Web集約質問処理のための検索エンジンの関係データベースインタフェース

大島裕明[†] 小山聡[†] 田中克己[†]

本稿では，「Web 集約質問」を処理するための Web 検索エンジンに対する関係データベースインタフェースを提案する．Web 検索の目的がページを発見することであるのに対して，Web 集約質問処理では Web 全体における集約された情報を取得することを目的とする．Web 集約質問処理には，(1) Web 検索，(2) 自然言語処理（NLP）によるデータ抽出，(3) 情報集約といった機能が必要である．関係データベースシステムはすでに強力な集約機能を保有しているため，我々は関係データベース上に Web 検索や自然言語処理機能のためのインタフェースを実装した．ユーザは SQL によって，アドホックな Web 集約質問処理や，そこで得られる知識とローカルデータベース上の既存のデータとの結合を行うことが可能となる．本稿では，これらのインタフェースの設計と実装，それを利用したアプリケーションについて述べる．

A Relational Database Interface to Search Engines for Processing Web Aggregate Queries

Hiroaki Ohshima,[†] Satoshi Oyama[†] and Katsumi Tanaka[†]

We propose a relational database interface to conventional Web search engines for processing “Web aggregate queries”. Whereas the purpose of a Web search is usually to find specific Web pages, the purpose of processing a Web aggregate query is to obtain aggregated information from the overall Web. To process Web aggregate queries, we need several functions such as (1) Web searching, (2) natural language processing for text extraction, and (3) aggregation of data. Because a relational database system has a robust aggregation capability, we implemented a relational database interface to conventional Web search engines with a natural language processing capability. Using SQL through the relational interface, users can formulate and execute several *ad hoc* SQL queries. Local databases can also be joined with Web search results through the relational database interface. We describe the design and implementation of the relational database interface, and its applications.

1. はじめに

Web 検索を利用する際の目的は，主としてページを見つけることである．Web 検索エンジンは莫大な量のメタデータを保有しており，ページの検索という Web 検索が持つ役割は現在非常に重要なものとなっている．Web 検索エンジンが保有するメタデータは Web 検索に利用されるばかりでなく，ある種の知識を取得することにも利用できるものである．たとえば，ある語の定義や，ある商品の評判，ある山の高さといった知識は，Web 検索を利用して Web 上の情報を集約することができれば得られる情報である．

あるユーザが「ポルシェ」のライバルを求めたいとする．そのような場合には，「VS」という語が競技対戦相手を接続する語であることに着目して，「ポルシェ VS」というクエリで Web 検索を行い，「VS」の直後に現れる「フェラーリ」や「ホンダ」といった「ポルシェ」のライバルを表す語を求める手法が考えられる．さらに，多くの検索結果を見ることによって，Web 全体では何が「ポルシェ」の典型的なライバルと考えられているか，といった集約的な知識が得られると考えられる．一般的な知識は Wikipedia [1] や Wiktionary [2] に記述されていることも多いが，世の中のあらゆる語が網羅されているわけではなく，また，評判情報などが記述されることは少ないため，Web 検索と簡単な言語パターンを用いた手動での集約的知識の取得が行われる機会は多い．しかし，このような作業はユーザにとって負担が大きいものである．

このような，Web からの集約的知識を取得する一

† 京都大学大学院情報学研究科社会情報学専攻
Department of Social Informatics, Graduate School of Informatics, Kyoto University

連の処理を我々は Web 集約質問処理と呼ぶ．典型的な Web 集約質問処理は次の 3 つの段階からなる．

Web 検索 知識取得のための言語パターンを考慮した Web 検索を行い，検索結果を取得する．

NLP によるデータ抽出 自然言語処理によって，言語パターンに適合する語やフレーズを集約質問の答えの候補として抽出する．

情報集約 得られた語に対して出現回数を数えたり，Web 検索の検索ページ数を用いたりして評価を行う．

このような Web 集約質問処理は**語レベルの知識**を得るためのものであるといえる．より単純な Web 集約質問としては，ある語を含むページが Web にどの程度存在しているかを Web 検索エンジンから得られる検索ページ数で求めることができる．それにより，与えた 2 語の共起度のような語の関係性を計算することが可能である．一方，我々が**語レベルの知識**と呼ぶものは，新たな語を発見するような知識である．すなわち，ユーザが 1 語与えたときに，ある特定の関係にある別の語を発見するというようなものである．そのための Web 集約質問処理では，Web 検索結果から言語パターンを用いて語を抽出するような処理を必要とする．共起度の計算などとは異なり，得られる知識は多様でより有用なものとなる．Web 集約質問処理は様々な状況において有用であることが多いにもかかわらず，現在はユーザが毎回プログラムを作成する以外にはほとんど方法がない．本稿では，このような Web 集約質問処理を行うことを目的として，関係データベース処理環境に Web 検索や自然言語処理機能を付加した環境を提案する．

関係データベースを Web 集約質問処理環境として利用することにはいくつかのメリットが存在する．

- Web 検索 API などの既存の Web サービスや自然言語処理機能を統合するより上位層の API を提供可能である．
- ローカルデータベース上のデータと Web 集約質問によって得られる知識の統合が可能となる．
- Web 集約質問処理はしばしばアドホックに行われるが，SQL はアドホックに利用する言語として優れている．ここで，アドホックな利用とはユーザがその場で思いついた考えを試すような利用のことである．

関係データベースはテーブルの関係性を扱うことができるため，Web 検索を含む様々な Web サービスを仮想テーブルとして実装することにより，SQL ベースの Web サービス統合環境となりうる．さらに自然言語処理機能を付加することによって，Web からの知識取得や Web サービスの統合のための上位層の API になる．Web 集約質問処理によって得られる知識を用いたアプリケーションの 1 つとしては，Web 検索前や検索後における利用が考えられる．たとえば，Web 検索前には，ユーザが与えたクエリキーワードから，上位語，下位語，同位語，トピックとなるような語を求めることによって，クエリの修正やキーワード想起支援を行うことができる．また，Web 検索後にも，Web から得られる知識を用いて，検索結果についてより多くの情報を提示するサービスなどが考えられる．Nakamura ら[3] の Web 検索結果のそれぞれのページでクエリの語の典型的な話題がどの程度網羅されているかという情報を付加して提示するサービスなどはそのようなサービスの一例であり，ここではクエリとして使われた語の典型的なトピックを表す語を求める技術が必要となる．次に，ローカルデータベース上のデータと Web 集約質問によって得られる知識の統合が可能となるメリットがある．データベースにいくつかのテーブルが存在しているときに，Web からの知識発見によって，新たな関係性やデータを既存のデータベースに付加することが可能になる．SQL はアドホックな利用にもよく使われる言語であり，知識取得のための言語パターンを考えればすぐに様々な質問が考えられる Web 集約質問に適しているといえる．既存のデータベースのデータを Web からの知識を用いてアドホックに分析することなども考えられる．

関係データベースにはもともと情報の集約を行うための機能があるため，Web 検索を行い，その結果に対して自然言語処理を行えるようになれば，Web 集約質問処理に必要な機能が揃う．我々は，Microsoft SQL Server 2005 上に，Web 検索や自然言語処理機能をテーブル値関数やスカラ値関数として実装した．テーブル値関数とは，引数を与えることによって結果がテーブルとして得られる関数で，関係データベース上での仮想テーブルである．

以降，2 章で関連研究について，3 章で Web 集約質問処理について，4 章で実装について，5 章でアプリケーション例について，6 章で既存システムとの比較と議論について，7 章でまとめについて述べる．

2. 関 連 研 究

既存の関連する研究として，関係データベースにおいて Web 検索を利用することに関する研究[4)~6)]，言語パターンをクエリとして利用できる検索システム[7),8)]，言語パターンを用いた知識発見[9)~11)]，語の関係性や

類似度を Web 検索の結果ページ数のみを用いて計算する研究[12)~15)]，Web サービスの統合のためのシステム[16),17)] などをあげることができる．また，我々はこれまで Web 集約に関する研究を行ってきた[18)]．

WSQ/DSQ[4)] は関係データベース環境で Web 検索を仮想テーブルとして扱うものである．`WebPages` という仮想テーブルで，Web 検索結果のランキング，URL，日付が取得できる．`WebCount` は Web 検索の結果ページ数を返す関数である．Google[19)] や AltaVista[20)] が `WebPages` や `WebCount` として利用することができるようになっている．WSQ/DSQ 上では Web 検索結果の統合や，結果ページ数を用いた簡単な Web 集約質問処理を行うことが可能である．しかし，検索結果に含まれるタイトルやスニペットといったテキスト情報を扱わず，自然言語処理機能もないため，Web 検索結果のテキスト情報に含まれる情報を集約することによる**語レベルの知識**の取得を行うことは難しい．

WebQL[5)] は，QL2 Software より発売されているデータマイニング，データ抽出や統合のためのツールである．SQL を拡張した言語により，様々なソースからのデータ取得機能や，データ抽出機能を利用することができる．利用できるデータソースの 1 つは Web ページであり，与えられた URL のページを取得可能である．また，与えられた URL から Web ページをクロールすることも可能である．データ抽出のための自然言語処理機能も保持する．WebQL では Web のクローリングを利用した Web 集約を行うことが可能であり，使用の一例は，Web 検索を行い，その検索結果で得られた URL のページをダウンロードし，その本文からある言語パターンを満足するような語を抽出するというものである．

Cafarella ら[7)] はクエリとして言語パターンが利用できる検索システムを作成した．KnowItAll[8)] は，そのシステムを利用した Web 上でのサービスである．大量の Web ページを収集してこの検索システムを利用すれば，上位語/下位語の発見を行うアプリケーションなどを作成することが可能となる．言語パターンによる質問処理に特化して設計された検索システムであり，自然言語処理を利用するサービスが高品質に実現可能である．しかし，既存のデータベースや他の Web サービスとの統合は考えられていない．また，文書を大量に用意する必要があるため，大規模なサービス提供における利用が中心になると考えられる．

Hearst[9)] は大量の文書からいくつかの言語パターンを用いることによって語の上位下位関係を発見する手法を提案した．言語パターンのうちの 1 つは “such as” というものである．我々は，与えられた 1 語の兄弟概念を表す語をいくつか発見する手法を提案した[10),11)]．兄弟概念を表す語が「や」で接続されることに着目し，与えられた語の前後に「や」を接続したクエリで Web 検索を行い，その検索結果のタイトルやスニペットに含まれる文字列から兄弟概念を表す語の候補を取得する．それぞれの語は「や」の前後で出現した回数の相乗平均をスコアとしており，「や」の両側に出現しなかった語は結果には含まれない．これは Web 集約質問処理の一例であり，提案している環境で実行可能な知識取得手法である．

Web 検索の結果ページ数のみを用いて語の関係性や類似度を計算する研究も存在する．類義語を発見する手法に利用するために，Turney[12)] は語の共起性を，Baroni ら[13)] は相互情報量を計算する手法を提案した．また，Google Similarity Distance[15)] は語の類似性を計算する手法である．Oyama ら[14)] はある 1 語が別の 1 語の詳細語であるかどうかを Web 検索の結果ページ数のみを用いて判定するために，文書のタイトルと本文という階層構造に着目し，“intitle:” や”intext:” といった検索エンジンの機能を利用した手法を提案した．これらの研究は，Web 検索による検索ページ数が大規模な文書コーパスの解析の代わりに利用できることを表している．

複数の Web サービスを統合する環境も多く提案されているが，それらは主にサービスの接続を実現することに注力している．Caméléon#[16)] は Web サービス，ローカルデータベース，XML データなどを統合することが可能なシステムである．各々のサービスのためにラッパを記述することによって，関連するサービスを統合したサービスを実現することができる．

3. Web 集約質問処理

3.1 Web 集約質問処理の概要

Web 集約質問処理とは，関係データベースにおける集約質問処理と同様に，Web の情報を集約して 1 つの値を求める処理である．関係データベースにおける集約質問処理とは，データの集合から，最大，最小，平均，レコード数といった 1 つの値を求めるものである．SQL では，`max()`，`min()`，`average()`，`count()` などの集約関数と `GROUP BY` 節を利用する．我々が提案する**語レベルの知識**取得のための Web 集約質問処理では，まず，Web 検索を用いて集約処理するためのデータを収集し，そこで得られたデータの最大，最小，平均，出現数などを取得する．特に**語レベルの知識**取得のための Web 集約質問処理は，典型的には以下の

3 つの段階からなる．

- Web 検索
- NLP によるデータ抽出
- 情報集約

Web 集約質問処理において集約されるデータは Web 上に存在する．まず，Web からデータを収集する必要がある．既存の Web 検索エンジンを利用することが最も容易な手法である．典型的な Web 集約質問は何らかの条件を満たす語のうち，しばしば現れるものを発見するというものである．

Web 集約質問の一例として，「cat」の上位語を求めるために「such as a cat」の直前に最もよく現れる語を求めるというものがある．「such as a cat」というクエリで Web 検索を行えば，Web 検索結果に含まれるタイトルやスニペットの部分に必要な部分が存在しており，元の Web 文書をダウンロードして収集する必要はない．実際の「such as cat」での Web 検索結果のスニペットには，「for exercising a small animal such as a cat or kitten is disclosed」や，「taking in a domesticated animal such as a cat or dog」といった文が含まれており，「such as a cat」の直前の語として「a small animal」や「a domesticated animal」を得ることができる．

次に必要なことは，何らかの条件を満たす語を発見することである．ここで自然言語処理が必要となる．基本的なものとしては以下のようなものがある．

- パターンに適合する語の抽出機能
- 文書の分かち書きと品詞情報付加機能

これらの機能を利用して語のリストを作成することによって，集約されるデータが集められたことになる．

最後はデータの集約処理である．Web 検索結果から抽出された語の出現回数を数えるなどして，Web 集約質問に対する回答を得る．このとき，Web 検索で得られる検索ページ数を抽出された語の評価のためなどに利用することも考えられる．

以上が，Web 集約質問処理の概要である．

3.2 Web 検索エンジンの関係データベースインタフェース

Web 集約質問処理では，「Web 検索」「NLP によるデータ抽出」「情報集約」が行われるが，我々は，関係データベース処理環境に Web 検索や自然言語処理機能を付加することにより，Web 集約質問処理環境を作成することを提案する．

図 1 は，提案する環境における，Web 検索，自然言語処理機能，ローカルデータベースの関係を表しており，これらはすべてユーザが SQL によって扱えるようになっている．

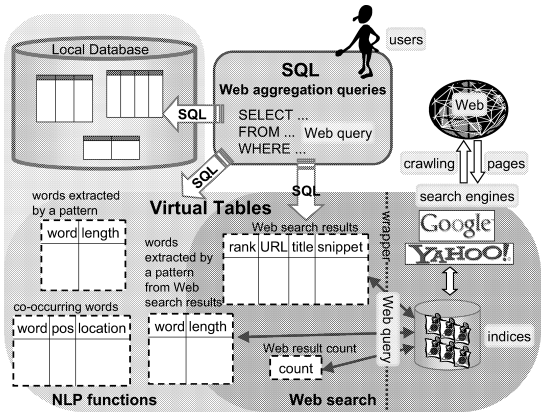


図 1 Web 検索エンジンの関係データベースインタフェース
Fig. 1 Relational database interface to Web search engines.

Web 検索エンジンが提供する API により，検索ランキング，URL，タイトル，スニペットといった情報が得られるが，それらの情報を関係データベース上でテーブルとして扱えるようにラッパを作成する．Web 検索結果のテーブルは通常のテーブルとは異なり，クエリが与えられることによって異なる値が出力される仮想テーブルである．文字列から指定した言語パターンに適合する語を抽出したり，文中である語と共起する語を取得したりするための機能も仮想テーブルにする．Web 検索結果のタイトルやスニペットの文字列から，ある言語パターンに適合する語を抽出することは，SQL では Web 検索の仮想テーブルと語抽出の仮想テーブルの JOIN を得ることになる．「Web 検索」と「NLP によるデータ抽出」が SQL で記述できれば，その結果として得られたデータの集約は，関係データベースが持つ集約機能を最大限利用することができる．主に，出現回数を数えることが多くなるため，SQL の `count()` 関数がよく使われることになる．また，あるクエリに対する Web での検索ページ数を得る関数を用意することで，検索ページ数を情報集約時に評価のために利用することも考えられる．

ユーザが利用する言語が SQL であるため，ローカルデータベースのデータをこれらの処理に組み込むことが容易に可能である．たとえば，ローカルデータベースに保有する文書が格納されているときに，Web 集約質問処理から「フェラーリ」のライバルを表す語を発見し，「フェラーリ」のライバルについて書かれたような文書をローカルデータベースから検索する，といったことが実行可能である．

表 1 SELECT * FROM Search_url_title_snippet('Kyoto University', 2) の結果例
Table 1 Example result of SELECT * FROM Search_url_title_snippet('Kyoto University', 2).

row 1	rank	1
	url	http://www.kyoto-u.ac.jp/index-e.html
	title	Kyoto University - HOME
	snippet	The Kyoto University Website: an overview of academic programs, campus life, research, education, news, general ...
row 2	rank	2
	url	http://www.bun.kyoto-u.ac.jp/phisci/English.html
	title	Philosophy and History of Science, Kyoto University
	snippet	Dept. of Phil.Hist.Science, Graduate School of Letters, Kyoto University. Sakyo, Kyoto 606-8501, JAPAN ...

表 2 SELECT * FROM Extract_word_length('Kyoto is a beautiful city', extractPattern) において extractPattern の値によって異なる出力結果
Table 2 Example result of SELECT * FROM Extract_word_length('Kyoto is a beautiful city', extractPattern).

extractPattern					
'Kyoto is <term>'		'is <term> city'		'<term> a beautiful city'	
word	length	word	length	word	length
a	1	a beautiful	1	is	1
a beautiful	2			Kyoto is	2
a beautiful city	3				

4. 実　　装

4.1 SQL Server 2005 における SQL CLR と APPLY 演算子

我々は，Microsoft SQL Server 2005 [21] を利用して，Web 集約質問処理環境を構築した．SQL Server 2005 は SQL CLR という機能を持っており，SQL 上で利用できる関数を，C#や Visual Basic などのプログラミング言語を利用して作成可能である．ユーザが作成可能な CLR 関数には，文字列や数値などの単一のスカラ値を返す**スカラ値関数**，結果としてテーブルを返す**テーブル値関数**，データの集合が与えられたときにそれらを計算した結果を単一のスカラ値として返す**集約関数**という 3 つのタイプがある．我々は，Web 検索や自然言語処理機能をテーブル値関数やスカラ値関数として実装した．

APPLY 演算子は SQL Server 2005 でテーブルと，その列の値をテーブル値関数の引数としたテーブルを仮想的に結合（JOIN）させるための演算子である．通常は FROM 節の後にテーブル名が記述され，続けて APPLY 演算子とともにテーブル値関数が記述される．機能は，テーブルの各行に対してテーブル値関数を適応させるものであり，結果はテーブルの行の数だけ実行されたテーブル値関数の結果テーブルの和演算をとったものとなる．

4.2 Web 検索のための関数

Search_url_title_snippet(query, num) はテーブル値関数であり，Web 検索を行った結果を返す．引数 query が Web 検索で用いられるクエリ文字列であり，num が検索結果として取得する最大数である．結果テーブルの各行は，Web 検索結果の各ページの順位，URL，タイトル，スニペットを保持する．Yahoo Web 検索 Web サービス [22] を利用して，英語と日本語のそれぞれの Search_url_title_snippet を実装した．使用例を**表 1** に示す．

WebCount(query) はスカラ値関数であり，与えられた Web 検索クエリで得られる検索ページ数を返す．この関数を用いて語の類似度計算などを行うことが可能である [12),13),15)]．

4.3 NLP によるデータ抽出のための関数

Extract_word_length(value, extractPattern) はテーブル値関数であり，ある言語パターンに適合する語を与えられた文字列から抽出する．引数 value が対象となる文章であり，extractPattern が言語パターンを表す正規表現である．extractPattern は内部に <term> という文字列を含み，その位置に適合する語が抽出される．結果テーブルの各行は，抽出された語と，その語がいくつの単語・形態素からなるかを表す length を保持する．**表 2** に例を示した．<term> が extractPattern の先頭や最後に存在する場合は，1 語の単語，2 語が結合された複合語，3 語が結合さ

表 3 SELECT * FROM Cooccurring_word_pos_location('Kyoto is a beautiful city.', 'beautiful') の結果
Table 3 Example result of SELECT * FROM Cooccurring_word_pos_location('Kyoto is a beautiful city.', 'beautiful').

word	pos	location
Kyoto	NNP	−3
is	VBZ	−2
a	DT	−1
beautiful	JJ	0
city	NN	1
.	.	2

れた複合語，などが結果として返される．もし，抽出したい語が 1 語の単語である場合は，WHERE 節に length = 1 という条件を付加すればよい．<term> が extractPattern 中の先頭や最後でない場所に書かれた場合は結果は 1 行となる．

Cooccurring_word_pos_location(value, targetWord) はテーブル値関数であり，与えられた文章中で対象の語のまわりにどのような語が共起しているかを求めるものである．引数 value が文章であり，targetWord が対象の語である．結果の各行は，共起している語，その語の品詞，対象の語からの位置を保持する．結果の例が**表 3** に示されている．品詞情報付加ツールとしては SStagger[23] を利用した．この関数の典型的な使用法は対象の語の特徴を表す語を取得することである．品詞情報が付加されているため，たとえば，共起する名詞のみを取得するといったことも可能である．

ToNumerical(value) はスカラ値関数であり，数値を返す．この関数は文字列の引数 value から数値である部分を発見し，強制的に数値に変換する．たとえば，「三千七百七十六」と「標高 3,776 メートル」は両方とも 3776 という数値に変換される．数値が発見できなかったときは 0 を返す．Web 情報集約では，文字列の切り出しによって情報を取得するが，数値情報である場合には表記の揺れが起こりやすいためこの関数が有用となる．

5. アプリケーション例

本章では，実装した環境を用いたアプリケーション例として，「渋谷」の典型的な印象を表す語の取得，与えられた名前の山の高さの取得，与えられた語の兄弟概念を表す語の取得，ローカルデータベース上のデータと Web 集約による知識の結合（JOIN）の例を示す．

5.1 「渋谷」の典型的な印象を表す語の取得

ここでは，渋谷の典型的な印象を表す語を求める手法を考える．様々な手法が考えられるが，ここでは近くに共起する形容詞が印象語であるという仮定に基づいた手法を用いる．**図 2** が渋谷の典型的な印象後を求めるための SQL 文である．

```
SELECT cw.word, COUNT(cw.word) c
FROM Search_url_title_snippet('Shibuya', 100) ws
CROSS APPLY
  Cooccurring_word_pos_location(ws.description, 'Shibuya') cw
WHERE cw.pos = 'JJ'
AND cw.location >= -10
AND cw.location <= 10
GROUP BY cw.word
ORDER BY c DESC;
```

図 2 「渋谷」の典型的な印象語を求めるための SQL 文
Fig. 2 SQL for obtaining typical impression of "Shibuya".

表 4 求められた「渋谷」の典型的な印象語の例
Table 4 Example result of typical impression of "Shibuya".

word	c
trendy	4
special	3
new	3
Japanese	2
famous	2
fashionable	2

まず，Search_url_title_snippet 関数により，*Shibuya* という語をクエリとして 100 件の検索結果を取得している．そして，Cooccurring_word_pos_location 関数を用いて，検索結果のスニペット中で *Shibuya* という語と共起する語を求めている．求める語の条件が WHERE 節に記述してあり，pos 列が「JJ」であること，すなわち品詞が形容詞であることと，語の出現位置が *Shibuya* から 10 語以内であることが指定されている．そして，条件に合致して抽出された語の出現回数を数えている．この結果の一例は**表 4** のようになる．

5.2 山の高さの取得

図 3 は山の名前が与えられたときにその高さを返すユーザ定義関数 MtHeightFunc(@query) を表している．たとえば，次の SQL 文はエベレストの標高（メートル）である 8848 を返す．

```
SELECT dbo.MtHeightFunc('Mt.Everest');
```

関数中では，まず，*Mt.Everest height* というクエリで Web 検索が行われる．得られたスニペット中の，*meters*，*metres*，*mtrs* などの前にある数値がエベレストの標高である可能性が高い．正規表現を使うことにより，様々なパターンに適合する数値を取得することができる．ToNumerical 関数で文字列を数値に変換し，最もよく現れた数値のみを結果として返す．

```
CREATE FUNCTION MtHeightFunc (@query varchar(50))
RETURNS int
AS
BEGIN
DECLARE @h int;
DECLARE @sr TABLE (snippet NVARCHAR(1000));
DECLARE @wex TABLE (height float);
-- Web Search
INSERT INTO @sr (snippet)
SELECT snippet
FROM Search_url_title_snippet(@query + ' height', 100);
-- Word Extraction by a Lingistic Pattern
INSERT INTO @wex (height)
SELECT dbo.ToNumerical(word) FROM @sr CROSS APPLY
Extract_word_length(snippet,
            '<term>(meters|metres|mtrs|mts|[mM][\s\)])')
WHERE length = 1;
-- Aggregation
SELECT TOP 1 @h = height FROM @wex
WHERE height > 0 GROUP BY height ORDER BY count(height) DESC;
RETURN(@h)
END
```

図 3 与えられた名前の山の高さを取得するユーザ定義スカラ値関数
Fig. 3 User-defined scalar-valued function for obtaining height of a mountain.

表 5 抽出された語の一部とそれらの変換によって得られた数値
Table 5 Part of extracted words and the numbers converted from the words.

word	ToNumerical(word)
8848	8848
in	0
2,500	2500
k	0
8848	8848
kilo	0
8844.43	8844.43
8,848	8848

表 5 は ExtractedWordLength で抽出された語とそれらを ToNumerical 関数によって数値に変換したものである．抽出された語の中には，数値を表すものではないものや，間違った数値が含まれているが，最も多く出現するのは 8848 という正解の数値である．データの集約によってある程度間違った情報が含まれていたとしても有益な情報を得ることが可能となる例といえる．

表 6 ではローカルデータベースのデータと関数 MtHeightFunc を利用した Web 情報集約の統合の例を示している．ローカルデータベースに Mountains というテーブルが存在し，フィールド name に山の名前が格納されているとする．そこには標高の情報がないため，高さの高い順に並べ替えることはできない．そこで，先ほど SQL で定義した関数 MtHeightFunc を利用した下記の SQL を実行することで，Mountains にある山の名前を高さの高い順に並べ替えることが可能となる．

表 6 既存の Mountains テーブルの内容（左）と SELECT name, dbo.MtHeightFunc(name) height FROM Mountains ORDER BY height DESC の結果（右）
Table 6 Mountains table (left) and result of SELECT name, dbo.MtHeightFunc(name) height FROM Mountains ORDER BY height DESC (right).

name
Aconcagua
Carstensz Pyramid
Mont Blanc
Mt.Elbrus
Mt.Everest
Mt.Fuji
Mt.Kilimanjaro
Mt.Kosciuszko
Mt.McKinley
Vinson Massif

name	hight
Mt.Everest	8848
Aconcagua	6959
Mt.McKinley	6194
Mt.Kilimanjaro	5895
Mt.Elbrus	5642
Vinson Massif	4897
Carstensz Pyramid	4884
Mont Blanc	4807
Mt.Fuji	3776
Mt.Kosciuszko	2228

```
CREATE FUNCTION CoordinateTermsFunc (@query varchar(50))
RETURNS @Results TABLE (word nvarchar(100),
                        ca int, cb int, v float)
AS
BEGIN
DECLARE @agAfter TABLE (word NVARCHAR(100), ca int);
DECLARE @agBefore TABLE (word NVARCHAR(1000), cb int);
-- Web Searching / Word Extraction / Counting the Appearance
INSERT INTO @agAfter (word, ca)
SELECT word, count(word)
FROM Search_url_title_snippet('"' + @query + ' or"', 100)
CROSS APPLY Extract_word_length(description,
                                @query + ' or <term>')
GROUP BY word;
INSERT INTO @agBefore (word, cb)
SELECT word, count(word)
FROM Search_url_title_snippet('"or ' + @query + '"', 100)
CROSS APPLY Extract_word_length(description,
                                '<term> or ' + @query)
GROUP BY word;
-- Return results
INSERT @Results
SELECT ta.word, ta.ca, tb.cb, SQRT(ta.ca * tb.cb) v
FROM @agAfter ta, @agBefore tb
WHERE ta.word = tb.word
RETURN
END
```

図 4 与えられた語の兄弟概念を表す語を取得するユーザ定義テーブル値関数
Fig. 4 User-defined table-valued function for obtaining coordinate terms.

```
SELECT name, dbo.MtHeightFunc(name) height
    FROM Mountains ORDER BY height DESC;
```

この処理には約 28 秒かかった．そのほとんどは Web 検索のための時間であり，ネットワークが十分に速ければより短い時間で処理可能である．

5.3 兄弟概念を表す語の発見

図 4 は与えられた語の兄弟概念を表す語を発見するユーザ定義のテーブル値関数 CoordinateTermsFunc (@query) を表している．たとえば，変数 @query の値が *Ferrari* のとき，*"Ferrari or"* と *"or Ferrari"* という 2 つによって Web 検索が行われ，取得された検

表 7 CoordinateTermsFunc の実行例
Table 7 Example result of CoordinateTermsFunc.

Ferrari			
word	ca	cb	v
Porsche	11	14	12.4
Lamborghini	8	10	8.9
McLaren	5	2	3.2
Renault	3	2	2.4
a Porsche	2	2	2.0
Aston	3	1	1.7

Paris			
word	ca	cb	v
London	8	23	13.6
Frankfurt	4	4	4.0
New York	2	5	3.2
in	2	3	2.4
Amsterdam	2	2	2.0
Venice	1	2	1.4

索結果のタイトルやスニペットから，*Ferrari or* の後に出現する語，*or Ferrari* の前に出現する語が抽出される．情報集約の段階では，*or* の前後に現れる回数の相乗平均を計算し，その値が 0 以上であるもののみを結果として返す．

表 7 は CoordinateTermsFunc のいくつかの実行例を示している．表中の ca は *or* の後に現れた回数，cb は *or* の前に現れた回数，v はそれらの相乗平均である．

5.4 Web からの知識を利用したローカル文書検索

ローカルデータベースにユーザの文書が保有されており，ユーザが *Ferrari* をタイトルに含む文書を検索するときには，以下の SQL が実行されるものとする．

```
SELECT docid, title
FROM UserDocuments
WHERE title like '%Ferrari%';
```

表 8 はこの SQL の結果例である．前節で述べた CoordinateTermsFunc を利用すると，*Ferrari* の兄弟概念，一種のライバルを表す語と考えられる語をいくつか取得することができる．これを利用すると，ユーザは *Ferrari* のライバルについてタイトルに含むような文書を検索することが，以下の SQL の実行によって可能となる．

```
SELECT docid, title
FROM UserDocuments,
 CoordinateTermsFunc('Ferrari') rivals
WHERE title like '%' + rivals.word + '%';
```

表 9 がその結果例である．Web から得られる知識から，*Ferrari* と関連すると考えられた文書が検索されている．

表 8 *Ferrari* についての文書を検索した結果例
Table 8 Example result of searching for documents about *Ferrari*.

docid	title
1	History of Enzo Ferrari
4	Scuderia Ferrari- Formula 1

表 9 *Ferrari* のライバルについての文書を検索した結果例
Table 9 Example result of searching for documents about rivals of *Ferrari*.

docid	title
2	The Excellence of Porsche
8	Lamborghini Gallardo Superleggera Coupe Specs
5	McLaren MP4-21 Formula One car
7	Renault Sales, Servicing and Used Cars

表 10 提案システムと既存システムとの違い
Table 10 Difference between related works and our system.

	WSQ/DSQ	WebQL	KnowItAll	提案システム
文書の蓄積が不要	○	○		○
既存言語の仕様拡張なし			(-)	○
自然言語処理機能あり		○	○	○
知識取得手法が蓄積可能				○
他ユーザが機能拡張可能				○

これは，ローカルデータベースのテーブルと Web 集約によって得られる知識の結合（JOIN）の簡単な例である．ここでは，兄弟概念を表す語を利用しているが，Web 集約質問処理によって得られる知識は多岐にわたり，さらに多くのローカルデータベースとの結合を行うアプリケーションが考えられる．

6. 既存システムとの比較と議論

我々の提案システムと WSQ/DSQ，WebQL，KnowItAll の違いを明らかにするため比較を行う．**表 10** は提案システムと他システムの違いを表している．

1 点目は，文書を事前に大量に収集しておく必要があるかどうかである．KnowItAll 以外のシステムは，Web 検索や URL を与えることによって文書を必要時に必要な分だけ収集する機能を持っているのに対し，KnowItAll は検索システムであり，あらかじめ用意した文書を保持し，インデックスを作成しておく必要がある．WSQ/DSQ では Web 検索を行うことができるが，文書を収集するというよりも，むしろ，検索結

果の URL を取得するというものである．WebQL では指定された URL の内容を取得することが可能であり，Web 検索結果を取得するための URL を与えることで，検索結果の内容のページを一 Web ページとして取得することは可能である．ただし，1 件 1 件の検索結果のタイトル，スニペット，URL を認識することは難しい．WebQL では主に，検索で得られた結果の URL のページを実際にクロールして収集し，それらのページの内容を解析することが行われる．提案システムは Web 検索結果として得られるタイトルやスニペットを利用することに特化している．

2 点目は，既存の言語からの言語仕様の拡張を行っているかどうかである．WSQ/DSQ と WebQL はともに SQL を基にした言語であるが，各種機能のために独自に言語仕様の拡張を行っている．4 章で述べた我々のシステムの実装では，Microsoft SQL Server 2005 を利用しているが，そのクエリ言語である Transact-SQL の拡張は行っていない．言語の拡張を行っていないため，既存のデータベースとの共存がより容易であり，また，ユーザが新たな言語仕様を習得する必要はない．KnowItAll は検索システムであるため，本項目は無関係である．

3 点目は自然言語処理機能の有無である．WSQ/DSQ 以外のシステムはいくばくかの自然言語処理機能を持つため，知識として新たな語を発見するようなことが可能であり，我々がいう**語レベルの知識**を取得可能である．WSQ/DSQ では，AltaVista[20] の near オペレータを利用しており，これは語の共起性を厳しく調べることが可能であることから一種の自然言語処理であると考えることができる．しかし，知識として，ユーザが与えた語ではない，新たな語を発見するためには，パターンによる語の抽出など，より高レベルの自然言語処理機能が必要である．また，WSQ/DSQ において取得できる Web 検索結果には，タイトルやスニペットといった文字列情報はそもそも扱われておらず，新たな語を発見するような知識取得を行うことは難しい．WebQL では正規表現によるデータ抽出機能が利用可能であるが，形態素解析を行うようなことはできず，たとえば，5.1 節で示した例のように，形容詞のみを抽出するといったことはできない．提案手法と KnowItAll では，品詞情報を扱うような自然言語処理まで行うことが可能である．

4 点目は，知識取得手法の蓄積が可能であるかという点である．5 章において具体例をあげたが，我々の実装システムでは Web 集約質問処理の一部を変数として，関数の定義を SQL で記述することによって，知識取得手法そのものを蓄積することが可能である．これにより，他ユーザが考案した知識取得手法を利用した新たな知識取得手法を作成することも可能になり，Web 集約質問処理による知識取得の可能性の幅が大きく広がる．他システムには，知識取得手法を関数化するなどして蓄積・利用するための機能はない．

5 点目は，他ユーザによる機能の拡張が可能であるかという点である．WSQ/DSQ と WebQL は独自システムであり，ユーザによる機能拡張を行うことは困難である．KnowItAll は，自然言語によるクエリに特化した検索システムを利用したアプリケーションであるが，KnowItAll の機能拡張を他ユーザが行うことは難しい．我々の実装システムは，Microsoft SQL Server 2005 の SQL CLR という機能を利用して実装しており，我々が用意した機能以外に機能が必要になったときにユーザが拡張を行うことは容易である．たとえば，我々が現在利用している Web 検索エンジン以外の検索エンジンを利用したい場合には，ユーザがその機能を自作することが可能である．

最後に，Java や Perl などの一般のプログラミング言語を利用した場合と，我々の提案システムで SQL を利用する場合の違いについて述べる．最も大きな違いは言語の抽象度の違いである．すなわち，SQL は抽象度が比較的高い宣言型プログラミングであるのに対して，Java はオブジェクト指向プログラミング，Perl は手続き型プログラミングである．宣言型プログラミングでは，どのようなデータを取得したいかを記述するのに対し，手続き型プログラミングでは，どのようにデータを取得するかを記述する．オブジェクト指向プログラミングでも Java などでは，メソッド内部で手続き型プログラミングと同様の記述を行う．この違いの端的な差は，プログラミングの長さに現れ，宣言型プログラミングのほうが短くなる．特に，SQL はデータの集合を扱うのに特化した言語であり，Web 集約質問処理というデータの集合を集約する処理に親和性が大きく，SQL を用いることによってプログラミングが非常に短くなる．Web からの知識取得手法は，誰にでも容易に思いつくことができるものであり，短く書くことによって気軽に試すことができる環境があるのは大きなメリットであるといえる．

表 11 は，簡単な Web 集約質問処理の 1 つを提案システムを利用して行った場合と，オブジェクト指向プログラミング言語 C#を利用して行った場合のプログラムの行数の違いを表している．プログラミング言語 C#では，SlothLib[24),25)] というライブラリを用いることで，Web 検索や自然言語処理などを，ライブ

表 11 提案システムとプログラミング言語での行数の違い
Table 11 Difference of code lines between a programming language and our system.

	提案システム	C#
変数や関数の定義	7	7
Web 検索結果の取得	1	2
自然言語処理	2	10
データ集約	1	3
順位付け	1	6
合計	12	28

ラリを用いない場合よりもはるかに短い行数で行っている．それでもなお，提案システムにおける記述の 2 倍以上の行数が必要となる．変数や関数を定義する部分を除いた実質のプログラムの部分では，提案システムでは 5 行にわたって書かれた SELECT 文 1 つであるのに対して，C#では 21 行と，その差が実質的に大きいものであることが明らかである．

7. ま と め

本稿では，Web 集約質問処理を行うために，関係データベース上で Web 検索や自然言語処理機能を利用する環境を提案した．典型的な Web 集約質問は，「Web 検索」「NLP によるデータ抽出」「情報集約」の 3 つの段階からなる．Microsoft SQL Server 2005 上にテーブル値関数やスカラ値関数として Web 検索や自然言語処理機能を実装した．実装したシステムでのアプリケーション例として，Web からの知識取得の事例を紹介し，また，既存のローカルデータベース上のデータとそれらの知識の統合の例も示した．

謝辞 本研究の一部は，文部科学省科学研究費補助金特定領域研究「情報爆発時代に向けた新しい IT 基盤技術の研究」における計画研究「情報爆発に対応する新 IT 基盤研究支援プラットフォームの構築」（研究代表者：安達淳，Y00-01，課題番号：18049073）ならびに計画研究「情報爆発時代に対応するコンテンツ融合と操作環境融合に関する研究」（研究代表者：田中克己，A01-00-02, 課題番号：18049041），および，文部科学省グローバル COE 拠点形成プログラム「知識循環社会のための情報学教育研究拠点」（研究代表者：田中克己，平成 19〜23 年度），および，文部科学省研究委託事業「知的資産の電子的な保存・活用を支援するソフトウェア技術基盤の構築」，異メディア・アーカイブの横断的検索・統合ソフトウェア開発（研究代表者：田中克己），および，平成 19 年度文部科学省科学研究費補助金若手研究（B）「Web からの履歴情報の発見とその呈示方式の研究」（研究代表者：小山聡，課題番号：19700091），および，平成 19 年度京都大学若手研究者スタートアップ研究費「関係データベース環境における Web からの知識取得環境の実現とその応用」（研究代表者：大島裕明）によるものです．ここに記して謝意を表します．

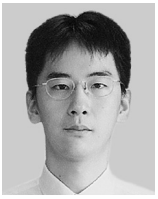
参 考 文 献

1) Wikipedia. http://en.wikipedia.org/
2) Wiktionary. http://en.wiktionary.org/
3) Nakamura, S., Konishi, S., Jatowt, A., Ohshima, H., Kondo, H., Tezuka, T., Oyama, S. and Tanaka, K.: Trustworthiness analysis of web search results, *Proc. 11th European Conference on Research and Advanced Technology for Digital Libraries* (*ECDL 2007*), pp.38–49 (2007).
4) Goldman, R. and Widom, J.: WSQ/DSQ: A practical approach for combined querying of databases and the Web, *Proc. 2000 ACM SIGMOD International Conference on Management of Data* (*SIGMOD 2000*), pp.285–296 (2000).
5) WebQL. http://www.ql2.com/
6) Raman, V. and Hellerstein, J.M.: Partial results for online query processing, *Proc. 2002 ACM SIGMOD International Conference on Management of Data* (*SIGMOD 2002*), pp.275–286 (2002).
7) Cafarella, M.J. and Etzioni, O.: A search engine for natural language applications, *Proc. 14th International Conference on World Wide Web* (*WWW 2005*), pp.442–452 (2005).
8) KnowItAll. http://www.cs.washington.edu/research/knowitall/
9) Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora, *Proc. 14th International Conference on Computational Linguistics* (*COLING 1992*), pp.539–545 (1992).
10) Ohshima, H., Oyama, S. and Tanaka, K.: Searching coordinate terms with their context from the web, *Proc. 7th International Conference on Web Information Systems Engineering* (*WISE 2006*), pp.40–47 (2006).
11) 大島裕明，小山 聡，田中克己：Web 検索エンジンのインデックスを用いた同位語とそのコンテキストの発見，情報処理学会論文誌（トランザクション）データベース，Vol.47, No.SIG19(TOD32), pp.98–112 (2006).
12) Turney, P.D.: Mining the Web for synonyms: PMI-IR versus LSA on TOEFL, *Proc. 12th European Conference on Machine Lerning* (*ECML 2001*), pp.491–502 (2001).
13) Baroni, M. and Bisi, S.: Using cooccurrence statistics and the web to discover synonyms in

a technical language, *Proc. 4th International Conference on Language Resources and Evaluation* (*LREC 2004*), pp.1725–1728 (2004).
14) Oyama, S. and Tanaka, K.: Query modification by discovering topics from web page structures, *Proc. 6th Asia Pacific Web Conference* (*APWeb 2004*), pp.553–564 (2004).
15) Cilibras, R.L. and Vitanyi, P.M.: The Google similarity distance, *IEEE Trans. Knowledge and Data Engineering*, Vol.19, No.3, pp.370–383 (2007).
16) Firat, A., Madnick, S.E., Yahaya, N.A., Kuan, C.W. and Bressan, S.: Information aggregation using the Caméléon# Web wrapper, *Proc. E-Commerce and Web Technologies: 6th International Conference* (*EC-Web 2005*), pp.76–86 (2005).
17) Petropoulos, M., Deutsch, A. and Papakonstantinou, Y.: Interactive query formulation over Web service-accessed sources, *Proc. 2006 ACM SIGMOD International Conference on Management of Data* (*SIGMOD 2006*), pp.253–264 (2006).
18) Oyama, S., Tezuka, T., Ohshima, H. and Tanaka, K.: Processing web aggregate queries by analyzing search engine indices, *Proc. DASFAA 2007 International Workshop on Scalable Web Information Integration and Service* (*SWIIS 2007*) (2007).
19) Google. http://www.google.com/
20) AltaVista. http://www.altavista.com/
21) Microsoft SQL Server. http://www.microsoft.com/sql/
22) Yahoo! Web search Web service. http://developer.yahoo.co.jp/search/web/V1/webSearch.html
23) Tsuruoka, Y. and Tsujii, J.: Bidirectional inference with the easiest-first strategy for tagging sequence data, *Proc. Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing* (*HLT/EMNLP 2005*), pp.467–474 (2005).
24) 大島裕明，中村聡史，田中克己：Slothlib：Web サーチ研究のためのプログラミングライブラリ，日本データベース学会 Letters，Vol.6，No.1，pp.113–116 (2007).
25) SlothLib Web サイト. http://www.dl.kuis.kyoto-u.ac.jp/slothlib/

(平成 19 年 6 月 20 日受付)
(平成 19 年 10 月 8 日採録)

(担当編集委員　関 洋平)



大島　裕明（正会員）
京都大学大学院情報学研究科社会情報学専攻特任助教．2007 年京都大学大学院情報学研究科博士後期課程修了．博士（情報学）．主に Web 検索やパーソナライゼーションの研究に従事．電子情報通信学会，日本データベース学会，ACM 各会員．

小山　　聡（正会員）
京都大学大学院情報学研究科社会情報学専攻助教．2002 年京都大学大学院情報学研究科博士後期課程修了．博士（情報学）．主に機械学習，データマイニング，情報検索の研究に従事．電子情報通信学会，人工知能学会，日本データベース学会，IEEE，ACM，AAAI 各会員．

田中　克己（正会員）
京都大学大学院情報学研究科社会情報学専攻教授．1976 年京都大学大学院修士課程修了．博士（工学）．主にデータベース，マルチメディアコンテンツ処理の研究に従事．IEEE Computer Society，ACM，人工知能学会，日本ソフトウェア科学会，日本データベース学会等各会員．