆の次元をいかに設計するかが問題となる．我々は階層として，UMLS（Unified Medical Language System）や GO（Gene Ontology）☆☆などのオントロジを用いるが，これらのオントロジは，従来の多次元データベースの多くが想定しているバランス木ではなく，DAG の一種であるので，スタースキーマなどで扱うことは困難である．オントロジの各節点を次元として扱う場合，各次元に対して各文書がただ 1 つの次元値を持つわけではなく，多くの欠値や多値が存在する．加えて，階層の節点の数や次元値となる語の種類は非常に多いので，それらの全組合せについて事前集計が必要となる MOLAP のような形態は不可能である．

本稿では，大量の文書データを分析するためのデータモデルとその上でのデータ操作を定義する．特に，下記の 2 点について詳細に述べる．

- オントロジを統合するためにいかにデータモデルとデータ操作を設計するか．
- 大量のデータを高速に集計するために，データをいかに関係データベースで保持，集計するか．

本稿の構成は以下のとおりである．2 章で階層とオントロジの関係について述べ，3 章でデータモデルとデータ操作について述べる．4 章では，データ操作を実行するための 1 つの実装方法を提案する．5 章では生物医療文献約 500,000 アブストラクトを用いて，性能を評価する．6 章では関連研究について議論し，7 章で本稿をまとめる．

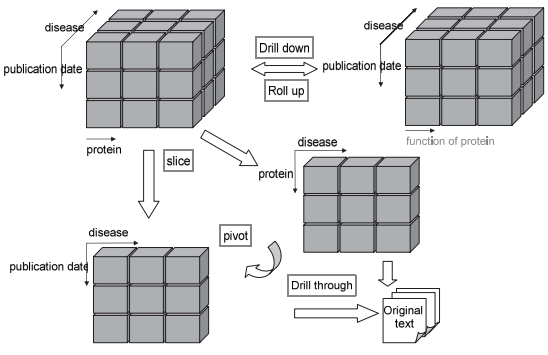


図 1 多次元データベースの操作
Fig. 1 Operations for multidimensional database.

2. 階層とオントロジ

本章では，文献 1) に従って階層とオントロジを定義する．S を空でない集合，$\leq \subseteq S \times S$ とするとき，$(S, \leq)$ を順序と呼ぶ☆☆☆．さらに S が反射的，推移的，反対称的な 2 項関係ならば，$(S, \leq)$ を半順序と呼ぶ．

定義 1（better）：$(S, \leq_1)$ と $(S, \leq_2)$ を順序とする．$\forall x, y \in S\ (x \leq_1 y \rightarrow x \leq_2 y)$ ならば，$(S, \leq_1)$ は $(S, \leq_2)$ より better であるという．さらに，$(S, \leq_1)$ が $(S, \leq_2)$ より better で，$(S, \leq_2)$ が $(S, \leq_1)$ より better でないとき，$(S, \leq_1)$ は $(S, \leq_2)$ より strictly better という．

定義 2（階層）：$(S, \leq)$ を半順序とする．S の階層は以下を満たす半順序 $(S, \preceq)$ である．

1. $(S, \preceq)$ は $(S, \leq)$ より better.
2. $(S, \leq)$ は $(S, \preceq)$ の反射的，推移的閉包である．
3. $(S, \leq)$ より strictly better で上記の 2 つ条件を満たす順序 $(S, \sqsubseteq)$ が存在しない．

定義 3（オントロジ）：Σ を文字列の有限集合，S を半順序関係とするとき，Σ に関するオントロジは Σ から S に対する階層へのマッピング Θ で定義される．

たとえば，S が $\{tire, car, hubcap\}$ であり，$tire$ が car の部品，$hubcap$ が car の部品，$hubcap$ が $tire$ の部品，すべてがそれ自身の部品だとすると，**図 2** に示されるように S の半順序関係は $\{(tire, tire), (car, car), (hubcap, hubcap), (tire, car), (hubcap, car), (hubcap, tire)\}$ であり，この中の階層は $\{(tire, car), (hubcap, tire)\}$ のみである．

階層は**図 3** に示されるように分類される [13]．

DAG：閉路がない有向グラフで定義される．

transitive anti-closed digraph：2 頂点間に長さ

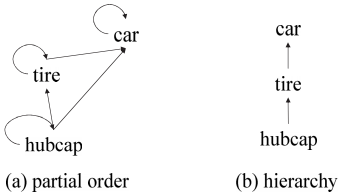


図 2 半順序と階層の例
Fig. 2 Examples of partial order and hierarchy.

☆ 実験データの異なり語は 13,640,593.
☆☆ UMLS: http://umlsinfo.nlm.nih.gov, GO: http://www.geneontology.org

☆☆☆ S の 2 要素の直接の関係を表すために $\leq$ を，推移的閉包を表すために $<^*$ を，推移的閉包か等号を表すために $\leq^*$ を用いる．

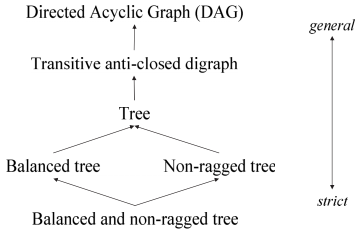


図 3 階層の分類
Fig. 3 Taxonomy of hierarchy.

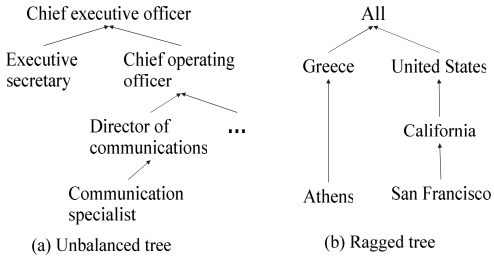


図 4 不平衡型階層，不ぞろい型階層の例
Fig. 4 An unbalanced tree and a ragged tree.

2 以上のパスと辺（長さ 1）があったときに，辺を取り除いた DAG を transitive anti-closed digraph と呼ぶ．

tree：1 つの節点（根）以外の節点が 1 つだけ上位の節点を持つ DAG で定義される．

balanced tree：**図 4** (a) は不平衡型の例である．一貫した親子関係を持っているが，論理的に不整合のレベルを持ち，階層の深さも異なる．最高執行責任者（COO）には部下がいるが，重役付きの秘書にはいない．最高経営責任者（CEO）と子節点の親子関係は一貫しているが，子節点が論理的に同等でない☆．

non-ragged tree：図 4 (b) は不ぞろい型の例である．各レベルの意味には一貫性があるが，Greece と Athens の間に節点がないため，深さに一貫性がない．

balanced and non-ragged tree：従来の多次元データベースの多くがこのクラスの次元階層である．

3. データモデルとデータ操作

本章では，文献 14) に従って，我々のデータモデルとその上でのデータ操作を定義する．

3.1 データモデル

階層（あるいは，オントロジ）$(S, \leq)$ が与えられたとき，ファクトスキーマは $\mathcal{S} = (\mathcal{F}, \mathcal{T})$ と定義される．ここで，$\mathcal{F}$ はファクトタイプ，$\mathcal{T}$ は $(S, \leq)$ より

☆ http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.db2_olap.doc/cmdhierarchy.htm

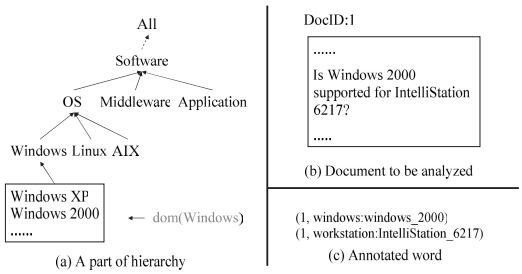


図 5 カテゴリ階層とファクト–階層関係の例
Fig. 5 Examples of a hierarchy and fact-hierarchy relations.

strictly better な階層タイプ $\mathcal{T} = (\mathcal{C}, \leq_{\mathcal{T}}, \top_{\mathcal{T}})$ である．文書集合を分析するために必要な $(S, \leq)$ 内の半順序関係は $\mathcal{T}$ に保持される．階層タイプは $(\mathcal{C}, \leq_{\mathcal{T}}, \top_{\mathcal{T}})$ であり，ここで，$\mathcal{C} = \{\mathcal{C}_j, j = 1, \cdots, n\}$ は $\mathcal{T}$ のカテゴリタイプの集合，$\leq_{\mathcal{T}}$ は $\top_{\mathcal{T}} \in \mathcal{C}$ を最上位に持つ $\mathcal{C}$ の半順序関係であり，$\forall \mathcal{C}_j \in \mathcal{C}$，$\mathcal{C}_j \leq^* \top_{\mathcal{T}}$ である．

タイプ $\mathcal{T}$ の階層インスタンス T は，$T = (C, \leq)$ である．ここで，C は $Type(c_j) = \mathcal{C}_j$ であるカテゴリ c_j の集合，$\leq$ は C の半順序関係である．c_j の親カテゴリ，子カテゴリを返す関数をそれぞれ $pred(c_j) = \{c' \mid c' > c_j\}$，$succ(c_j) = \{c' \mid c' < c_j\}$ とする．各カテゴリ $c \in C$ は定義域 $dom(c)$ を持ち，$dom(c)$ の各要素は，$c : v$ と表される．さらに各カテゴリに属する語，あるいは子孫節点に属する語を返す関数 $below$ を $below(c) = \{dom(c') \mid c' \leq^* c\}$ と定義する．

$F = \{f_i, i = 1, \cdots, m\}$ をファクトの集合とする．F と T の間のファクト–階層関係は，$f \in F$，$c \in C$，$v \in dom(c)$ とすると，集合 $R = \{(f, c : v)\}$ である．もし $\exists c' \in C((f, c' : v') \in R \wedge c' \leq^* c \wedge v = v')$ であるとき，$f \rightsquigarrow c : v$ と表す．

本稿のデータオブジェクトは $D = \{\mathcal{S}, F, T, R\}$ であり，ここで $\mathcal{S} = (\mathcal{F}, \mathcal{T})$ はファクトスキーマ，F は $Type(f) = \mathcal{F}$ であるファクト集合，$T = (C, \leq)$ は $c_j \in C$ と $\mathcal{C}_j \in \mathcal{C}$ に対して $Type(c_j) = \mathcal{C}_j$ である階層インスタンス，R は $(f, c : v) \in R \Rightarrow f \in F \wedge \exists c \in C\ (v \in dom(c))$ を満たすファクト–階層の関係である．

たとえば，階層の一部が**図 5** (a) で与えられたとき，F は文書 ID の集合である．図 5 (b) の 1 文（文書 ID は 1）は前処理で，語 “windows 2000” に対して，カテゴリ “windows” が付与され，R に $(1, windows : windows\_2000)$ がストアされる．

概念的に，R は正規化されていない関係 $R' \subseteq 2^{dom(c_1)} \times \cdots \times 2^{dom(c_n)}$ に相当する．ここで R' はスタースキーマでのファクト表に対応し，その各行と

$$g^{(1)}(c) = \{(f, c' : v') \mid (f, c' : v') \in R \wedge c = c' \wedge v' = dom(c')\}$$
$$g^{(2)}(c) = \{(f, c : c'') \mid (f, c' : v') \in R \wedge c'' \in succ(c) \wedge c' \leq^* c'' \wedge v' \in dom(c')\}$$
$$g^{(3)}(c) = \{(f, c' : v') \mid (f, c' : v') \in R \wedge c' \leq^* c \wedge v' \in dom(c')\}$$

図 6 ユーザ定義関数の例
Fig. 6 Examples of user-defined functions.

列は各ファクト（文書），次元（カテゴリ）に対応する．R' は各次元（属性）c_j に対し多数の欠値，多値を持つ．次元数は既存の OLAP で想定されている次元数よりはるかに大きく（5 章の実験データの c_j の数は約 240,000），それらの次元間に複雑な関係が存在するので，既存の手法ではそれらのデータを保持し，階層に沿って集計演算を高速に実行することは困難である．

集計操作の準備として関数 $g(c)$ と $G(g(c))$ を定義する．関数 $g(c)$ はファクト–階層関係を返すユーザ定義関数であり，関数 $G(g(c))$ は

$$G(g(c)) = \{(c : v) \mid (f, c : v) \in g(c)\}$$

と定義される．$k = 1, \cdots, q$ と $(c_k : v_k) \in G(g_k(c_k))$ に対し，関数 $Group$ を $Group(c_1 : v_1, \cdots, c_q : v_q) = \{f \mid f \in F \wedge \bigwedge_{k=1,\cdots,q}(f, c_k : v_k) \in g_k(c_k)\}$ と定義する．これらの関数はユーザが指定した各カテゴリ，キーワードに対する文書の分布を集計するために使われる．$g(c)$ の例として，**図 6** のような関数 $g^{(1)}$，$g^{(2)}$，$g^{(3)}$（$g \in \{g^{(1)}, g^{(2)}, g^{(3)}, \cdots\}$）の定義が可能である．$g^{(1)}(c)$ は，指定したカテゴリ c に属するキーワードごとに文書の分布を集計するために使われる．$g^{(2)}(c)$ は，指定されたカテゴリ c のサブカテゴリごとの文書の分布集計のために使われる．$g^{(3)}(c)$ は，$below(c)$ に属するキーワードごとに文書の分布を集計するために使われる．様々なユーザ定義関数を定義することで，ユーザの意図に応じた分析を可能とする．

3.2 データ操作

我々のデータモデルにおけるデータ操作を定義する．

選択 σ'：$c : v$，あるいは $c : *$ で表される原子述語 p_i の選言 $P = p_1$ *or* $\cdots$ *or* p_l が与えられたとき，選択 σ' は，$\sigma'_P(D) = (\mathcal{S}, F', T, R')$ で与えられる．ここで $F' = \{f \mid f \in F \wedge (f \rightsquigarrow p_1 \vee \cdots \vee f \rightsquigarrow p_l)\}$，$R' = \{(f', c : v) \in R \mid f' \in F'\}$ である．たとえば，カテゴリ ‘software’ に属する任意のキーワードを持つ文書集合は $\sigma'_{software':*}(D)$ となり，カテゴリ ‘gene_name’ に属する語 ‘BIKE’ を含む文書集合は $\sigma'_{gene\_name':'BIKE'}(D)$，カテゴリ c に属する語 v_1 と v_2 を含む文書集合は $\sigma'_{c:v_1}(\sigma'_{c:v_2}(D))$ となる．

差 $-$：$\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}$ であるデータ $(\mathcal{S}_1, F_1, T_1, R_1)$ と $(\mathcal{S}_2, F_2, T_2, R_2)$ が与えられたとき，差は $(\mathcal{S}, F_1, T_1, R_1) - (\mathcal{S}, F_2, T_2, R_2) = (\mathcal{S}, F', T_1, R')$ で表される．ここで $F' = F_1 - F_2$，$R' = \{(f', c : v) \mid f' \in F', (f', c : v) \in R\}$ である．たとえば v_1 を含むが v_2 を含まない文書集合は $\sigma'_{\top:v_1}(D) - \sigma'_{\top:v_2}(D)$ となる．

射影 π'：射影 π' は $\pi'_{c_1,\cdots,c_l}(D) = (\mathcal{S}, F, T, R')$ で定義される．ここで，$R' = \{(f, c : v) \in R \mid f \in F \wedge (f \rightsquigarrow c_1 : * \vee \cdots \vee f \rightsquigarrow c_l : *)\}$ である．

集計 α：カテゴリと関数の集合 $T = (c_1, \cdots, c_q, g_1, \cdots, g_q)$ が与えられたとき，集計 α は $\alpha[T, 'count'](D) = (\mathcal{S}', F', T', R')$ と定義される．ここで，各 $\mathcal{S}'$，F'，T'，R' は**図 7** で定義される．$('count', \top)$ は，半順序 $'count' < \top$ を表している．

たとえば，**表 1** は生物医療文献に対する集計 $\alpha['protein', g, 'count1'](\sigma'_{disease:'diabetes'}(D))$ の結果を表し，従来の多次元データベースにおける上位 N ランキングに相当する．表 1 の結果は糖尿病に共起する蛋白質名の高頻度語である．特許文献に対する $\alpha['company', 'GeneSymbol', g_1, g_2, 'count2'](D)$ の結果は**表 2** のようになり，製薬会社は他社と遺伝子の戦略を見い出せるかもしれない．同様に $\alpha['protein', 'protein', g_1, g_2, 'count3'](D)$ は蛋白質相互作用の可能性を文献から調べるのに有用であるであるが，この演算は，従来の多次元データベースと異なり，同じカテゴリを解析の軸に選択している．

上記の定義を用いて OLAP 操作は以下のように示される．

roll-up & drill-down：従来の多次元データベースでは，次元ロールアップと階層ロールアップの 2 種類がある．たとえば，S をある商店で購入された商品，場所，購入日を持つファクト表 $S(prod, city, time, sale)$ とし，$L(city, state, country)$ を場所に関する次元表とするスタースキーマを考える．次元ロールアップは，1 次元が削減される場合は ${}_{prod,city}\chi_{prod,city,SUM(sale)}(S)$ と表され，2 次元の場合は ${}_{city}\chi_{city,SUM(sale)}(S)$ と表される．階層ロールアップは

${}_{prod,state,time}\chi_{prod,state,time,SUM(sale)}(S \bowtie_{city} L)$

と表される☆．一方，我々のデータモデルにおける次

☆ ${}_{a}\chi_{a,sum(b)}$ は SQL “SELECT a, SUM(b) FROM ... GROUP BY a” に相当する演算である．

$$
\begin{aligned}
&\mathcal{S}' = (\mathcal{F}', \mathcal{T}')\\
&\mathcal{F}' = 2^{\mathcal{F}}\\
&\mathcal{T}' = (\mathcal{C}', \leq'_{\mathcal{T}'}, \top'_{\mathcal{T}'})\\
&\mathcal{C}' = \mathcal{C} \cup \{'count'\}\\
&\leq'_{\mathcal{T}'} = \leq_{\mathcal{T}} \cup \{('count', \top)\}\\
&\top'_{\mathcal{T}'} = \top_{\mathcal{T}}\\
&F' = \{Group(c_1 : v_1, \cdots, c_q : v_q) \mid (c_1 : v_1, \cdots, c_q : v_q) \in G(g_1(c_1)) \times \cdots \times G(g_q(c_q))\\
&\qquad\qquad \wedge Group(c_1 : v_1, \cdots, c_q : v_q) \neq \emptyset\}\\
&T' = (C', \leq')\\
&C' = C \cup \{'count'\}\\
&\leq' = \leq_T \cup \{('count', \top)\}\\
&R' = R'_1 \cup R'_2\\
&R'_1 = \{(f', c' : v') \mid \exists (c_1 : v_1, \cdots, c_q : v_q) \in G(g_1(c_1)) \times \cdots \times G(g_q(c_q))\\
&\qquad\qquad (f' = Group(c_1 : v_1, \cdots, c_q : v_q) \wedge f' \in F' \wedge \exists k\ (c_k : v_k = c' : v'))\}\\
&R'_2 = \bigcup_{(c_1:v_1, \cdots, c_q:v_q) \in G(g_1(c_1)) \times \cdots \times G(g_q(c_q))} \{(s, 'count' : |s|) \mid s = Group(c_1 : v_1, \cdots, c_q : v_q) \wedge s \neq \emptyset\}
\end{aligned}
$$

図 7 集計の定義
Fig. 7 Definition of aggregation.

表 1 Top N ランキング
Table 1 Top N ranking.

protein	# of documents
Flavohemoprotein	347
Lamin L	240
Insulin	151
nterferon gamma precursor	97
···	···

表 2 2 次元マップ
Table 2 2 dimensional map.

	LEPR	TM4SF2	INS	ADAM2	···
company1	x_{11}	x_{12}	x_{13}	x_{14}	···
company2	x_{21}	x_{22}	x_{23}	x_{24}	···
company3	x_{31}	x_{32}	x_{33}	x_{34}	···
···	···	···	···	···	···

元ロールアップは，1 次元削減に対して $\alpha[c_1, \cdots, c_q, g_1, \cdots, g_q, 'count'](D)$ から $\alpha[c_1, \cdots, c_h, c_{h+2}, \cdots, c_q, g_1, \cdots, g_h, g_{h+2}, \cdots, g_q, 'count'](D)$ へ移る操作に相当し，階層ロールアップは $c'_h \in pred(c_h)$ とすると $\alpha[c_1, \cdots, c_q, g_1^{(1)}, \cdots, g_q^{(1)}, 'count'](D)$ から $\alpha[c_1, \cdots, c_{h-1}, c'_h, c_{h+2}, \cdots, c_q, g_1^{(1)}, \cdots, g_{h-1}^{(1)}, g_h^{(2)}, g_{h+1}^{(1)}, \cdots, g_q^{(1)}, 'count'](D)$ へ移る操作に相当する．

slice & dice：従来の多次元データベースにおけるスライスの例を，関係代数式で表すと，${}_{city,time}\chi_{city,time,SUM(sale)}(\sigma_{prod=p_1}(S))$，ダイスは ${}_{prod,city,time}\chi_{prod,city,time,SUM(sale)}(\sigma_P(S))$ で表される．ここで $P = (prod \in \{p_1, p_2\}\ and\ city \in \{c_3, c_4\})$ である．一方，我々のデータモデルにおけるスライス操作は $\alpha[T, 'count'](\sigma'_p(D))$ であり，ダイス操作は $\alpha[T, 'count'](\sigma'_{p_1\ or\ p_2}(\sigma'_{p_3\ or\ p_4}(D)))$ で表される．

pivot：ピボット操作は視点の軸を入れ替え，表示を転置する可視化のための操作であり，$\alpha[c_1, c_2, g_1, g_2, 'count']$ から $\alpha[c_2, c_1, g_2, g_1, 'count']$ へ移る操作に相当する．

4. 実　　装

多次元データベースの応答速度をあげるための 1 つの方法は事前に集約しておくことである．しかしながら，問合せを受け取る前に，すべての値とカテゴリの組合せに対して事前に集計しておくことは不可能である．たとえば，5 章で用いるデータは，24 万以上のカテゴリを持ち，1,300 万以上の異なり語を持つ．

本章では，できるだけ短い応答時間を得るためのデータ構造とテーブルスキーマを定義する．文書分析のためのカテゴリはバランス木ではなく transitive anti-closed digraph であるので，スタースキーマにストアすることはできない．文書の集計の計算速度をあげるために，階層を以下のように索引付けする．タイプが $\top_{\mathcal{T}}$ に等しい根節点 $c_{root} \in C$ からはじめて，各節点に前順，後順，深さを割り当てながら深さ優先に探索していく．ただし，内部節点ではバックトラックせずに，葉節点でのみバックトラックする．

3.2 節で定義したように，選択，集計など演算では，指定したカテゴリ c の子孫カテゴリとその値をいかに効率良くデータから抽出するかが重要となるが，c とデータのカテゴリが子孫関係にあるかを判定するために，以下の条件で容易に判定することが可能である．もし節点 A が節点 B の祖先であるなら，以下が成り立つ[3]．

$$g^{(1)}(c) = \pi_{id,preorder,value}(V \bowtie_{preorder1=preorder} H_c)$$
$$g^{(2)}(c) = \pi_{id,H.preorder1,H.catename\ as\ value}((H_c \bowtie_{H.parent=H_c.preorder1} H) \bowtie_{H.preorder1 \le V.preorder \le H.preorder2} V)$$
$$g^{(3)}(c) = \pi_{id,preorder,value}(V \bowtie_{preorder1 \le preorder \le preorder2} H_c)$$

図 9 ユーザ定義関数の実装例
Fig. 9 Implementation of user-defined functions.

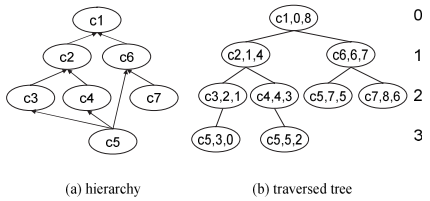


図 8 階層カテゴリと巡回木
Fig. 8 A hierarchy and a traversed tree.

A の前順 $< B$ の前順　かつ
A の後順 $> B$ の後順　(1)

たとえば，**図 8** (a) のカテゴリ階層を深さ優先探索すると，図 8 (b) のような木が得られる．各節点内の文字は，カテゴリ名，前順，後順である．この図で，c_2 の子孫の前順は c_2 の前順より大きく，c_2 の子孫の後順は c_2 の後順より小さい．図 8 (b) を巡回木と呼ぶ．

巡回木とファクト-階層関係をストアするために，テーブル CATEGORY H と KEYWORD V を定義する．

```
CATEGORY (CATENAME    CHARCTER,
          PATH        CHARCTER,
          PREORDER1   INTEGER,
          PREORDER2   INTEGER,
          PARENT      INTEGER)
KEYWORD (ID          INTEGER,
         PREORDER    INTEGER,
         VALUE       CHARCTER)
```

H の各レコードは巡回木の各節点に相当し，CATENAME，PATH，PREORDER1，PREORDER2，PARENT は，それぞれ節点のカテゴリ名，ルート節点からのパス，前順，後順と深さの和，親節点の前順である．V の各レコードは，$(f, c : v)$ に相当し，ID，PREORDER，VALUE は，それぞれ文書 ID，カテゴリ c の前順，$dom(c)$ 内の値 v である．後順の代わりに PREORDER2 を使う理由は，祖先–子孫関係を条件式 (1) の代わりに

$A's\ preorder1\ (= A$ の前順$) < B$ の前順
$\le A's\ preorder2 = A$ の後順 $+ A$ の深さ (2)

で，調べることができるからであり，これにより表と索引をストアする領域を削減することができる．

以上の 2 つのテーブルを用いて，3.2 節で述べたデータ操作を以下のように実装する．以下の定義で，入力として与えられる c と等しい値を持つタプルは H に複数存在するが，任意に 1 つが選択される．すなわち，“$P = (catename = c)$ である．$\sigma_P(H)$” は “$\sigma_P(H)$ FETCH FIRST 1 ROWS ONLY”（以下，H_c と表記）に置き換えられが，結果には影響を与えない．

選択 σ'：選択は $\sigma'_{c:v}(D) = (H, V')$ で実行できる．ここで $V' = \sigma_{id\ in\ I}(V)$，$I = \pi_{id}(V \bowtie_P H_c)$，$P = (preorder1 \le preorder \le preorder2\ and\ value = v)$ である．この演算は H と V の結合を必要とするが，H_c から返されるタプルは 1 つのみであるので，V の選択とほぼ同様の計算時間で実行できる．

差 $-$：2 つのデータオブジェクトの差は $(H, V_1) - (H, V_2) = (H, V')$ で定義される．ここで $V' = \sigma_{id\ in\ (\pi_{id}(V_1) - \pi_{id}(V_2))}(V_1)$ である．

射影 π'：射影演算は $\pi'_c(D) = (H, V')$ で実行できる．ここで $V' = \pi_{id,preorder,value}(V \bowtie_P H_c)$，$P = (preorder1 \le preorder \le preorder2)$ である．

集計 α：集計は $\alpha[(c_1, \cdots, c_q, g_1, \cdots, g_q), 'count'](D) = {}_{value_1,\cdots,value_q}\chi_{value_1,\cdots,value_q,count(distinct\ id)}(X)$ で実装される．ここで，$X = g_1(c_1) \bowtie_{id=id} \cdots \bowtie_{id=id} g_q(c_q)$ である．ユーザ定義関数 $g^{(1)}$，$g^{(2)}$，$g^{(3)}$ は，**図 9** に示されるように実装される．

5. 評価実験

5.1 使用データと前処理

提案した手法を評価するために生物医療文献を用いた．多くのライフサイエンス研究者によって生物医療文献データベース MEDLINE☆が広く使われている．MEDLINE は，米国立医学図書館（NLM）の米国立バイオテクノロジー情報センタ（NCBI）☆☆により管理されている巨大な生物医療文献アーカイブであり，1960 年代から生物医療文献が登録されており，現在 1,300 万件を超える文献が登録されている．本稿の実験では，著者や Mesh Term などの定型項目と非構造化情報のアブストラクトを MEDLINE から 503,989 件選択し，実験を行った．前処理の詳細は文献 8) を

☆ http://www.nlm.nih.gov/databases/databases_medline.html
☆☆ http://www.ncbi.nlm.nih.gov & http://www.nlm.nih.gov

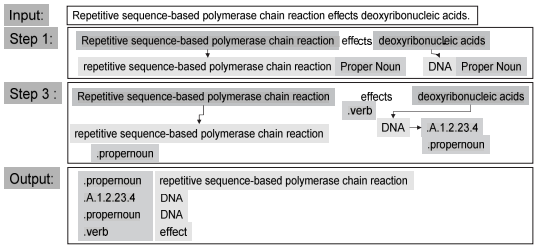


図 10 前処理の例
Fig. 10 Example of preprocessing.

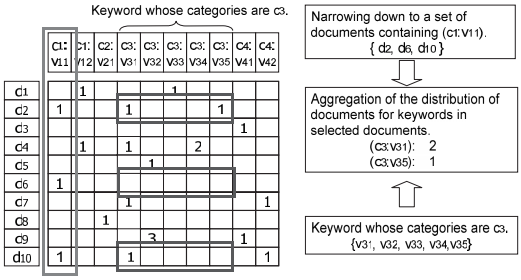


図 11 DTM による実装
Fig. 11 Implementation using a DTM.

参照されたい．

図 10 はある 1 文の処理の過程である☆．入力文の “deoxyribonucleic acids” に対し，“DNA” を正書形として，“proper noun” を品詞として付与する．構文解析後，各語にカテゴリを付与する．図 10 で，“.A.1.2.23.4” はルート節点からある節点までのパスを表している．

503,989 件のアブストラクトを前処理後，$(f, c : v)$ の種類，巡回木の節点数，異なり語はそれぞれ，193,185,919，340,154，13,640,593 であった．カテゴリの中には出版日などの構造情報も含むので，通常の OLAP のように出版日を解析の軸とすることも可能である．

5.2 文書–用語行列による実装

4 章で提案した手法と比較するため，文書–用語行列（Document-Term Matrix: DTM）による手法を用いた．一般論として，専用のアルゴリズムと索引による実装の方が，関係データベースのような汎用的な手法による実装よりも高速である．一方で，新たな機能の追加や他のシステムとの統合など拡張は後者の方が容易である．本節，および次節では汎用手法を用いた手法でも適切な索引を行うことで，十分に専用手法と同等の応答時間が得られることを示す．

本節では，簡単な例を用いながら DTM による $\alpha[c, g, 'count'](\sigma'_p(D))$ の計算方法について述べる．語の集合と文書集合をそれぞれ $\{t_1, \ldots, t_n\}$，$\{d_1, \ldots, d_m\}$ とする．DTM は $m \times n$ の行列 $M = (m_{ij})$ であり，各要素 m_{ij} は文書 d_i 中に語 t_j が何回出現したかを表している．この行列全体をそのままメモリ上に保持するのは困難であるが，この行列は非常に疎なので，非ゼロの要素の位置と m_{ij} の値のみを保持することでメモリ上に保持することが可能である．

前章で述べたように，各語にはカテゴリが付与されているので，以下のような DTM を用いる．DTM の行は文書集合 $\{d_1, \ldots, d_m\}$ であり，列はカテゴリと語のペアの集合 $\{c_j : v_{jk} \mid v_{jk} \in dom(c_j), j = 1, \cdots, n\} \cup \{c_j : * \mid j = 1, \cdots, n\}$ とする．$c_j : *$ はサブカテゴリの集計 $g^{(2)}$ を高速に処理するために使われ，d_i と $c_j : *$ に対応する行列 M の要素は，$d_i \rightsquigarrow c_j : *$ であるとき，非ゼロである．

図 11 は，ユーザがカテゴリ c_1 が付与された語 v_{11} で文書集合を絞り込んだ後に，カテゴリ c_3 を指定したときに，$\alpha[c_3, g^{(1)}, 'count'](\sigma'_{c_1:v_{11}}(D))$ をいかに演算するかを示している．はじめに $(c_1 : v_{11})$ を含む文書集合 $\{d_2, d_6, d_{10}\}$ に絞り込む（処理 1）．それと同時に，カテゴリ c_3 が付与された語の集合 $\{c_3 : v_{31}, c_3 : v_{32}, c_3 : v_{33}, c_3 : v_{34}, c_3 : v_{35}\}$ を出力する（処理 2）．これらに処理の後に，$\{d_2, d_6, d_{10}\}$ 内の語に対する文書分布 $\{(c_3 : v_{31}) : 2, (c_3 : v_{35}) : 1\}$ を出力する（処理 3）．処理 3 は処理 1，2 に比べ長い計算時間を要する．

ユーザがカテゴリ階層の内部節点を指定した場合は，サブカテゴリごとの集計 $\alpha[c_3, g^{(2)}, 'count'](\sigma'_P(D))$ も可能である．この場合は，処理 2 で $\{c'_3 : * \mid c'_3 \in succ(c_3)\}$ が返される．

5.3 比較実験

4 章で述べた手法は Java で実装され，SQL を生成し，JDBC（Java Database Connectivity）経由で RDB にアクセスする．一方，5.2 節で述べた比較手法は C++で実装されている．比較実験のため，IBM IntelliStation（Windows XP，Opteron-2.2 GHz，メモリ 2 GB）の計算機を使用し，性能比較を行った．

図 12 は全 340,154 カテゴリに対する，$\alpha[c, g, 'count'](D)$ の計算時間の結果を示している．この図の DB と DTM は，4 章と 5.2 節で述べた手法を表し，KW と SUB は $\alpha[c, g, 'count'](D)$ の関数 g として，$g^{(1)}$ と $g^{(2)}$ が使われたことを表している．“DTM SUB 1k” は全文書からランダムに選択された 1,000 文書に対する結果であり，手法 DB はサンプリングを行

☆ この例では 1 文のみの処理過程であるが，実際はアブストラクト（文書）単位で処理される．

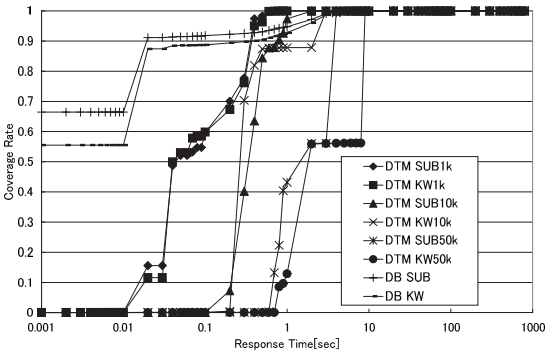
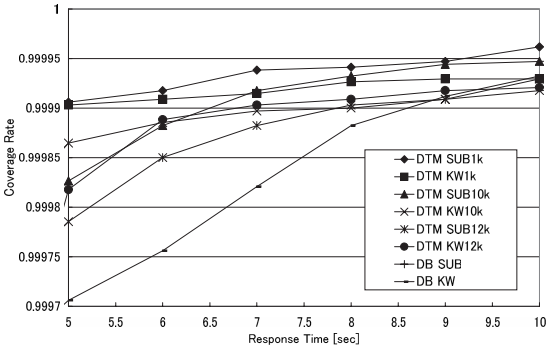


図 12 全データに対する実験結果
Fig. 12 Results for all documents.



∗“DB SUB” のカバー率は約 0.99775
図 13 “cancer” を含む文書に対する実験結果
Fig. 13 Results for documents containing “cancer”.

わず全文書を集計した．図中の各点 (x, y) は全 340,154 カテゴリ中 y%に対して，x 秒以内で集計結果が得られたことを示している．理想的な結果は，左上に凸な曲線である．DB KW は 0.1 秒で約 89%のカテゴリに対して結果を返したのに対し，DTM はサンプル数 1,000 文書に対し約 60%，10,000 文書に対し約 0.01%であった．さらに，語 ‘cancer’ を含む文書集合に対しても同様の実験 $\alpha[c, g, 'count'](\sigma_{\top:'cancer'}(D))$ を行ったが，図 12 と同様の結果が得られた．したがって，図 12 からは，DB の方が DTM よりも高速であるといえる．

一方，本稿は対話的分析手法に関する手法であるので，ウェブページは 8 秒以内に結果返すべきという 8 秒ルールの観点から結果を見ると，10 秒前後のカバー率が高い手法の方がより良い手法であるといえる．**図 13** は 5 秒から 10 秒，99.97%から 100%を拡大した $\alpha[c, g, 'count'](\sigma_{\top:'cancer'}(D))$ の結果である．8 秒ルールの観点から図 13 は DB が DTM よりも劣っていることを示している．**表 3** は $\alpha[c, g, 'count'](D)$ と $\alpha[c, g, 'count'](\sigma_{\top:'cancer'}(D))$ の結果を要約したものである．手法 DB の平均計算時間は DTM よりも短いが，10 秒以内に結果を返せないカテゴリ数は，DTM よりも DB の方が多いといえる．

“cancer” を含む文書の選択 $\sigma'_{\top:'cancer'}(D) = (H, V')$ に対する V' を得るための問合せは，

$$V' = \sigma_{id\ in\ \pi_{id}(V \bowtie_{(value='cancer')} H_c)}(V) = \pi_{id,V_b.preorder,V_b.value}(V_a \bowtie_P V_b) \quad (3)$$

となる．ここで，$P = (V_a.id = V_b.id\ and\ V_a.value = 'cancer')$，$V = V_a = V_b$ である．式 (3) は

```
SELECT Vb.ID, Vb.PREORDER, Vb.VALUE
FROM V AS Va, V AS Vb
WHERE Va.ID=Vb.ID AND Va.VALUE='cancer'
```

と表される．DB が特定のカテゴリに対して多くの計算時間を必要とする理由の 1 つは，レコード数 193,185,919 であるテーブル V のセルフジョインである．そこで，V を以下のように複数のテーブルに分割することを考える．$id(V)$ をテーブル V の文書 ID を返す関数とする．V を $id(V) = \bigcup_i id(V_i)$ と $id(V_i) \cap id(V_j) = \emptyset (i \neq j)$ を満たすように，複数のテーブル V_i に分割する．式 (3) は WHERE 節に $V_a.id = V_b.id$ を持つので，次の式は共通の ID を保持しないテーブルの結合を避けることができる．

$$\alpha[c, g^{(1)}, 'count'](\sigma_{\top:'cancer'}(D)) = {}_{value}\chi_{value,sum(count)}(\bigcup_i R_i), \quad (4)$$

ここで $R_i = {}_{value}\chi_{value,count(distinct\ id)\ as\ count}(V'_i)$,

表 3 実験結果の要約（上段：全文書，下段：“cancer” を含む文書）
Table 3 Summary of experimental results.

Method	平均計算時間	10 sec.†	100 sec.‡
DTM KW 1k	0.143 [sec]	1	0
DTM KW 5k	0.284 [sec]	26	0
DTM KW 10k	0.526 [sec]	16	0
DTM KW 50k	4.293 [sec]	75	0
DB KW	0.185 [sec]	10	0
DTM SUB 1k	0.138 [sec]	2	1
DTM SUB 5k	0.176 [sec]	2	0
DTM SUB 10k	0.405 [sec]	5	1
DTM SUB 50k	2.091 [sec]	33	0
DB SUB	0.137 [sec]	20	2

Method	平均計算時間	10 sec.†	100 sec.‡
DTM KW 1k	0.177 [sec]	24	5
DTM KW 5k	0.355 [sec]	116	5
DTM KW 12k	0.594 [sec]	27	7
DB KW	0.267 [sec]	23	0
DTM SUB 1k	0.195 [sec]	13	1
DTM SUB 5k	0.352 [sec]	65	0
DTM SUB 12k	0.534 [sec]	24	2
DB SUB	0.413 [sec]	706	5

†，‡ はそれぞれ 10 秒，100 秒以内に結果を返せないカテゴリ数．

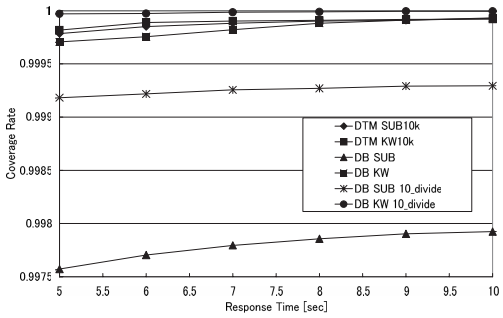


図 14 10 分割したテーブルに対する計算時間
Fig. 14 Computation times for KEYWORD divided into 10 tables.

$\bigcup$ は UNION ALL，V_i' は式 (3) により計算される．式 (4) を書き換えると

```
SELECT VALUE, SUM(COUNT)
FROM
 (
   SELECT Vb.VALUE, COUNT(DISTINCT Vb.ID)
    FROM V_0 AS Va, V_0 AS Vb
    WHERE Vb.PREORDER=pre  AND
      Va.ID=Vb.ID AND Va.VALUE='cancer'
    GROUP BY VALUE
   UNION ALL
   ...
   UNION ALL
   SELECT Vb.VALUE, COUNT(DISTINCT Vb.ID)
    FROM V_k AS Va, V_k AS Vb
    WHERE Vb.PREORDER=pre  AND
      Va.ID=Vb.ID AND Va.VALUE='cancer'
    GROUP BY VALUE
 ) A
 GROUP BY VALUE,
```

と表される．ここで pre はカテゴリ c の前順である．

図 14 は KEYWORD を 10 分割した場合の DB の結果，しなかった場合の DB の結果，DTM の結果を比較した結果である．分割することで，10 秒におけるカバー率は KW の場合 99.993%から 99.999%に，SUB の場合 99.79%から 99.93%に上昇していることが分かる．この実験は単一の計算機で行われたが，商用のデータベースシステムは並列化をサポートしているので，容易に複数の計算機上で実行することが可能である．

6. 考　　察

$\alpha[c, g, 'count'](D)$ を頻度の降順に並べ替えた結果は上位 N ランキングに相当するが，上位 N ランキングはしばしば自明な高頻度語を返す場合がある．文書集合の特徴的な語を取り出すために，相対頻度を用いる[19]．D は全文書集合であり，選択により絞り込まれた文書集合を D_s とする．語 $p_{jk} = (c_j : v_{jk})$ の相対頻度は，以下のように計算される．

$$relative\_freq(p_{jk}, D_s) = \frac{c(p_{jk}, D_s)/|D_s|}{c(p_{jk}, D)/|D|},$$

ここで，$c(p_{jk}, D)$ は D 中で語 p_{jk} を含む文書数である．たとえば，図 11 の場合，$D_s = \{d_2, d_6, d_{10}\}$ であり，$relative\_freq((c_3 : v_{31}), D_s)$ は $\frac{\frac{2}{3}}{\frac{4}{10}} = \frac{5}{3}$ となる．

4 章で述べたような前順，後順のような手法を用いなかったら，節点 A と節点 B の距離を n とすると，テーブル CATEGORY を n 回結合する必要がある．しかしながら，提案手法は節点の祖先–子孫関係を調べるために，テーブルの結合を必要とせず，高速に集計が可能である．前順–後順による索引法は 1982 年に提案された方法であり，XML 文書を索引付けするために最近注目されている方法である[4]．前順–後順法以外にも接頭辞ラベル[2]，Dewey order 法[18]，素数ラベル[20]，VLEI コード[6]，k 分木への埋め込み[7] などの方法が XML 文書を索引付けするために提案されている．前順–後順法や素数ラベルの短所は XML 文書に要素を加えたときに，前順などのラベルを振りなおす必要があることであるが，カテゴリ階層は頻繁に更新されないと想定しているので，本稿では前順，後順による階層の索引法を採用した．またオントロジと辞書の更新によるテキスト処理結果への更新波及や DB への再ロードのコストは階層索引に比較して，圧倒的に高いため，分析的なアプリケーションにおける本稿の RDB による実装方法がきわめて高い実用性を持つ．

文献 11)，12)，19) では，テキストマイニングにおいて関係抽出が重要であることを述べている．これはキーワード単位の集計から文書集合の頻出単語の傾向を把握することができるが，実際どのようなことが述べられているかを把握することができないためである．そこで，コールセンタを対象とする場合に「何がどうした」という情報を，生物医学文献の場合には「何が何にどう作用した」を前処理で取り出すだけで，大まかな問題を把握するのに有用となる．前者は（主語，動詞）の 2 項関係，後者は（主語，動詞，目的語）の 3 項関係であり，これらの関係は前処理で抽出され，分析時にこれらの関係は 1 つの語 v として扱われる．

文献 16)，17) もまた文書を対象とした多次元データモデルを提案している．本稿と違い，文献 17) では，文書に述べられている取引内容などをファクトとして扱い，同じ取引内容が複数の文書で述べられている場

合，それらの文書から 1 つファクトを生成する．ファクトの抽出後は，月ごと，地域ごとの取引金額の集計分析など従来の多次元データベースが想定していた分析を行う．一方，本稿では，文書を 1 ファクトとして扱っているために，月ごとの文書数の増減，語の分布の増減などの傾向を分析するのに適している．たとえば，企業などでは，顧客からの問合せなどをコールセンタで対応し，文書を含む記録を蓄積している．蓄積されたデータは，コールセンタのコスト削減や将来の機能拡張，商品に不具合の早期発見などに用いられている．典型的な例として，お客様から問合せの急激な増加を対話的分析で発見し，FAQ としてホームページに掲載する運用業務に用いられている．また，掲載により問合せ件数の減少を検証するためにも用いられている[11]．

文献 9)，10) もまた，本稿と同様に文書集合を対話的に分析する手法である．下記は文献 9) で使われた MDX（Multi-Dimensional eXpression）の例であり，1998 年第 1 四半期にニューヨークで出版された文書のうち，語 “forests” を含む文書を分析する問合せである．

```
SELECT not empty [DocId].members on rows,
    {[Measure].[Tf]} on columns
FROM docInfo
WHERE ([Term].[forest],[1998][quarter 1],
                    [location].[New York])
```

この問合せで，[Term].[forest] で示されるように，Term の次元は，最上位節点 ⊤ と語の 2 階層であるので，本稿のように複雑な階層は取り込めず，roll-up，drill-down のような語の意味階層を考慮したような分析はできない．文献 10) は文書全体に付与されたカテゴリ（生物医療文献では Mesh Term）を次元値として使うことを想定しているが，文献に付与される Mesh Term は平均 23.5 であり，本稿が想定している語数（実験は 1 文書あたり 383.3）に比べてはるかに少なく，十分な応答時間を得るのは難しい．

7. おわりに

本稿では，大量の文書集合を解析するために，オントロジとファクト–階層関係を統合するためのデータモデルとデータ操作を提案し，効率良く実行するための実装方法を示した．提案手法の性能を示すために生物医療文献約 500,000 件を使い，文書の絞り込み，集計演算を高速に実行することができることを示した．

謝辞 本研究は，筆頭著者が日本アイ・ビー・エム株式会社東京基礎研究所に所属時に行われた研究成果である．本研究の遂行にあたり，テキストマイニングシステム IBM TAKMI（Text Analysis and Knowledge MIning）の研究・開発に携わってこられたの方々に多くのサポート，助言をいただいた．つつしんで感謝の意を表する．

参考文献

1) Bonatti, P.A., Deng, Y. and Subrahmanian, V.S.: An ontology-extended relational algebra, *Proc. 2003 IEEE International Conference on Information Reuse and Integration*, pp.192–199 (2003).
2) Cohen, E., Kaplan, H. and Milo, T.: Labeling dynamic xml trees, *Proc. 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp.271–281 (2002).
3) Dietz, P.F.: Maintaining order in a linked list, *Proc. 14th Annual ACM Symposium on Theory of Computing*, pp.122–127 (1982).
4) Grust, T.: Accelerating xpath location steps, *Proc. 2002 ACM SIGMOD International Conference on Management of Data*, pp.109–120 (2002).
5) 市村，長谷川，渡部，佐藤：テキストマイニング—事例紹介，人工知能学会誌，Vol.16, No.2, pp.192–200 (2001).
6) Kobayashi, K., Liang, W., Kobayashi, D., Watanabe, A. and Yokota, H.: Vlei code: An efficient labeling method for handling xml documents in an rdb, *Proc. 21st International Conference on Data Engineering*, pp.386–387 (2005).
7) Lee, Y.K., Yoo, S.-J., Yoon, K. and Berra, P.B.: Index structures for structured documents, *Proc.1st ACM International Conference on Digital Libraries*, pp.91–99 (1996).
8) 松澤，長野，村上，浦本，武田：ライフサイエンス向けテキストマイニングツール MedTAKMI，情報処理学会研究会報告，No.51, pp.33–40 (2003).
9) McCabe, M.C., Lee, J., Chowdhury, A., Grossman, D.A. and Frieder, O.: On the design and evaluation of a multi-dimensional approach to information retrieval, *Proc. 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.363–365 (2000).
10) Mothe, J., Chrisment, C., Dousset, B. and Alau, J.: Doccube: Multi-dimensional visualisation and exploration of large document sets, *Journal of the American Society for Information Science and Technology*, Vol.54, No.7, pp.650–659 (2003).
11) Nasukawa, T. and Nagano, T.: Text analysis

and knowledge mining system, *IBM Systems Journal*, Vol.40, No.4, pp.967–984 (2001).
12) 那須川：テキストマイニングを使う技術/作る技術—基礎技術と適用事例から導く本質と活用法, 東京電機大学出版局 (2006).
13) Niemi, T., Nummenmaa, J. and Thanisch, P.: Logical multidimensional database design for ragged and unbalanced aggregation, *Proc. 3rd International Workshop on Design and Management of Data Warehouses*, p.7 (2001).
14) Pedersen, T.B. and Jensen, C.S.: Multidimensional data modeling for complex data, *Proc. 15th International Conference on Data Engineering*, pp.336–345 (1999).
15) Pedersen, T.B. and Jensen, C.S.: Multidimensional database technology, *IEEE Computer*, Vol.34, No.12, pp.40–46 (2001).
16) Pérez, J.M., Pedersen, T.B., Llavori, R.B. and Aramburu, M.J.: IR and olap in xml document warehouses, *Proc.27th European Conference on IR Research*, pp.536–539 (2005).
17) Pérez, J.M., Llavori, R.B., Aramburu, M.J. and Pedersen, T.B.: A relevance-extended multi-dimensional model for a data warehouse contextualized with documents, *Proc. ACM 8th International Workshop on Data Warehousing and OLAP*, pp.19–28 (2005).
18) Tatarinov, I., Viglas, S., Beyer, K.S., Shanmugasundaram, J., Shekita, E.J. and Zhang, C.: Storing and querying ordered xml using a relational database system, *Proc. 2002 ACM SIGMOD International Conference on Management of Data*, pp.204–215 (2002).
19) Uramoto, N., Matsuzawa, H., Nagano, T., Murakami, A., Takeuchi, H. and Takeda, K.: A text-mining system for knowledge discovery from biomedical documents, *IBM Systems Journal*, Vol.43, No.3, pp.516–533 (2004).
20) Wu, X., Lee, M.-L. and Hsu, W.: A prime number labeling scheme for dynamic ordered xml trees, *Proc. 20th International Conference on Data Engineering*, pp.66–78 (2004).

(平成 18 年 9 月 15 日受付)
(平成 19 年 2 月 27 日採録)

(担当編集委員 石川 博, 有次 正義, 片山 薫, 中島 伸介)

猪口 明博（正会員）

1975 年生．2000 年大阪大学大学院工学研究科通信工学専攻博士前期課程修了．同年日本アイ・ビー・エム（株）に入社，東京基礎研究所に配属．2004 年大阪大学大学院工学研究科通信工学専攻博士後期課程修了．工学博士．データマイニング，機械学習，テキストマイニング，データウェアハウス，医療情報処理の研究・開発に従事．2002 年 Journal of Computer Aided Chemistry 論文賞，人工知能学会 2003 年度研究会（知識ベースシステム研究会）優秀賞受賞．2007 年 4 月大阪大学産業科学研究所助教．

武田 浩一（正会員）

1981 年京都大学工学部情報工学科卒業．1983 年同大学大学院工学研究科情報工学専攻修了．同年日本アイ・ビー・エム入社．東京基礎研究所にて自然言語処理や機械翻訳システムの研究開発に従事．1987〜1989 年米国カーネギー・メロン大学客員研究員．現在はテキストマイニングおよび医療情報分析の研究に従事．ACM，電子情報通信学会，言語処理学会各会員．