人文科学とコンピュータ 54－4
（２００２．５．３１）

万葉集データベースにおける和歌検索の効率に関する検討

中田 充　　吉村 誠　　葛 崎偉
山口大学　教育学部

万葉集には多くの写本や注釈書が存在し、和歌本文以外にも和歌の題詞、左注、異訓、異同などの様々な情報がある。万葉集研究においては、これらの情報を含めて和歌を検索可能なシステムが必要とされている。筆者らは、このような和歌検索が可能な万葉集検索システムを実現することを目的として、万葉集データベースとそのグラフィカルユーザインタフェースの構築を行っている。本稿では、万葉集データベースシステムにおける和歌検索の効率について検討する。まず、和歌に関する代表的な問い合わせの実行時間を測定することで和歌検索の効率について検討する。次に、指定された読みを持つ漢字を起点としたる和歌検索の実装方法に関する検討を行う。

On Efficiency of Japanese Poems' Searching in a MANYO-SHU Database System

Mitsuru NAKATA,　Makoto YOSHIMURA,　Qi-Wei Ge
Faculty of Education　Yamaguchi University

There exist many kinds of MANYO-SHU books such as hand-transcribed books, annotated books and revised edition books. Although there are no great differences between these books for a same Japanese poem, subtle differences can often be found in the information. In the research field of Manyo-Shu, it is important to handle these differences and so that we try to design an advanced and rapid database system and a Graphical User Interface. In this paper, we discuss the efficiency of Japanese poems' searching in the MANYO-SHU Database. At first, we will do the performance evaluation by measuring run times of some typical queries. And then, we'll consider a searching method of Japanese poems by KANJI characters that indicate readings.

1　はじめに

万葉集は、8 世紀後半から 9 世紀初頭に編纂されたと考えられる全 20 巻からなる古代の和歌集である。4516 首の歌が載せられており、皇族や貴族を中心としながら庶民の歌までも含むという、後の勅撰和歌集には見られない特徴を持っている。他の多くの古典作品と同様、原本は失われており、書写された写本という形でのみ伝えられている。そのため文字の誤写等による異本が生じており、約 20 種類の写本が現存している。さらに、複数の写本から原本を推定する校訂作業を経て作成された校定本、漢字表記の原文に対して後世に付された訓読をもとにした注釈書などが併せて 30 種類ばかり作られている。これらには、約 3400 箇所にわたる文字の相違（異同）と 3500 首あまりにわたる読みの違い（異訓）が存在する。本稿では、これら様々な万葉集をまとめて諸本と呼ぶ。

万葉集の和歌は、5 音または 7 音からなる句を基本単位として、5 句からなる短歌（約4200 首）、複数の長句からなる長歌（約 300 首）によって構成される（以降、これらを「和歌」と呼ぶ）。これに加え、和歌の右側に、詠作の場や主題や作者などを漢文体で記した「題詞」、和歌の左側に題詞と同様のことを補足的に説明した「左注」が付けられている。和歌は国歌大観番号と称する明治期に付された和歌番号によって整理されており、研究者はこれを用いてそれぞれの和歌を識別している。さらに、万葉集の諸本には、異訓、異同、ルビ、校異などの万葉集研究において重要な情報が存在する。これらを容易に検索可能とするために、検索システムが強く望まれている。なお、電子化にあたっては、異体字も含めた文字異同を発見する機構や、写本に多い本文の修正（ミセケチ）や、挿入、補注などの付随情報も扱える機構なども望まれている。これらは、文献学的に万葉集の研究を行う上で、写本間で従来にない詳細な関連情報をとらえることが可能となるためである。

上記の背景のもと、筆者らはデータベース管理システムを用いた万葉集検索システムを構築している[1]。このシステムは、万葉集の和歌を異訓、異同などを含めて検索可能なシステムである。本稿では、構築中の万葉集データベースシステムにおける和歌検索の効率について検討する。以降、2 節では構築中のデータベースシステムについて述べ、3 節で和歌に関する代表的な問い合わせの実行時間を測定することで、和歌検索の効率について検討する。4 節では、現システムでは実行不可能な「ある読みを持つ漢字による和歌の検索」について、その実装方法に関する検討を行う。5 節はまとめである。

2 万葉集検索システムについて

2.1 求められる機能と設計方針

万葉集検索システムに求められる機能には、以下のような事項がある。

a) 和歌や題詞、左注の異訓、異同を含めた和歌データの検索機能。
検索に際しては、異訓や異同を意識することなく全ての諸本の和歌を検索可能であり、かつ、同じ一つの和歌の異訓、異同を検索可能な機構が求められる。

b) 読み、漢字、漢字仮名交じり文による和歌、題詞、左注（以降、これらを和歌データと呼ぶ）の検索機能。
この際、単に「特定の読みを含む和歌」をだけではなく、「ある読みをする全ての漢字を含む和歌」などを検索する機構が求められる。

c) 題詞、左注と和歌との多対多関係の表現が可能であること。

d) 多様な注の表現とそれらの情報による和歌の検索が可能であること。注とは和歌、題詞、左注に関して、本文に表記されている以外に考えられる原文や読みなどの情報である。

e) データ検索時において利用可能なグラフィカルユーザインタフェース（GUI）。

これらの要求を満たすため、以下のような基本方針のもとに万葉集データを格納し、システムを実現している。

1) データ入力の際に基準となる「底本」を定めずに、全ての本のデータを同一形式で格納する。また、和歌の集合として諸本を表現する。
2) 和歌は句ごとに分解して格納し、全ての検索は句を単位とする。
3) 和歌、題詞、左注は、それぞれ、「原文」、「仮名」、「訓読」、「訓読ルビ付き」の情報を持つ。
4) 和歌、題詞、左注に付加されている様々な注は、部分注と全体注とにまとめて格納する。
5) 和歌データの管理は、データベース管理システム（DBMS）を用い、検索のインターフェースとしての GUI を実現する。

すなわち、万葉集の全ての諸本のデータを本、和歌、句、題詞、左注、全体注、部分注に分けてデータベースに格納する。また、和歌を句に分割して格納することで、諸本の相違を句単位で検索でき、部分的な注と句の関連を明確に表現可能となる。「原文」とは漢字表記の原文、「仮名」はそれに対する読み、「訓読」は漢字仮名混じり文である。「訓読ルビ付き」は、利便性のために「訓読」中の漢字に対して現代の読みを付加した情報である。和歌、題詞、左注に対する注は、全体注あるい

は部分注に分類して格納する。これは注の種類が多様なため、その種類ごとにリレーションを作成することが困難であるためである。和歌データは、オブジェクトリレーショナル DBMS PostgreSQL [2]を用いて管理し、検索用の GUI は Web サーバを用いた Web ベースのシステムとする。これにより、検索システムを Internet 上で公開することを可能とする。

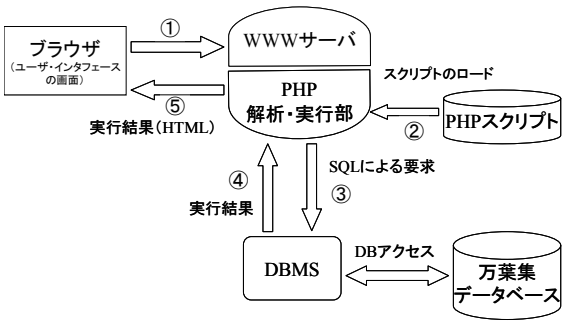


図1：システム構成

2.2 システム構成

システムの実装は、Compaq ProLaint ML530（メモリ:256M、Red Hat Linux6.1J）で行っている。Web サーバが Apache ver.1.3.6、DBMS はオブジェクトリレーショナル DBMS PostgreSQL ver.7.1.1 であり、プログラミング言語として PHP ver.3.0.12 を用いている。PHP [3]はデータベースと組み合わせて使用する言語で、HTML 埋め込み型のサーバサイド・スクリプト言語である。

万葉集検索システムの構成を図 1に示す。ユーザはブラウザ上で検索条件を入力する。入力された条件は Web サーバを通して PHP 解析・実行部に渡される(図中の①)。PHP の解析・実行部は PHP スクリプトを実行することで、ユーザの入力を解析して SQL 文を生成し(②)、その SQL 文を DBMS に発行する(③)。DBMS は万葉集データベースにアクセスし、実行結果を PHP 解析・実行部に渡す(④)。PHP 解析・実行部は、実行結果をもとに生成した HTML ファイルをブラウザに返し、ブラウザはそれを表示する(⑤)。万葉集検索システムは、検索のための PHP スクリプト群と万葉集データベースから構成される。

次に、万葉集データベースのスキーマについて述べ、GUI を用いた簡単な和歌の検索例を挙げる。

2.3 データベーススキーマ

図 2は万葉集データベースのリレーションの一覧である。図中の下線は、各リレーションにて識別子（id）以外でインスタンスを一意に識別するキーとなる属性の集合である。本節では、次節以降に関連のあるリレーション「本」、「和歌」、「句」についてのみ述べる。

(1)は本を表すリレーションであり、本の識別子「本 id」、本の「名前」、「備考」を属性として持つ。

(2)は和歌を表すリレーションである。識別子「和歌 id」、その和歌が含まれる巻の「巻番号」、国歌大観番号に基づいた「和歌番号」、和歌が属する本の「本id」を属性として持つ。本 id を属性に含むため、同じ和歌であっても本が異なればインスタンスが異なる。

(3)のリレーション句は、属性として、「句 id」、和歌における句の順番を示す「番号」、「原文」、「仮名」、「訓読」、「訓読ルビ付」、句が含まれる和歌を示す「和歌 id」を持つ。「和歌 id」を持つために、「和歌」と同様に同じ和歌の同じ句でも本ごとにインスタンスが作成される。さらに、「番号」と「和歌 id」の属性値が両方とも同じインスタンスが存在し得る。

2.4 システムの利用例

ここでは、検索システムの GUI について、簡単な検索例を挙げることで示す。検索例の条件は、「”やまとのくに”と読む句を含む和歌の全ての句の原文、仮名、訓読、訓読ルビ付きを表示する」である。図 3の画面(1)～(6)は、この条件を入力する画面とその検索結果を示した画面である。画面(1)は検索システムの初期画面であり、ここで「仮名」を選択し、「文字・数字の指定」をクリックする（①、②）。すると、画面(2)が表示されるので、「やまとのくに」を入力し、「を含む」を選択し、「ここまでの条件で」をクリックして画面(3)へ進む（③、④、⑤）。画面(3)では、「句」を選択し、「に関する」をクリックする（⑥、⑦）。ここまでの作業で、「“やまとのくに”と読む句」という条件が入力されたことになる。画面(4)では、「和歌」を選択し、「の」をクリックすることで画面(5)が表示される（⑧、⑨）。画面(5)は、検索結果として和歌のどのような情報を表示するかを選ぶ画面であり、この例の検索では、「すべて」をチェックして「検索」ボタンを押す(⑩、⑪)。この時点で、入力された条件をもとに SQL 文が生

(1) 本(本id, 名前, 備考)　　(2) 和歌(和歌id, 巻番号, 和歌番号, 本id)

(3) 句(句id, 番号, 原文, 仮名, 訓読, 訓読ルビ付, 和歌id)

(4) 題詞(題詞id, 至近和歌id, 番号, 原文, 訓読, 訓読ルビ付)

(5) 左注(左注id, 至近和歌id, 番号, 原文, 訓読, 訓読ルビ付)

(6) 全体注(全体注id, 至近和歌id, 名前, データ)

(7) 部分注(部分注id, 至近和歌id, 名前, データ, 開始句番号, 終了句番号, 開始offset, 終了offset)

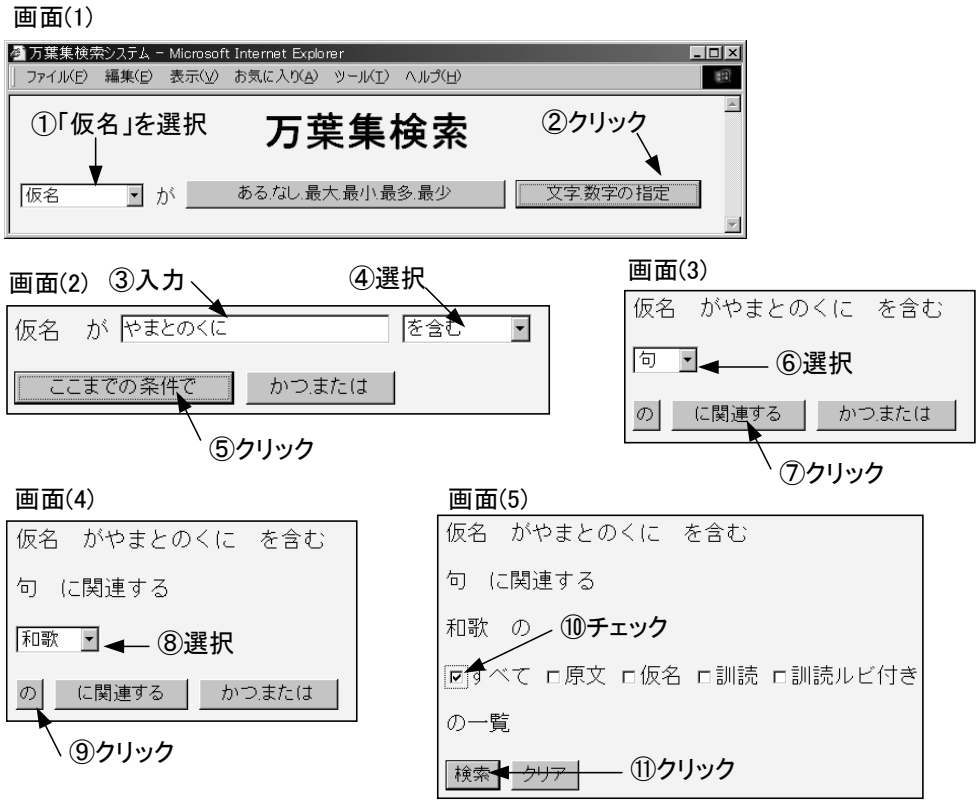
(8) 和歌-題詞関連(題詞id, 和歌id)　　(9) 和歌-左注関連(左注id, 和歌id)

(10) 全体注-和歌関連(全体注id, 和歌id)　　(11) 全体注-題詞関連(全体注id, 題詞id)

(12) 全体注-左注関連(全体注id, 左注id)　　(13) 部分注-句関連(部分注id, 和歌id)

(14) 部分注-題詞関連(部分注id, 題詞id)　　(15) 部分注-左注関連(部分注id, 左注id)

図2：データベーススキーマ



画面(6)

結果検索

表1	本名	和歌番号	巻番号	句1	句2	句3
原文	おうふうダミー1	1	1	籠毛與	美籠母乳	布久思毛
仮名	おうふうダミー1	1	1	こもよ	みこもち	ふくしも
訓読	おうふうダミー1	1	1	籠もよ	み籠持ち	ふくしも
訓読ルビ付き	おうふうダミー1	1	1	籠（こ）もよ	み籠（こ）持（も）ち	ふくしも

図3：検索例

成され、DBMS に対して SQL 文が発行されて検索結果が得られる。得られた検索結果は、画面(6)のように表示される。この例のように、本システムでは、「”やまとのくに”と読む句を含む和歌の全ての句の原文、仮名、訓読、訓読ルビ付きを表示する」などの検索条件を、複数の画面に分けて入力するが、各画面には、これまでに入力した条件が表示されるため、段階的に条件を入力することが可能となっている。

なお、現代においては約 50 種の諸本が存在するが、本システムはそのうち 40 種類程度をデータベース化の対象としており、現状は 40 種のうちの 3 つの本（西本願寺本[4]、おうふう社の万葉集[5]（以降、おうふう本）、万葉集全注[6]）の第 1 巻のみがデータベースに格納されている。和歌数にして 252 首（1 冊あたり 84 首）、句の数は一首が 5 句からなるとして約 1260 である。1 冊の万葉集につき約4500 首であるので、最終的に約18 万首、句の数は 90 万程度になる。また、GUI を用いての検索は、上記の例のような単純な検索のみが実行可能な状態である。

3 和歌検索の効率についての検討

本節では、和歌に関する代表的な問い合せを実行して、その実行時間を測定することで、和歌検索の効率について検討する。なお、実際の測定にあたっては、入力済みの 252 首では和歌数が少なすぎるため、これらをコピーしてダミーのデータを生成し、25,200 首の和歌に対して計測を行う。具体的には、3 冊の本の第 1 巻のデータをコピーし、西本願寺本 1, 西本願寺本2, …, 西本願寺本100というように、それぞれ 100 個のコピーを作成し、これらに対して同じ検索を 5 回行い、その平均を求める。なお、実行時間は SQL 文の実行前と実行後の時刻の差分とする。

和歌に関する代表的な問い合わせとして、以下の問い合わせが考えられる。

問い合せ 1：ある読みを含む句を持つ和歌の検索。

- 例 1:「やまとのくに」を仮名に含む句と「とりよろふ」を仮名に含む句の両方を含む和歌の検索。

問い合せ 2：「ある読みを含む句と同じ原文を持つ句」を含む和歌の検索。

- 例 2-1:「“やまと”と読む句の原文」を含む、全ての句の句番号と原文と読み、その句が含まれる和歌の和歌番号と本の名前の検索。
- 例 2-2:「“やまと”と読む句の原文を含む句」を含む和歌の和歌番号と本の名前と、その和歌の全ての句の句番号と原文と仮名の検索。
- 例 2-3:「“やまと”と読む句の原文を含む句」と「“とりよろふ”と読む句の原文を含む句」の両方を含む和歌の和歌番号と本の名前と、その和歌の全ての句の句番号と原文と仮名の検索。

問い合せ 1 は、「読みによる和歌の検索」の一般的な例として挙げている。問い合せ 1 の例 1 の SQL 文は、リスト 1のように 2 種類考えられる。リスト中ではリレーション名や属性名はすべてローマ字で表記する。SQL 文 1-1 は以下のように考えられる。

1) ”とりよろふ”を仮名に含む句を検索し、その句が含まれる和歌 id を得る。
2) ”やまとのくに”を仮名に含み、かつ、和歌 id に 1)で得た和歌 id を持つ句を得る。この和歌 id が所望の和歌の id である。
3) 2)で得た和歌 id をもとに、その和歌の和歌番号、和歌に含まれるすべての句の原文、仮名などを得る。一番外側の副問い合せは、和歌を句単位に分けて格納しているために必要となる。

同様に、SQL 文 1-2 は以下のように考えられる。

1) “やまとのくに”を仮名に含む句の和歌 id の集合と“とりよろふ”を含む句の和歌 id の集合を求める。
2) 和歌 id が 1)の両方の集合に含まれる句を得る。
3) 2)で得た和歌 id をもとに、その和歌の和歌番号、和歌に含まれるすべての句の原文、仮名などを得る。

これらの SQL 文は同じ結果を返すが、読みに関する条件を複数個指定する際に、論理和(and)を用いるか、副問い合せを用いるかという点が異なる。こ

れらの実行時間を表 1（SQL 文 1-1, 1-2 の列）に示す。表より、SQL 文 1-1 の方がより高速に結果を得ることがわかり、問い合せ 1 において、複数の条件を指定する際には、GUI に入力された条件を副問い合せでつなぎ合わせる必要があることがわかる。

問い合せ 2 は「ある読みを持つ句」を検索し、その句と同じ原文を持つ句を含む和歌を検索する問い合せである。例えば、「“やまと”と読む句」として、“大和国野”、“山都庭”、“倭”、“山戸乃国”などがある場合、これらと同じ原文を含む句を持つ和歌を検索する。この種類の問い合せは、上記の原文が「やまと」と読まれない和歌も検索されるため、「ある読みの句が、他の和歌や本ではどのように読まれているか」を求める、異訓を対象とした検索において多用される。

問い合せ 2 の例 2-1 は、「“やまと”と読む句の原文を含む句」を検索し、例 2-2 はそのような句を含む和歌を検索する。例 2-3 は例 2-2 における条件を複数個指定した問い合せである。これらの問い合せを行う SQL 文はリスト 2に示したとおりであり、その実行時間は表 1（SQL 文 2-1～2-3 の列）が示すとおりである。SQL 文 2-1 と 2-2 の結果の比較より「”やまと”と読む句の原文を含む句」を検索する処理よりも、そのような句を含む和歌のすべての句を検索する処理（リスト 2 中の例 3-2 の下線部分）に時間がかかっていると考えられる。なお、SQL 文 2-3、2-2 よりも実行時間が短いのは、条件が複数指定されることにより、副問い合せにおいてマッチする句が少なくなるためであると考えられる。これらのことから、本システムでは、和歌を句ごとに分割して格納しているが、当該の和歌 id を得てから対応する全ての句を効率よく検索可能な機構を導入することで、全体の検索実行時間の短縮が図ることが可能であると思われる。問い合せ 1, 2 の実行時間のグラフを図 5に示す。グラフより、異訓の検索に用いられる問い合せ 2 の実行時間は、問い合せ 1 と比較して極端に時間がかかるものではないことがわかる。但し、問い合せの条件によっては、SQL2-2 のように時間がかかることがあるため、この点を改善する必要がある。

4　指定された読みを持つ漢字による和歌の検索方式に関する検討

3 節に示したように、本システムでは、「ある読みを含む句と同じ原文を持つ句」を含む和歌の検索を行える。これにより、「ある読みの句が、他の和歌や本ではどのように読まれているか」を調べる異訓を対象とした検索が可能となるが、この機構は不十分である。例えば、“やまと”という仮名を含む句の原文として、“大和国野”、“山都庭”、“倭”、“山戸乃国”があった場合に、これらと全く同じ原文を含む句は検索することができるが、“やまと”とよむであろう“山都”、“山戸”を部分的に含む句は検索されない。これは、システムが句を最小単位として和歌のデータを格納しているためである。異訓を対象とした検索は、句単位ではなく漢字単位でなければ実用性が低い。つまり、「ある読みの漢字が、他の和歌や本ではどのように読まれているか」という問い合せが実行可能であることが要求される。

この要求を満たすために、和歌を句よりも小さい基本単位に分割して格納することも考えられるが、この場合は和歌の検索効率が非常に悪くなることが容易に想像できる。また、ひとまとまりの漢字を基本単位とすると、和歌における異訓、異同、校異（校訂の違い）を検索することが困難となるという問題も生じる。

そこで、和歌の格納方式はそのままに、句の原文から、検索条件の読みの長さを基準として n-gram [7] の漢字文字列を生成し、その文字列を用いて検索を行う方式を考える。

n-gram 文字列を用いた検索

1)　検索条件に指定された仮名文字列を $string_a$ とする。

2)　検索条件の読みの長さを m、漢字 1 文字に対して付されている仮名の最大文字数を K_{max} とする。

3)　句の原文中の先頭から x+1 文字目～後から y+1 文字目までの文字列 $string_b$ を対象に n-gram 文字列の集合 $STRING_N$ を生成する。ここで、生成される n-gram 文字列の gram 数は、$\lceil m/K_{max} \rceil$, $\lceil m/K_{max} \rceil + 1, \ldots, m$ であり、$(x = \lceil a/K_{max} \rceil \mid a$ は句

の仮名の先頭から $string_a$ までの文字数)、
($y = \lceil c/K_{max} \rceil$ | c は仮名の中の $string_a$ 以降の文字数)である。

4) $STRING_N$ から一つの文字列を取り出し、その文字列を原文に含む句を持つ和歌を検索する。

5) $STRING_N$ のすべての要素に対して 4)を実行したら終了。

□

図 4に $K_{max} = 3$ として、「“やまと”と読む漢字を含む和歌」を n-gram 文字列を用いて検索する例を示す。図中の①は 1 文字目の仮名あるいは漢字を表す。この例では、$string_a$ = “やまと”、m = 3, a = 4, c = 3, x = 2, y = 1, $string_b$ = “③大和⑥⑦⑧⑨”である。従って、$STRING_N$ = {“③”、“大”、“和”、“⑥”、“⑦”、“⑧”、“⑩”、“③大”、“大和”、“和⑥”、“⑥⑦”、“⑦⑧”、“⑧⑨”、“③大和”、“大和⑥”、“和⑥⑦”、“⑥⑦⑧”、“⑦⑧⑨”}となり、これら $STRING_N$ 中の要素を“やまと”と読む漢字の候補として、これらの漢字を原文に含む句を持つ和歌を検索する。

この検索方式により、指定した読みを持つ漢字による検索が可能となるが、以下の様な問題点がある。

- 指定された読みを持つ漢字の候補($STRING_N$)が大量に発生するため、検索に時間がかかる。
- 指定された読みを持つ漢字の候補の中に、誤った漢字が含まれる可能性がある。

これらの問題点を解決しつつ、本方式の詳細を検討することは今後の課題である。

5 まとめ

本稿では、万葉集データベースシステムにおける和歌検索の効率について検討した。その結果、条件によっては実用的な時間で検索が可能なものの、所望の和歌を得た後に、和歌に含まれる全ての句を検索する時点で時間がかかっている。これは、当該の和歌 id を得てから対応する全ての句を効率よく検索可能な機構を導入することで、全体の検索実行時間の短縮が図ることが可能であると思われる。次に、指定された読みを持つ漢字による和歌の検索方式に関する検討を行い、n-gram 文字列を用いる検索方式を提案した。検索時間の短縮、n-gram 文字列を用いた検索方式の詳細の検討、GUI の完成などが今後の課題である。

SQL 文 1-1
```
select hon.name, waka.waka_no, ku.genbun,
ku.kana, ku.kundoku from hon, ku, waka where
hon.oid=waka.hon_idandku.waka_id=waka.oid
and waka.oid in (select distinct waka_id from
ku where ku.kana ~ 'やまとのくに' and ku.waka_id
in (select distinct ku.waka_id from ku where
ku.kana ~ 'とりよろふ'));
```
SQL 文 1-2
```
select hon.name, waka.waka_no, ku.genbun,
ku.kana, ku.kundoku from hon, ku, waka where
hon.oid=waka.hon_idandku.waka_id=waka.oid
and waka.oid in (select distinct waka_id from
ku where ku.kana ~ 'やまとのくに') and ku.waka_id
in (select distinct ku.waka_id from ku where
ku.kana ~ 'とりよろふ');
```
リスト1：問い合せ 1 の SQL 文

SQL 文 2-1
```
select hon.name, waka.waka_no, ku.ku_no,
ku.genbun, ku.kana  from hon, waka, ku where
hon.oid=waka.hon_idandku.waka_id=waka.oid
and ku.genbun in (select distinct ku.genbun
from ku where ku.kana ~ 'やまと');
```
SQL 文 2-2
```
select hon.name, waka.waka_no, ku.ku_no,
ku.genbun, ku.kana  from hon, waka, ku where
hon.oid=waka.hon_idandku.waka_id=waka.oid
and waka.oid in (select distinct ku.waka_id
from ku where ku.genbun in (select distinct
ku.genbun from ku where ku.kana ~ 'やまと'));
```
SQL 文 2-3
```
select hon.name, waka.waka_no, ku.ku_no,
ku.genbun, ku.kana  from hon, waka, ku where
hon.oid=waka.hon_idandku.waka_id=waka.oid
and waka.oid in (select distinct ku.waka_id
from ku where ku.genbun in (select distinct
ku.genbun from ku where ku.kana ~ 'やまと') and
ku.waka_id in (select distinct ku.waka_id from
ku where ku.genbun in (select distinct
ku.genbun from ku where ku.kana ~ 'とりよろふ
')));
```
リスト2：問い合せ 2 の SQL 文

表1：問い合せ 1, 2 の実行時間

和歌数	検索時間(s)				
	SQL文 1-1	SQL文 1-2	SQL文 2-1	SQL文 2-2	SQL文 2-3
5040	1.2	4.2	1.4	4.8	2.3
10080	2.9	15.3	2.7	15.5	5.3
15120	5.1	33.2	4.1	32.1	8.9
20160	7.9	57.9	5.4	57.1	13.2
25200	11.2	89.4	6.8	95.3	18.1

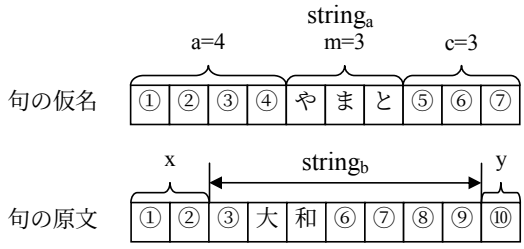


図4：n-gram 文字列を用いた検索の例

参考文献

[1] 中田充、吉村誠、葛崎偉:“データベース管理システムを用いた万葉集データベースの一構成法”、情報処理学会研究報告 Vol2001, No96、pp.9-16、2001.

[2] 石井達夫:“PostgreSQL 完全攻略ガイド”、技術評論社、2001.

[3] 堀田倫英 石井達夫 広川類:PHP 徹底攻略 Web とデータベースの連携プログラミング、SOFT BANK

[4] 書写者未詳、“西本願寺本”、14 世紀ごろ書写.

[5] 鶴 久、森山隆:“萬葉集”、おうふう、1972.

[6] 伊藤博:“万葉集全注”、有斐閣、1983.

[7] “N-gram が開く世界”、漢字文献情報処理研究第二号 特集 2、pp49-73、好文出版.

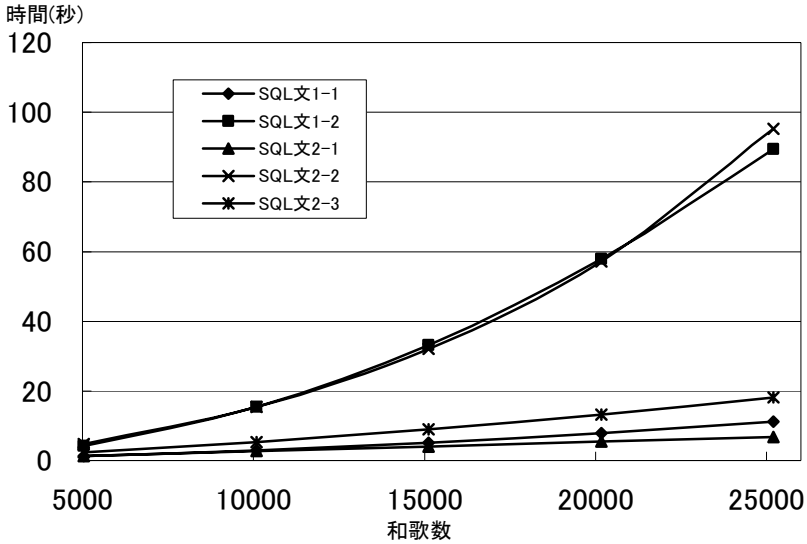


図5：問い合せ 1, 2 の実行時間のグラフ