図書館情報学 修士論文

Z39.50に基づく書誌データ検索システムの構築

高久 雅生

2000年3月

内容梗概

Z39.50 は情報検索のための通信プロトコルであり、現在欧米を中心に普及が進んでいる。その特徴は複数の異なる検索システム間の相互利用性を考慮し、多種多様なデータベースを透過的に検索できる点である。

一方、WWW（World Wide Web）がインターネット上の情報共有システムとして広く普及し、大量の情報が提供されるようになってきた。その結果、これらの情報を検索する WWW 上のサーチエンジンも現れた。しかし、サーチエンジンによる検索はノイズが多く、利用者が欲しい情報を見つけるのは依然として難しい状況である。そのため、WWW に限らず電子媒体上の情報資源に関するデータ、すなわちメタデータを記述することで、質の高い情報検索が可能になるという考え方が出てきた。

そこで本研究では、実験システムとして次の特徴を持つ Z39.50 情報検索システムを開発した。

1. 本システムの検索エンジンは Z39.50 と WWW の両者で利用可能であり、情報資源を共有できる。
2. Z39.50 の持つ相互利用性の一層の向上と情報資源の共有化を目指し、メタデータの枠組みである Dublin Core や RDF（Resource Description Framework）を組み込んだ。
3. 日本語書誌データとして代表的な JAPAN/MARC 約 102 万件の大量データを蓄積し、検索できる。

具体的には、JAPAN/MARC 検索システム（以下、Z39.50-JP システムと呼ぶ）および Dublin Core メタデータ検索システム（以下、Z39.50-DC システムと呼ぶ）の 2 つの Z39.50 検索システムを構築した。Z39.50-JP システムは上記の 1 と 3 の特徴を持ち、Z39.50-DC システムは 1 と 2 の特徴を持つ。

まず、Z39.50-JP システムでは日本語書誌データ JAPAN/MARC を対象として、約 102 万件の大規模データを検索できるシステムを構築した。また、全文検索エンジン Namazu の改造によって情報資源の共有化を実現した。検索エンジン Namazu は元々WWW 用の全文検索エンジンであるため、本システムの情報を Z39.50 でも WWW でも提供できるとともに、既存の Namazu を採用している WWW 上の検索サービスも本研究で開発した Z39.50 サーバを組み込むことでそのまま Z39.50 でも提供できるなど、情報資源の共有が可能になった。

次に、この Z39.50-JP システム構築の過程でも問題となった Z39.50 におけるアトリビュートセットのマッピング問題に着目し、この問題を解決することを目的に Z39.50-DC システムを構築した。この Z39.50-DC システムは、Z39.50-JP システムとは別に新たに構築したもので、Z39.50 の持つ相互利用性をさらに向上させることを目的としたものである。Z39.50-DC システムでは特に、Dublin Core メタデータを共通スキーマとして利用することで、このアトリビュートセットのマッピング問題を解決することを目指した。

アトリビュートセットとは、Z39.50 で定義されているデータベースの実装に依存しない論理的なスキーマである。しかしながら、Z39.50 では複数のアトリビュートセットを使うと透過的な検索が妨げられるため、結果として大部分の Z39.50 サーバが Bib-1 というひとつのアトリビュートセットを使っている。ところが、実際の検索項目を Bib-1 にマッピングする際のスキーマが定められていないため、このマッピングがシステム依存になってしまっており、異なる検索システムの間では同じフィールドが別のマッピングをしている可能性があるなど検索処理の一貫性が保たれていないという問題点が既に指摘されている。また、このデータ間のスキーマの違いを吸収する枠組みとして Dublin Core メタデータが適用可能であると指摘が既になされている。

Z39.50-DC システムでは JAPAN/MARC を対象として Dublin Core メタデータへの変換を行ない、Z39.50 の Bib-1 の Dublin Core 用アクセスポイントを通じた検索ができるシステムを構築した。JAPAN/MARC だけでなく他のデータについても Dublin Core を共通スキーマにすれば本システムを利用して Z39.50 で検索できる。

これらのシステムの構築を通じて、大規模な Z39.50 サーバの構築、WWW 上の情報資源の共有、メタデータの利用、日本語検索の問題点の解決といった実験システムとして充実したシステムが構築できた。なお、日本語の構造による問題、海外サーバを含む他のサーバとの相互運用性、レコードシンタックスおよびセマンティックス、フルテキストなど書誌データ以外の処理機能など Z39.50 における問題は数多く残っているが、本研究で用いた Z39.50 の実験システム構築という手法はこれらの問題の解決にも有効であると考えられる。

SYNOPSIS

Z39.50 is a communication protocol for information retrieval and is now spreading in USA and Europe. Features of Z39.50 protocol are interoperability among different information retrieval systems and ability to search various heterogeneous databases transparently.

Meanwhile WWW (World Wide Web) spreads widely as an information sharing system on Internet. Huge information has been provided on WWW. As a result, search engines on WWW was developed to retrieve the huge information. However it is still difficult for users to find desired information because the search engine gives users many noises. Then, concept of metadata has appeared. Metadata made higher quality information retrieval possible by describing data about information resources in electronic media including WWW pages.

I developed Z39.50 information retrieval system as an experimental system. This system contains the following features;

1. The search sub-system in the system can be used with both of Z39.50 and WWW. This architecture is useful to share information resources.
2. The system used Dublin Core and RDF (Resource Description Framework), frameworks for describing and expressing metadata, with the aim of more interoperability of Z39.50 and sharing of the information resources.
3. The system stored and retrieved about 1020 thousands huge data of JAPAN/MARC that is a typical Japanese bibliographic data.

Two Z39.50 information retrieval systems are constructed. One is the JAPAN/MARC retrieval system (Z39.50-JP). The other is the Dublin Core metadata retrieval system (Z39.50-DC). Z39.50-JP has features No.1 and No.3 mentioned above. Z39.50-DC has features No.1 and No.2.

Z39.50-JP can retrieve about 1020 thousands JAPAN/MARC. Namazu, a full-text retrieval engine, is extended to Z39.50-JP for sharing of information resources. Namazu was developed for WWW originally. Then, bibliographic data in this system can be provided through Z39.50 and WWW. Any existing search services by Namazu on WWW can be also provided through Z39.50 by using Z39.50 server developed in this study.

Z39.50-DC was developed for solution of the mapping problem of the attribute set in Z39.50 that had been pointed out and was pointed out in this study also. Z39.50-DC used Dublin Core metadata as a common schema to solve the mapping problem and to improve interoperability of Z39.50.

Attribute set defined in Z39.50 is the logical schema which is not dependent on database implementations. Z39.50 defines several attribute sets. But most of Z39.50 servers use only the Bib-1 attribute set, because transparency among databases decreases when more than one attribute set is used. However, the schema that is used in mapping fields of search record to Bib-1, is not determined. The mapping is dependent on a system. The same field may or may not be mapped to different Bib-1 attribute in different systems. Such lack of consistency of search implementations has been pointed out. This is the mapping problem. It has been also pointed out that Dublin Core metadata might be applicable as a framework to absorb the mapping problem.

In Z39.50-DC, JAPAN/MARC record was converted into Dublin Core metadata. Data fields of JAPAN/MARC was mapped to corresponding elements of Dublin Core. Users can retrieve through Dublin Core access points in Bib-1. This method was applicable to other databases.

I successfully studied constructions of huge Z39.50 server, sharing of information resources on WWW, use of metadata, and solution for retrieval of Japanese data by means of constructing experimental systems mentioned before. This approach will be effective to solve remaining problems concerning to Z39.50: interoperability among servers including overseas, record syntax, semantics, and full-text.

目 次

図 目 次

表 目 次

第1章　はじめに

1970年代からDIALOGやORBITなどがデータベースベンダとして多くの商用データベースを抱え、オンライン検索システムのサービスを開始した。時同じくしてアメリカ議会図書館、OCLC、RLINなどが機械可読目録（MARC）のデータベースを構築し、書誌ユーティリティとしての機能を果たし始めた。このように、オンライン情報検索システムが幅広く利用できるようになるにつれ、各システム間の接続方法や検索コマンド、検索式の構造、アクセスポイント指定方法などが異なるために、複数のデータベースの利用が現実には困難になるという問題が指摘されるようになっていった[1][2]。

これらの問題を解決するために登場したのがZ39.50プロトコル[3]である。Z39.50の特徴は複数の異なる検索システム間の相互利用性を考慮し、多種多様なデータベースを透過的に検索できる点にある。また、ステートフルな接続機能を持つこと、すなわち一回一回の検索にかかわる情報を保持することによって、旧来の情報検索システムで実現されていた履歴検索機能をもっている点も特徴である。

Z39.50の歴史は1979年の主要な書誌ユーティリティ間で目録データを交換し、互いのデータを透過的に検索できることを目指したプロジェクトの開始にさかのぼることができる。その後、OSIの規格と平行してANSIの構成団体であるNISOで議論され、1988年に標準情報検索プロトコルZ39.50として規格が制定された。その後、1990年代に入り、TCP/IPをベースとするインターネットの普及にあわせ、Z39.50の規格はOSIベースの1992年のVersion2からTCP/IPへの対応を念頭においた1995年のVersion3へと発展した。この時期に、アメリカ議会図書館、OCLCをはじめ、アメリカ、ヨーロッパで図書館システムを中心に数多くのZ39.50検索システムが実装されるようになった[4][5][6]。例えば、2000年1月現在、Index Data社が作成しているZ39.50サーバリスト[7]だけでも243サーバが登録されている。また、日本でも欧米の動きに呼応するように、1999年にはJIS規格[8]として採用され、実装の側でも安齋によるZ39.50システムの試作[9][10]、早稲田大学のWINE[11]、図書館情報大学ディジタル図書館[12][13]、東京工業大学電子図書館[14][15]などで相次いでZ39.50システムが構築されるようになってきた。

そこで本研究ではZ39.50に基づいた検索システムを実験システムとして構築し、Z39.50で依然として残されている日本語検索問題の解決、Z39.50の特徴である相互利用性の向上、情報資源共有のための基盤構築を目指す。このための実験システムとして、Z39.50に基づく日本語書誌データJAPAN/MARC検索システム（以下、Z39.50-JPシステムと呼ぶ）およびDublin Coreメタデータ検索システム（以下、Z39.50-DCシステムと呼ぶ）の2種類のZ39.50検索システムを構築した。なお、Z39.50-JPシステムとZ39.50-DCシステムについてはそれぞれその一部を論文として発表した[16][17]。

まず、Z39.50-JPシステムでは、日本語書誌データとして代表的なJAPAN/MARC[18][19]を対

象として、約 102 万件の大規模データを検索できるシステムを構築し、さらに、全文検索エンジン Namazu[20] の改造によって既存の情報資源の共有につながる Z39.50 検索システムを構築した。

次いで、この Z39.50-JP システム構築の過程でも大きく問題となった、Z39.50 におけるアトリビュートセットのマッピング問題に着目し、この問題を解決することを目的に Z39.50-DC システムを構築した。この Z39.50-DC システムは、今述べた Z39.50-JP システムとは別に新たに構築したもので、Z39.50 の持つ相互利用性をさらに向上させることを目的としたものである。

アトリビュートセットとは、Z39.50 で定義されているデータベースの実装に依存しない論理的なスキーマである。しかしながら、Z39.50 では複数のアトリビュートセットを使うと透過的な検索が妨げられるため、結果として大部分の Z39.50 サーバが Bib-1 というひとつのアトリビュートセットを使っている。ところが、実際の検索項目を Bib-1 にマッピングする際のスキーマが定められていないため、このマッピングがシステム依存になってしまっており、異なる検索システムの間では同じフィールドが別のマッピングをしている可能性があるなど検索処理の一貫性が保たれていないという問題点が指摘されている [21]。また、このデータ間のスキーマの違いを吸収する枠組として Dublin Core Metadata Element Set（以下、Dublin Core と呼ぶ）[22] が適用可能であると指摘されている [1]。

Z39.50-DC システムでは特に、Dublin Core や RDF（Resource Description Framework）[23][24] などのメタデータの枠組を Z39.50 検索システムで利用することで、このアトリビュートセットのマッピング問題を解決することを目指している。具体的には、JAPAN/MARC データを例にとり Dublin Core を共通スキーマとしてマッピングを行ない、Z39.50 の Bib-1 における Dublin Core 用のアクセスポイントを通して JAPAN/MARC を検索できるシステムを構築した。本システムにより JAPAN/MARC だけでなく、その他の書誌データについても Dublin Core を共通スキーマとした Z39.50 検索システムを容易に構築できるようになる。

本論文の構成は以下の通りである。2 章では Z39.50-JP システムについて、システムの位置づけ、システムの構成や機能の詳細について述べる。3 章では、同じく Z39.50-DC システムについて、システムの位置づけ、システムの構成や機能について詳しく述べる。4 章では、上記 Z39.50-JP システムと Z39.50-DC システムを比較検討し、その検討を元にして、本研究の意義と今後の課題について考察する。最後に、5 章では本研究の成果をまとめる。

第2章　Z39.50-JPシステム[16]

2.1　本システムの位置づけ

Z39.50[3] はクライアント・サーバ環境において、クライアントとサーバが通信する際のデータ構造やデータ変換の処理方式を規定したプロトコルである。すなわち、システム内部でのデータ構造や検索処理がシステムごとに異なっていても、通信する際に同じ構造と処理方式になるので、それぞれのシステムを透過的に利用できる。

1 章で述べたように、Z39.50 は現在欧米では図書館システムの OPAC などで普及が進んでいる。また、日本においてもここ 1・2 年の間に普及が進み、次世代分散型検索システムの基盤として注目を集めている。

一方、インターネット上では 1994 年頃から WWW（World Wide Web）と呼ばれる情報共有のための分散型情報システムが爆発的に普及を始めた。WWW はもともとは研究者間の研究情報を共有するという目的で開発されたブラウジング指向のシステムであり、WWW ブラウザ上のハイパーリンクをたどることによって、関連情報を次から次に見ることができる。ハイパーリンクは HTML（HyperText Markup Language）と呼ばれるマークアップ言語によって記述し、HTML ファイルは WWW 用の転送プロトコル HTTP（HyperText Transfer Protocol）によって転送される。WWW 上で、情報提供が活発になると同時に、それらの情報を検索するためのサーチエンジンも多く出現してきた [25]。

しかし、これらのサーチエンジンは互いに検索式の指定方法などが異なるため、複数のサーチエンジンを同時に使うことは難しい。また複雑な検索を支援する機能も乏しい。これらは Z39.50 で検索サービスを提供すれば解決できる問題である。一方で、WWW 上での検索システムでは、簡単な検索を行なう分には検索結果のブラウジングが容易にできるため、検索語の絞り込みが容易になるなど、Z39.50 に比べて有利な面も併せもつ。

このような状況の下、本研究では Z39.50 と WWW の両者に対応しうるシステムを開発した。本システムで採用した検索エンジン Namazu は、元々WWW 用の全文検索エンジンであるため、本システムの情報を Z39.50 でも WWW でも提供できるとともに、既存の Namazu を採用している WWW 上の検索サービスも本研究で開発した Z39.50 サーバを組み込むことでそのまま Z39.50 でも提供できるなど情報資源の共有も可能になる。

また、本システム構築を開始した 1999 年初めには、まだ本格的な Z39.50 検索システムの運用は報告されていなかった。そこで本システムは、Z39.50 システムの構築技術とその運用に関する実験システムという位置づけの下で構築され、大規模データへの対応、日本語検索の問題の解決を目指し、日本語書誌データとして代表的な JAPAN/MARC[18][19] を対象とした約 102 万件の大規模データを検索できるシステムを構築した。

2.2 システムの構成

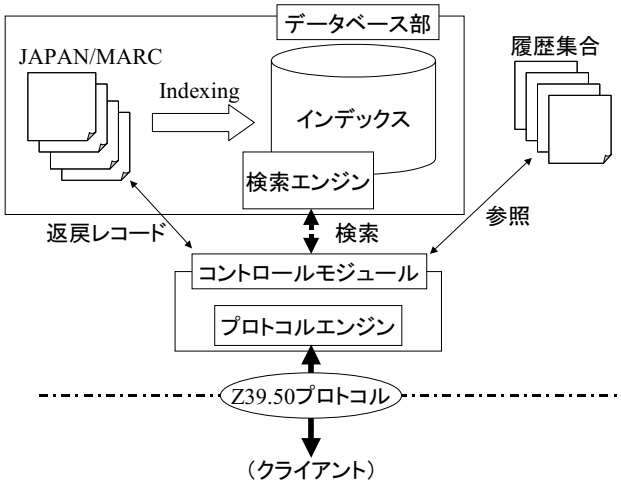

図 2.1: システムの構成

本システムでは日本語書誌データベースを検索するために Z39.50 に準拠したサーバを UNIX ワークステーション Sun Enterprise 3000 上で開発した（図 2.1 参照）。サーバは JAPAN/MARC をデータとして持ち、Z39.50 に基づいたクライアントからの検索要求に応える。

Z39.50 のクライアントとサーバは APDU（Application Protocol Data Unit）と呼ばれる情報フレームをそれぞれのプロトコルモジュール間で通信することで検索セッションを維持する [26]。

本サーバはコントロールモジュール、プロトコルエンジン、検索エンジン、データベース部から構成され、データベース部はさらにインデックスと JAPAN/MARC レコードから構成される。要求は全てクライアントから APDU 形式で送られ、サーバは受け取った APDU をプロトコルエンジンで解析しコントロールモジュールに渡す。コントロールモジュールはクライアントからの要求が検索要求であれば検索エンジンを呼び出す。検索エンジンはデータベースを検索し、検索結果をコントロールモジュールに返す。コントロールモジュールは検索結果を履歴集合として保存し、ヒット件数をプロトコルエンジンに渡す。プロトコルエンジンはヒット件数を APDU としてクライアントに返す。また、コントロールモジュールへのクライアントからの要求が返戻要求の場合、履歴集合として保存してある検索結果を元に JAPAN/MARC レコードを取り出し、そのレコードを SUTRS（Simple Unstructured Text Record Syntax）形式でクライアントに返戻する。

本システムは Index Data 社が開発した Z39.50 システム構築用ツールキット YAZ（Yet Another Z39.50 Toolkit）[27] を利用した。YAZ は C 言語で書かれたプログラム群から構成され、Z39.50 の定義するデータ構造を忠実に再現している。また、サーバの検索エンジンには全文検索エンジン Namazu[20][28] を用いた。

2.2.1 JAPAN/MARC の処理

この節では、JAPAN/MARC データを全文検索エンジン Namazu で利用できるように変換し、インデックス化するまでの処理について述べる。

なお全文検索エンジン Namazu は、フリーで利用可能な WWW 上の検索エンジンとして広く使われているが、本システムでは、これを JAPAN/MARC のタグ・フィールドに対応するように改造し、Z39.50 サーバで利用可能とした。

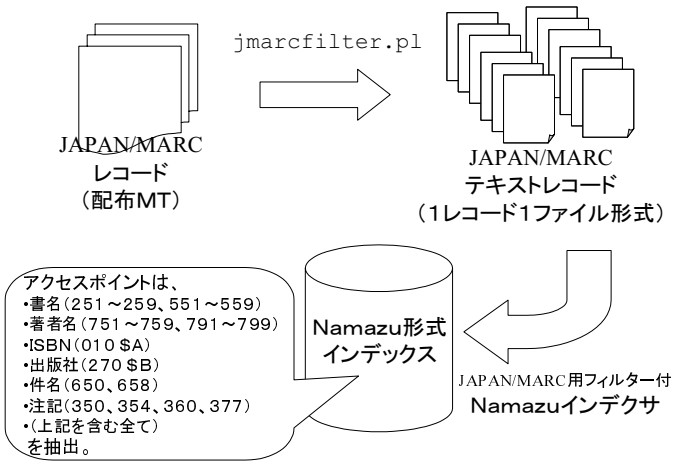

図 2.2: JAPAN/MARC の変換

本システムは 1983 年～1998 年 5 月までの本学購入分 1,019,696 レコードをデータとして持つ。国立国会図書館から配布された JAPAN/MARC レコードのデータは Perl プログラム（付録 A.3 参照）で 1 レコード・1 ファイル形式に変換した。このレコードファイルから JAPAN/MARC のタグ・フィールドの情報 [18][19] をもとに、ISBN、書名、著者名、件名、出版者、注記ごとのフィールド用インデックスと全アクセスポイントを対象にした全文インデックスを作成した（図 2.2 参照）。この際、日本語のテキストデータは日本語形態素解析ツール・茶筌 [29] でわかち書きを行なったものを Namazu のインデックスに登録しておく。実際に、UNIX ワークステーション Sun Enterprise 3000 でインデクシングした際の所要時間は 14 時間 25 分 16 秒で、茶筌で切り出したキーワード数は 4,414,062 語であった。

Z39.50 以前の検索システムではシステム毎に検索質問の構造が異なり、複数のデータベースを利用する際の障害となっていた。Z39.50 ではこの問題に対して、検索質問の包括的なスキーマを定義し、実装の際にこのスキーマを実際のアクセスポイントにマッピングすることで対処している。書誌情報に対するスキーマは 2 つの数字の組み合わせで表される Bib-1 アトリビュートセット [30] で定義されている。例えば、Bib-1 を使えば、書名・著者名・前方一致・後方一致・フレーズなどの検索質問の特性を特定できる。本システムではこのアトリビュートセット Bib-1 を用いた。Bib-1 と JAPAN/MARC でのタグ・フィールドとの対応関係を表 2.1 に示す。

表 2.1: Bib-1 と JAPAN/MARC の対応表

Bib-1（USE Attribute）	JAPAN/MARC タグ	JAPAN/MARC 説明
ISBN（1=7）	010$A	: 国際標準図書番号（ISBN）
Title（1=4）	251 〜 259	: 記述フィールド
	$A	: 書名
	$B	: 副書名
	$D	: 巻次等
	551 〜 559	: 書名アクセスポイント
	$A	: カタカナ形
	$X	: ローマ字形
	$D	: 巻次の読み
Author（1=1003）	751 〜 759	: 著者名アクセスポイント
	$A	: カタカナ形
	$X	: ローマ字形
	$B	: 漢字形
	791 〜 799	: 多巻ものの各巻著者標目
	$A	: カタカナ形
	$X	: ローマ字形
	$B	: 漢字形
Subject Headings （1=21）	650	: 個人名件名標目
	658	: 一般件名標目
Publisher （1=1018）	270$B	: 出版者、頒布者等
Note （1=63）	350	: 一般注記
	354	: 原タイトル注記
	360	: 装丁と定価に関する事項
	377	: 内容注記
Any （1=1016）	数字 3 桁 $.	:タグ・フィールドを除いた全て

2.2.2 Z39.50 サーバの機能

本 Z39.50 サーバは接続機能、検索機能、返戻機能、終了機能の基本機能を持つ。接続機能は複数の Z39.50 クライアントによる接続要求を受け付け、検索・返戻時に用いる推奨メッセージサイズやサーバが提供している機能、Z39.50 のバージョン情報などのパラメータの折衝を行なう機能である。

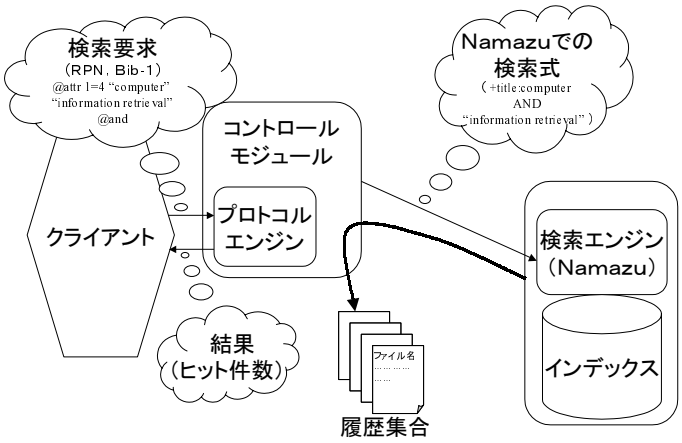

図 2.3: 検索処理

検索はコントロールモジュールがプロトコルエンジンから検索要求を受け取り、コントロールモジュールが逆ポーランド記法の検索式を検索エンジンに渡すことによって行なわれる。検索エンジンはインデックスを検索して、コントロールモジュールに検索結果を返す。コントロールモジュールはヒットしたレコードのファイル名を履歴集合ファイルに出力した後、プロトコルエンジンを通してクライアントにヒット件数を返す（図 2.3 参照)。サーバは検索式として Type1-Query と Type101-Query を受け付ける。検索質問は AND, OR, AND-NOT の論理演算、Bib-1 のアクセスポイント指定、履歴集合指定が可能である。履歴検索は履歴集合同士や、検索語と履歴集合の論理演算を行なえる。履歴集合同士の検索は履歴ファイルのみで行ない、再度インデックスを検索しないので短時間で処理できる。

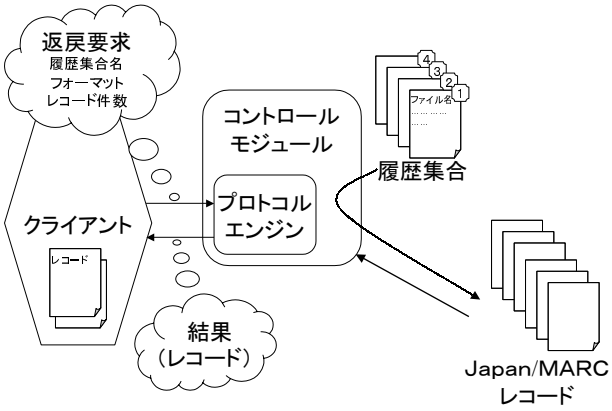

図 2.4: 返戻処理

返戻はまず履歴集合名、レコードシンタックス、返戻レコード開始番号、レコード件数を指定し、指定された履歴集合から実際の JAPAN/MARC レコードファイルを抜き出し、JAPAN/MARC レコードファイルを SUTRS 形式で返すことによって行なわれる（図 2.4 参照)。クライアントへ返戻するレコードのサイズは、接続要求時の推奨メッセージサイズのパラメータ指定による。

終了機能はクライアントとの接続を切り、そのセッションでの履歴集合ファイルの削除を行なう機能である。

2.3 検索例

実際に Z39.50 クライアントで本システムに接続し、検索している例を図 2.5 に示す。Z39.50 クライアントは本学の江草によって開発されている日本語の扱えるもの [31][32] を利用した。先に述べたように、検索結果を返戻する際は、SUTRS 形式で返戻する。

この例では、

1. 「ガラス」を検索: ヒット件数 667 件
2. 「工芸」を検索: ヒット件数 1371 件
3. 1 番目と 2 番目の履歴集合の AND: ヒット件数 19 件

という検索を行なっており、表示されているレコードは 3 番目の検索によるものである。

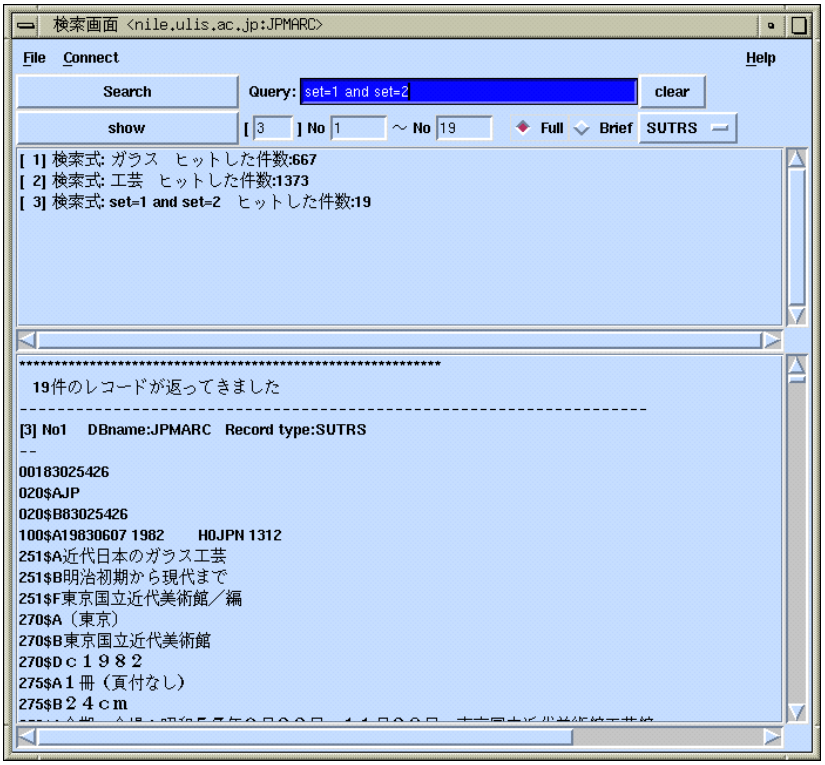

図 2.5: Z39.50-JP システムの検索例

2.4 考察

以上、本研究で開発した Z39.50-JP システムのサーバの構成と機能について述べてきた。ここでは、先行システムと比較しながら本システムの意義について考察する。

本学ではこれまで 3 つの Z39.50 検索システムが構築されてきた。ここでは、今回開発した Z39.50-JP システムと以前の安齋による Z39.50 システム [9][10] および真野による Z39.50 システム [33][32] との相違点についてまとめた。

表 2.2: 開発システムの相違点

	安齋システム （1996 年 11 月）	真野システム （1998 年 2 月）	Z39.50-JP システム （1999 年 5 月）
構成	サーバ WWW ブラウザから検索 （CGI 接続プログラム）	サーバ	サーバ
開発環境	Sun SPARC Station10 YAZ-1.2 DBMS: ADABAS C 言語	IBM RS6000 YAZ-1.3 DBMS: Oracle7 C 言語, SQL	Sun Enterprise 3000 YAZ-1.4pl2 検索エンジン: Namazu C 言語, Perl
データ	JAPAN/MARC 1 年分 （73,549 件） ULIS-OPAC （85,651 件）	JAPAN/MARC （1,601 件）	JAPAN/MARC 15 年 5ヶ月分 （1,019,696 件）
検索機能	論理演算可 フィールド指定可 （書名, 著者, ISBN, 件名, NDC, NDLC, JP-No） 履歴検索可	論理演算一部可 フィールド指定可 （書名, 著者, 全指定） 履歴検索可 （履歴集合同士のみ）	論理演算可 フィールド指定可 （書名, 著者, ISBN, 件名, 出版者, 注記, 全指定） 履歴検索可
レコード	ADABAS のレコード	Oracle のテーブル	1 レコード 1 ファイル
インデックス語 の切りだし	MHSA[34]	なし （SQL の LIKE）	茶筌

本システムでの主な改良点は以下の 3 点である。

1. 大規模システムの構築（データ量の増大）
   Z39.50-JP システムでは約 102 万件のデータを Z39.50 で提供する基盤を構築するために検索レスポンスの面での大幅な改善を求めた。そのために、Namazu を中心とする全文検索エンジンの導入に至った。検索実験も約 102 万件のデータに対して行ない、支障のないレスポンスタイムで検索できた。

2. Namazu の改造による資源共有 [35]
   フリーで入手可能な Namazu に対して JAPAN/MARC に対応させるための改造を行なったことにより、WWW 上で広く利用されている Namazu の検索システムの資源（インデック

ス）をそのまま Z39.50 で提供できる。

3. 複数文字コードの受け付け
   本サーバは複数の文字コードを受け付ける。つまり、日本語の文字コードとして代表的な EUC-JP、ISO-2022-JP、Shift_JIS の 3 種類の文字コード [36] を自動判別し、内部コードの EUC コードでの検索が可能である。

また、本システムは Z39.50 の基本機能については実装したが、細かい部分については今後の課題として残っている。例えば、Bib-1 のアトリビュートタイプの内、アクセスポイント指定（Use）以外の Relation、Position、Structure、Truncation、Completeness の指定には対応しておらず、また返戻時の ElementSetName（詳細・簡略）の指定にも対応していない。Sort や Scan などの拡張機能についても未対応である。しかし、これらの機能は書誌データ検索としては必須の機能ではないと判断し、今後の課題とした。

文字コードについては、サーバの内部では EUC で処理し、JAPAN/MARC は ISO-2022-JP で保存してある。そのため、サーバからクライアントへのレコード返戻時には ISO-2022-JP で転送する。さらに、クライアントからの入力文字コードを内部文字コードである EUC に変換して処理している。このため、日本語文字については言語折衝 [37] がなくても、日本語で Z39.50 に基づく検索ができる。しかし、言語折衝の機能を適切に利用すれば、コード変換の手間をクライアント・サーバ双方で行なう必要がなくなるため、将来的には言語折衝 [37] の部分も実装したいと考えている。このようにシステム構築を通して様々な試みを行なっており、今後も引続き実験を行なう予定である。

2.5 まとめ

本章では情報検索標準プロトコル Z39.50 に基づいた日本語書誌データ検索システムの構築について述べた。日本でも 1999 年の東京工業大学や図書館情報大学の電子図書館システムなど、自館の OPAC データを Z39.50 でサービスするシステムが出てきたものの、日本語処理の問題、海外サーバを含む他のサーバとの相互接続性、レコードシンタックスおよびセマンティックス、メタデータと既存目録データの交換方式、フルテキストなど書誌データ以外の処理機能など解決すべき問題は数多く残っており、Z39.50 の実験システムとして Z39.50-JP システムの意義は大きい。

なお、Namazu を利用した本システムの WWW 上での検索サービスを
`http://nile.ulis.ac.jp/~masao/ulisonly/jmarc-search/` にて試験的に提供している。

第3章　Z39.50-DCシステム[17]

3.1　本システムの位置づけ

Z39.50[3] は情報検索のための国際標準プロトコルであり、その特徴は多種多様な環境で構築された情報検索システムに対して透過的に検索できる点にある。2 章で述べたように、筆者はこれまでに JAPAN/MARC のタグ・フィールドと Z39.50 の代表的アトリビュートセットである Bib-1 とのマッピングを行なうことにより、JAPAN/MARC の書誌データを検索できる Z39.50 システムを構築した [16]。本章ではさらに JAPAN/MARC の書誌データを Dublin Core のスキーマを通して検索できる Z39.50 検索システムの構築について述べる。

Z39.50 ではデータベースの実装に依存しないスキーマを目指し、アトリビュートセットと呼ばれる論理的なスキーマを定義している。しかしながら、Z39.50 は複数のアトリビュートセットを使うと透過的な検索が妨げられるため、結果として大部分の Z39.50 サーバが Bib-1 というひとつのアトリビュートセットを使っている。ところが、実際の検索項目を Bib-1 にマッピングする際のスキーマが定められていないため、このマッピングがシステム依存になっており、異なる検索システムの間では同じフィールドが別のマッピングをしている可能性があるなど検索処理の一貫性が保たれていないという問題点が指摘されている [21]。このデータ間のスキーマの違いを吸収する枠組として Dublin Core の適用可能性が指摘されている [1]。

Dublin Core[22] は基本的な 15 エレメントを定義することによってインターネット上の多様な情報資源に対応し、情報資源発見の効率化をはかるために定義されたコアメタデータである。Dublin Core は単にシンプルなメタデータとして理解するのではなく、様々な分野に共通な要素、すなわち様々な分野にできるだけ共通の概念として認められた属性の要素の集まりとして定義されたコアメタデータとして理解すべきである [38]。Dublin Core が担う最も重要な役割は相互利用性（Interoperability）であり、Z39.50 のアトリビュートセット、JAPAN/MARC や US-MARC などの目録データ、Web コンテンツ、既存の各種メタデータなど多様な情報資源を Dublin Core をゲートウェイとして相互に結ぶことで、情報資源の発見が容易になる。

筆者は検索システム間の相互接続性を目指した Z39.50 と、データスキーマ間の相互利用性を目指した Dublin Core を結合させ、様々な情報資源を統一的に検索できるシステムの構築を目指している。本システムでは Z39.50 と Dublin Core を透過的に利用できる基盤整備という観点でシステム構築を行なった。そこで、JAPAN/MARC データを例にとり Dublin Core を共通スキーマとしてマッピングを行ない、Z39.50 の Bib-1 における Dublin Core 用のアクセスポイントを通して JAPAN/MARC を検索できるシステムを構築した。本システムにより JAPAN/MARC だけでなく、その他の書誌データについても Dublin Core を共通スキーマとした Z39.50 検索システムを容易に構築できるようになる。

3.2 システムの構成

本システムは JAPAN/MARC レコードを Dublin Core に基づいて記述し、利用者は Z39.50 の Bib-1 の Dublin Core 用アクセスポイントから検索できる。本システムはプロトコルエンジン、コントロールモジュール、データベース部、メタデータ変換部からなる（図 3.1）。

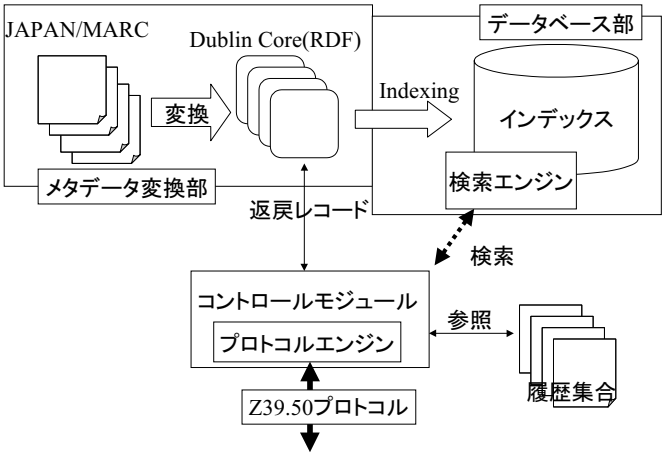

図 3.1: システムの構成

プロトコルエンジンは Z39.50 プロトコルを解釈し、コントロールモジュールに渡す。コントロールモジュールはプロトコルエンジンの解釈に基づき、検索エンジンへパラメータを渡し、セッション毎の情報と履歴集合を管理する。プロトコルエンジンには Index Data 社の YAZ[27] を利用した。

メタデータ変換部は既存の書誌データを Dublin Core に基づくメタデータに変換する。現在は JAPAN/MARC から Dublin Core への変換機能を持っている。JAPAN/MARC のデータ [18] を元に Dublin Core の RDF 表現 [39] に変換した。RDF[23][24] は外部表現として XML（eXtensible Markup Language）[40] を用いるため、実際には Dublin Core は XML で表記される。データの変換は Perl スクリプト（付録 B.1 参照）により行なった。変換元の JAPAN/MARC データと変換後の Dublin Core データの一例をそれぞれ付録 B.2, B.3 に示す。変換時に JAPAN/MARC のフィールド（サブフィールド）と Dublin Core の項目の間のマッピングを行なった。変換の際のマッピングの定義は、表 3.1 に示す。マッピングは Dublin Core に対して一対多となるように行ない、1 つの書誌項目が複数の Dublin Core エレメントに含まれることはない。また、1 つの Dublin Core エレメントに対して、複数のデータ項目が存在した場合は、RDF におけるリスト要素を表す Bag を用いて表現した。なお、JAPAN/MARC で独自に定義されている外字については、適宜変換を行なった[1]。

[1] Â, Î ,Û, Ê, Ô などは A, I , U, E, O に、その他の追加文字は空白文字に変換した。また、JAPAN/MARC では長音（ー）の文字がマイナス（−）記号になっているので、カタカナ・平仮名の直後に現れるマイナス記号は長音に変

表 3.1: Dublin Core と JAPAN/MARC の対応関係

Dublin Core	JAPAN/MARC	フィールドの意味
Title	251–9$A	タイトルと責任表示に関する事項：本タイトル
	251–9$B	タイトルと責任表示に関する事項：タイトル関連情報
	251–9$D	タイトルと責任表示に関する事項：巻次、回次、年次等
	280$A	叢書名に関する事項：叢書名
	280$B	叢書名に関する事項：叢書番号
	280$D	叢書名に関する事項：副叢書名
	280$F	叢書名に関する事項：副叢書番号
	281–3$A	シリーズに関する事項：本シリーズ名
	281–3$B	シリーズに関する事項：シリーズ名関連情報
	281–3$D	シリーズに関する事項：シリーズ番号
	281–3$S	シリーズに関する事項：下位シリーズ名
	281–3$T	シリーズに関する事項：下位シリーズ番号
	281–3$X	シリーズに関する事項：シリーズの ISBN
	291–9$A	多巻ものの各巻のタイトルと責任表示に関する事項：タイトル
	291–9$B	多巻ものの各巻のタイトルと責任表示に関する事項：副書名
	291–9$D	多巻ものの各巻のタイトルと責任表示に関する事項：巻次、回次、年次等
	354$A	原タイトル注記：翻訳資料の原タイトル
	551–9$A	タイトル標目（タイトル関連情報の読み等を含む）：カタカナ形
	551–9$X	タイトル標目（タイトル関連情報の読み等を含む）：カタカナ形ローマ字形
	580$A	叢書名標目：カタカナ形
	580$X	叢書名標目：ローマ字形
	581–3$A	シリーズのタイトル標目：カタカナ形
	581–3$X	シリーズのタイトル標目：ローマ字形
	591–9$A	多巻ものの各巻のタイトル標目：カタカナ形
	591–9$X	多巻ものの各巻のタイトル標目：ローマ字形
Creator	251–9$F	タイトルと責任表示に関する事項：責任表示
	751–9$A	著者標目：カタカナ形
	751–9$B	著者標目：漢字形
	751–9$X	著者標目：ローマ字形

換した。

（前ページの続き）

Dublin Core	JAPAN/MARC	フィールドの意味
Subject	650$A	個人件名：カタカナ形
	650$B	個人件名：漢字形
	650$X	個人件名：ローマ字形
	658$A	一般件名：カタカナ形
	658$B	一般件名：漢字形
	658$X	一般件名：ローマ字形
	677$A	NDC：分類記号
	685$A	NDLC 分類：分類記号（またはかな付）
	685$X	NDLC 分類：ローマ字付分類記号
Description	350$A	一般注記：一般注記
	377$A	内容注記：内容に関する注記
Publisher	270$B	出版・領布に関する事項：出版者、領布者等
Contributor	281–3$F	シリーズに関する事項：シリーズに関する責任情報
	291–9$F	多巻ものの各巻のタイトルと責任表示に関する事項：著者表示
	781–3$A	シリーズの著者名標目：ローマ字形
	781–3$B	シリーズのの著者標目：巻次の読み
	781–3$X	シリーズのの著者標目：漢字形
	791–9$A	多巻ものの各巻の著者標目：ローマ字形
	791–9$B	多巻ものの各巻の著者標目：巻次の読み
	791–9$X	多巻ものの各巻の著者標目：漢字形
Type	（該当無し）	
Date	270$D	出版・領布に関する事項：出版、領布年月
Identifier	010$A	国際標準図書番号：ISBN
	020$B	全国書誌番号：全国書誌番号（JP 番号）
	905$A	NDL の請求記号：請求記号
	906$A	NDL の印刷カード番号：印刷カード番号
Format	（該当無し）	
Language	101$A	著作の言語（翻訳物に適用）：テキストの言語
Source	（該当無し）	
Relation	（該当無し）	
Coverage	270$A	出版・領布に関する事項：出版地、領布地等
Rights	（該当無し）	
（該当無し）	001	レコード識別番号：レコードコントロール番号
	020$A	全国書誌番号：国名コード

（前ページの続き）

Dublin Core	JAPAN/MARC	フィールドの意味
	100$A	一般的処理データ
	101$C	著作の言語（翻訳物に適用）：原文の言語
	251–9$W	タイトルと責任表示に関する事項：資料種別表示
	261$A	並列タイトルに関する事項：並列タイトル
	265$A	版に関する事項
	275$A	形態に関する事項：特定資料種別と資料の数量
	275$B	形態に関する事項：大きさ
	275$E	形態に関する事項：付属資料
	360$A	装丁と定価に関する事項：装丁
	360$B	装丁と定価に関する事項：本体価格
	360$C	装丁と定価に関する事項：税込価格
	386$A	ファイル内容に関する注記（コンピュータファイル）：ファイル内容注記
	387$A	システム要件に関する注記（コンピュータファイル）：システム要件注記
	551–9$B	タイトル標目（タイトル関連情報の読み等を含む）：漢字形（所在フィールドの識別子）
	551–9$D	タイトル標目（タイトル関連情報の読み等を含む）：巻次の読み
	580$B	叢書名標目：漢字形（所在フィールドの識別子）
	580$D	叢書名標目：叢書番号
	581–3$B	シリーズのタイトル標目：漢字形（所在フィールドの識別子）
	581–3$D	シリーズのタイトル標目：巻次等の読み
	591–9$B	多巻ものの各巻のタイトル標目：漢字形（所在フィールドの識別子）
	591–9$D	多巻ものの各巻のタイトル標目：巻次の読み
	677$V	NDC：NDC 版次

データベース部はメタデータの検索を行なう。データベースの検索エンジンには、全文検索エンジン Namazu[20] を採用した。本システムでは、Dublin Core の RDF 表現を解析し、エレメントの内容を抽出するフィルタ（付録 B.4 参照）を Namazu のインデクサ（`mknmz`）のフィルタ機構に組み込んだ。このフィルタでは、XML パーサとして Perl モジュール `XML::Parser` を利用し、文字列がどのエレメントに含まれているかを判別して、Namazu のフィールド用インデクスに登録する。現在本システムに登録されているデータ件数は 1,593 件である。

本システムでは、Z39.50 の Bib-1 の Use アトリビュートに追加された Dublin Core 用の 15 項目 [41] に対応した検索ができる（表 3.2 参照）。この 15 項目のアクセスポイント指定は Dublin Core

の項目と 1 対 1 の関係にあるため、アトリビュートの扱いがあいまいになることはない。Z39.50 プロトコルでの検索要求は、内部的に Namazu の検索式に変換され、検索エンジンに渡される。また、検索要求としては履歴検索も扱える。

表 3.2: Bib-1 に追加された Dublin Core エレメント

Z39.50 Bib-1 Use Attribute		Dublin Core
Name	Value	
DC-Title	1097	Title
DC-Creator	1098	Creator
DC-Subject	1099	Subject
DC-Description	1100	Description
DC-Publisher	1101	Publisher
DC-Date	1102	Date
DC-ResourceType	1103	Type
DC-ResourceIdentifier	1104	Identifier
DC-Language	1105	Language
DC-OtherContributor	1106	Contributor
DC-Format	1107	Format
DC-Source	1108	Source
DC-Relation	1109	Relation
DC-Coverage	1110	Coverage
DC-RightsManagement	1111	Rights

3.3 検索例

実際に Z39.50 クライアントで本システムに接続し、Dublin Core アクセスポイントで検索した例を図 3.2 に示す。検索結果を返戻する際は、SUTRS 形式で返戻する。

この例では、

1. DC-Title で「河童」を検索: ヒット件数 1 件
2. DC-Creator で「赤川次郎」を検索: ヒット件数 2 件

という検索を行なっており、表示されているレコードは 2 番目の検索によるものである。

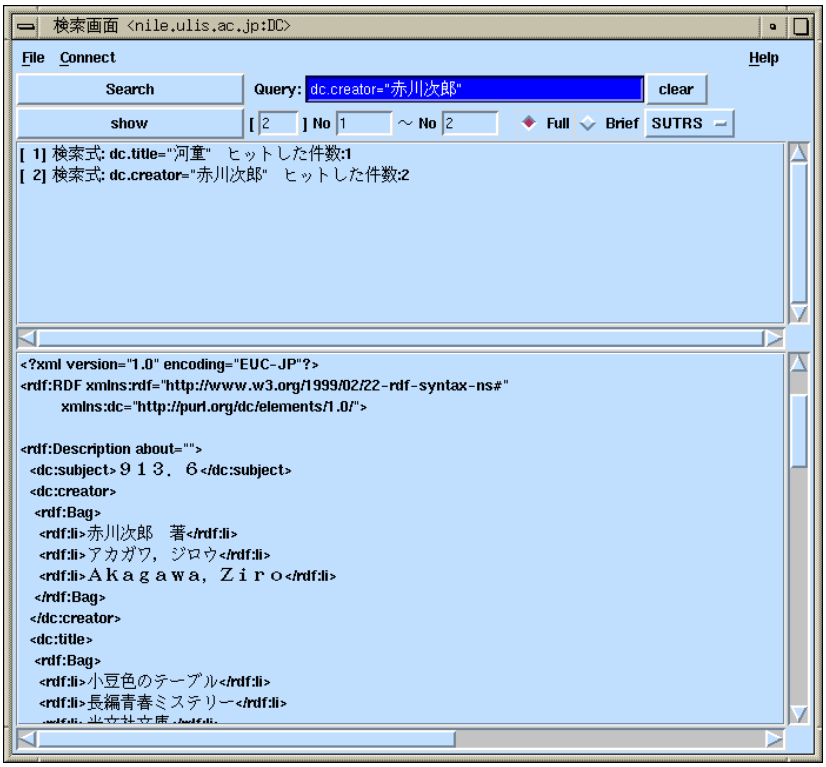

図 3.2: Z39.50-DC システムの検索例

3.4 考察

3.4.1 関連研究

多様な情報資源を容易に利用できる仕組みを作りたいという要求は、どの分野でも共通の話題であり、それぞれの立場から様々な提案が行われている。この節では特に Z39.50 に関わる研究について考察する。

Z39.50 は ZIG（Z39.50 Implementors Group）によって規格の検討と合意がなされており、現在 ZIG におけるこのマッピング問題に対処することを目指した議論の中で、複数のアトリビュートセットを同時に扱うことを目的とした Attribute Architecture[42] の定義が進められている。この枠組の中心は Cross-Domain Attribute Set[43] であり、これは、Dublin Core を基本にした 13 項目のアクセスポイント指定からなるアトリビュートセットで、分野を横断した検索に利用することが期待されている。しかし、この方法は現行の Z39.50 システムに対して、Bib-1 から Cross-Domain Attribute Set に変更を求めることになり、アトリビュートセットの移行によって相互利用性が損なわれる可能性がある。

また、JAPAN/MARC と Z39.50 との関わりについて石田 [44][45] は、Bib-1 で流用するのではなく JAPAN/MARC 専用のアトリビュートセットを提案している。石田のシステムは JAPAN/MARC に特化したアトリビュートセットを設計し、実際にシステムを公開して評価実験を行っている。し

かし、この手法ではJAPAN/MARCを検索する際にはJAPAN/MARCの構造に従った検索ができるものの、JAPAN/MARC以外のデータに対しては適用できず、Bib-1との互換性もなくなってしまう欠点がある。

Z39.50とDublin Coreを結合した例としては図書館情報大学電子図書館（ULIS-DL）[12]のZ39.50サービスがある。ULIS-DLはサブジェクトゲートウェイを目指したメタデータの構築を行っており、ULIS Coreと呼ばれるDublin Coreに基づいたエレメントセットを用いている。ULIS-DLはもともとDublin Coreを基本として作られたシステムであるので、標準検索プロトコルZ39.50と組み合わせるのは自然であり、データそのものが元々Dublin Coreで作成されているため、システム構築時にアトリビュートセットのマッピングによる問題が生じることはない。しかし、今後ULIS-DLがサブジェクトゲートウェイとしての能力を高め、他のデータを対象にする場合には本システムで研究しているスキーマ変換機能が必要になるだろう。

また、齋藤らはWoPEcとWAGILSという2つのメタデータをDublin Coreとマッピングし、Dublin Coreのエレメントで検索するシステム[46]も構築している。

以上のように、Z39.50の視点を重視するのか、JAPAN/MARCなどある特定データの流通性を重視するのか、多様なメタデータの相互利用性を重視するのかによってアプローチが異なるが、本研究ではZ39.50の視点を踏まえた相互利用性を重視する立場からシステム構築を行なった。

3.4.2 JAPAN/MARCデータのDublin Coreへの変換

本論文はZ39.50とDublin Coreのシステム基盤構築に焦点があるため、JAPAN/MARCとDublin Coreのマッピングについては深く立ち入らなかった。マッピングについては今後の問題として、今回のマッピング方針と将来展望について述べる。

Dublin Coreには、Dublin Core Simple（DCS）とDublin Core Qualifier（DCQ）という考え方の相違による2つのタイプがある。DCSは15項目のデータ要素をさらに細かく分けることはしない書き方で、DCQはこの基本要素を細かく分けて記述する書き方である。今回はDCSを採用し、データは並列に並べた。JAPAN/MARCのタグに表されているデータのみをDublin Coreに変換し、JAPAN/MARCレコードのデータ部に含まれていない項目はDublin Coreへの変換を行なわず該当なしとした。データのマッピングについては現在検討中で、DCQによる記述も検討している。

3.5 まとめ

本章ではJAPAN/MARCを例としてDublin Coreを共通スキーマとしてマッピングを行ない、Z39.50のBib-1のDublin Core用アクセスポイントを通してJAPAN/MARCを検索できるシステムの構築について述べた。このZ39.50-DCシステムによりJAPAN/MARCだけでなく他のデータについてもDublin Coreを共通スキーマにすれば本システムを利用してZ39.50で検索できる。

第4章　考察

4.1　Z39.50-JP システムと Z39.50-DC システムの比較

本節では、Z39.50-JP システムと Z39.50-DC システムの相違点などについて比較検討する。

まずユーザから見た場合、Z39.50-JP システムでは JAPAN/MARC の書誌データを検索できる。検索の際には、Bib-1 の Author や Title などの一般的なアクセスポイントを指定して検索を行なう。一方、Z39.50-DC システムでは Dublin Core メタデータを検索でき、アクセスポイントには Dublin Core 専用の DC-Creator や DC-Title などを用いる。

また、Z39.50 サーバ構築という観点から見た場合、Z39.50-JP システムでは Bib-1 と JAPAN/MARC のフィールドとの間のマッピングを行なった（表 2.1 参照）。しかし、このようなマッピングが複数の Z39.50 システムの間で異なってしまうことが、Z39.50 での検索処理における問題点とされている。そこで、この課題を解決するために構築された Z39.50-DC システムでは Dublin Core と Bib-1 とのマッピングは既に一対一の定義がされており（表 3.2 参照）、サーバ構築の際に新たなマッピングを行なう必要はない。このような Dublin Core 検索システムが JAPAN/MARC 以外にも適用・普及することで、Z39.50 の持つ相互利用性がさらに向上すると考えられる。また Z39.50-DC システムでは、Dublin Core と既存の書誌データである JAPAN/MARC の間の変換を行なったため、この変換の際のマッピングが必要となった。

また、Z39.50-JP システム・Z39.50-DC システムともに Z39.50 と WWW の両者に対応したシステムとして構築された。本システムで採用した検索エンジン Namazu は元々WWW 用の全文検索エンジンであるため、両システムの情報を Z39.50 でも WWW でも提供できる。また、既存の Namazu を採用している WWW 上の検索サービスも本研究で開発した Z39.50 サーバを組み込むことでそのまま Z39.50 でも提供できるなど情報資源の共有も可能になった。

また、Z39.50-JP システムでは約 102 万件の大規模データを検索できる。Z39.50-DC システムは現在約 1000 件のレコードしか入力していないが、全文検索エンジンなどデータベース部の構成は同一のものであるので、大量のデータを入力しても問題はない。また、前処理として行なう Dublin Core への変換処理も処理時間上の問題はない。

また、Z39.50 サーバから返戻されるレコードは両システムとも SUTRS 形式のレコードである。しかし、Z39.50-JP システムではレコードの内容は JAPAN/MARC のタグつきテキストであるが、Z39.50-DC システムでは RDF で記述された Dublin Core メタデータである。

4.2　研究の意義と今後の課題

本研究の意義は以下の 2 点にまとめられる。

- 情報資源の共有

  本研究では、Z39.50 と WWW の両方で利用可能な検索システムを開発した。これは、本システムの情報を Z39.50 の WWW の両方で提供可能であるだけでなく、既存の WWW で提供されていた検索システムであっても Namazu を使用していれば、本研究で開発した Z39.50 サーバを組み込むことでそのまま Z39.50 で提供できる。

  さらに、JAPAN/MARC の書誌データを変換した Dublin Core メタデータは、本研究以外のシステムでも Dublin Core メタデータとして利用が可能である。

- 実験システム

  また、実験システムとして Dublin Core を利用し、大規模データに対応しうる Z39.50 検索システムの開発を行なった。

一方、本研究では Z39.50 の基本機能についてはほぼ実装できたが、Z39.50 の拡張機能の部分についてはまだ実装を行なっていない部分が多く、これらについては今後の課題として残されている。

また、Dublin Core の実際の記述では、RDF をメタデータ記述のための枠組として利用したが、JAPAN/MARC の内、Dublin Core の 15 エレメントの中に表現できなかった項目については、対応していない。今後、これらのデータ項目を RDF スキーマ [24] で表現し、より JAPAN/MARC の詳細な表現が可能となることで、検索システムなどでのメタデータの再利用性がさらに高まると考えられる。

第5章　おわりに

本論文では情報検索のための国際標準プロトコル Z39.50 に基づいた日本語書誌データ検索システムの構築について述べた。Z39.50 検索システムとしては Z39.50-JP システム、Z39.50-DC システムの 2 種類のシステムを構築し、Z39.50 の相互利用性のさらなる向上、WWW での情報資源の共有、大規模データへの対応などを実現した。

日本でも 1999 年の東京工業大学や図書館情報大学のディジタル図書館システムなど、自館の OPAC データを Z39.50 でサービスするシステムが出てきたものの、日本語の構造に基づく処理の問題、海外サーバを含む他のサーバとの相互接続性、レコードシンタックスおよびセマンティックス、フルテキストなど書誌データ以外の処理機能など解決すべき問題は数多く残っており、今後も Z39.50 の実験システムの必要性は高い。本システムは前述の成果を挙げた。実験システム構築による手法の検討という観点において本研究の意義は大きいと考える。

なお、本研究の一部は、情報知識学会論文誌 [16]、ディジタル図書館ワークショップ [17] で発表を行なった。

また、筆者は 1999 年 7 月以降、全文検索システム Namazu の共同開発者の一員としても活動していることを付記しておく。

謝辞

本研究全般にわたり、常に力強い励ましとご指導をいただきました石塚英弘先生、宇陀則彦先生に感謝致します。

本研究用データとして JAPAN/MARC の使用を許可頂いた国立国会図書館の関係各位に感謝致します。Dublin Core メタデータの最新動向を教えて頂いた杉本重雄先生に感謝致します。全文検索システム Namazu という有用なソフトウェアを開発された高林哲さんに感謝致します。また、namazu-dev@ring.gr.jp の皆様には、プログラミング技法など多くのことを教えて頂きました。感謝致します。

また、江草由佳さんは、私が Z39.50 システムに取り組むきっかけを与えてくれ、その後のシステム構築にも数多くの助言をいただきました。また、貴重な研究成果である Z39.50 クライアントも利用させて頂きました。深く感謝致します。

石塚研究室、図書館情報システム論講座（2 講）の皆様をはじめとして、研究を通じてお世話になった全ての方に感謝致します。

最後に、私の学生生活を支えてくれた両親に感謝致します。

参照文献

[1] Lynch, Clifford A. The Z39.50 Information Retrieval Standard : Part I: A Strategic View of Its Past, Present and Future. D-Lib Magazine. 1997.
URL: <http://www.dlib.org/dlib/april97/04lynch.html>.

[2] 上田修一. Z39.50 とその可能性. 情報の科学と技術. Vol. 48, No. 3, 1998, p. 126–133.

[3] ANSI/NISO Z39.50-1995. Information Retrieval（Z39.50）: Application Service Definition and Protocol Specification. 1995, 156p.
URL: <ftp://ftp.loc.gov/pub/z3950/official/>.

[4] 牛崎進. Z39.50 : IR サービスの通信プロトコル. カレントアウェアネス. No. 175, 1994, p. 4.

[5] 上田修一. Z39.50 の可能性と問題点. 三田図書館・情報学会研究大会予稿集, 1996, p. 37–40.
URL: <http://www.slis.keio.ac.jp/~ueda/z3950/mita96.html>.

[6] 松林正己. 標準情報検索プロトコル Z39.50 の国際的展望. 情報の科学と技術. Vol. 48, No. 3, 1998, p. 144–155.

[7] Index Data. Z39.50 Target Information. last update 2000-01-14, Continually updated.
URL: <http://www.indexdata.dk/targettest/targetstat.shtml>.

[8] JIS X 0806 （ISO 23950）. 情報検索（Z39.50）応用サービス定義及びプロトコル仕様. 東京, 日本規格協会, 1999, 272p.

[9] 安齋宏幸, 山本毅雄, 石塚英弘. Z39.50 を用いた日本語書誌情報サーバの試作. 情報処理学会情報学基礎研究会研究報告. Vol. 96, No. 116, 1996, p. 9–16.

[10] 安齋宏幸. インターネット環境における日本語書誌情報システムの構築. つくば, 図書館情報大学, 1997. 修士論文.

[11] 早稲田大学学術情報システム（WINE）.（参照 2000-01-15）.
URL: <http://wine.wul.waseda.ac.jp/>.

[12] 図書館情報大学附属図書館. 図書館情報大学電子図書館（ULIS-DL）.（参照 2000-01-15）
URL: <http://lib.ulis.ac.jp/>.

[13] 平岡博, 真中孝行, 横山敏秋, 阪口哲男, 杉本重雄, 田畑孝一. 図書館情報大学ディジタル図書館システム. 情報管理. Vol. 42, No. 6, 1999, p. 471–479.

[14] Titech Digital Library（東京工業大学電子図書館）. last update 1999-03-25.
URL: <http://tdl.libra.titech.ac.jp/>.

[15] 小島明, 田中純, 篠原正紀, 落合崇道. 大学での取り組みが進む電子図書館. NTT 技術ジャーナル. Vol. 11, No. 1, 1999, p. 55–58.

[16] 宇陀則彦, 江草由佳, 高久雅生, 石塚英弘. Z39.50 による日本語書誌データ検索システム. 情報知識学会誌. Vol. 9, No. 2, 1999, p. 1–15.

[17] 高久雅生, 江草由佳, 宇陀則彦, 石塚英弘. Z39.50 による書誌データ検索システムの構築：Dublin Core を共通スキーマとして. ディジタル図書館. No. 16, 1999, p. 97–106.

[18] 国立国会図書館. JAPAN/MARC マニュアル：図書編. 第 2 版. 東京, 国立国会図書館, 1998, 186p. （ISBN 4-87582-527-7）.

[19] 黒澤正彦, 西村徹 編. マークをうまく使うには：機械可読目録入門. 東京, 三洋出版貿易, 1986, 310p. （ISBN 4-87930-030-6）.

[20] 高林哲. 全文検索システム Namazu. last update 1999-09-28.
URL: <http://openlab.ring.gr.jp/namazu/>.

[21] Lynch, Clliford A. Building the Infrastructure of Resource Sharing : Union Catalogs, Distributed Search, and Cross-Database Linkage. LIBRARY TRENDS. Vol. 45, No. 3, 1997, p. 448–461.

[22] Dublin Core Metadata Initiative. The Dublin Core Metadata Element Set Version 1.1. last update 1999-07-02.
URL: <http://purl.org/dc/documents/rec-dces-19990702.htm>.

[23] Ora Lassila; Ralph R. Swick, ed. Resource Description Framework （RDF） Model and Syntax Specification. World Wide Web Consortium, 1999, REC-rdf-syntax-19990222.
URL: <http://www.w3.org/TR/REC-rdf-syntax/>.

[24] Dan Brickley; R.V. Guha, ed. Resource Description Framework （RDF） Schema Specification. World Wide Web Consortium, 1999, REC-rdf-schema-19990303.
URL: <http://www.w3.org/TR/PR-rdf-schema/>.

[25] 原田昌紀. サーチエンジン徹底活用術. 東京，オーム社，1997, 250p.（ISBN 4-274-06230-9）.

[26] 安齋宏幸. Z39.50 の技術解説. 情報の科学と技術. Vol. 48, No. 3, 1998, p. 134–139.

[27] Index Data. The YAZ Toolkit. last update 1999-12-10.
URL: <http://www.indexdata.dk/yaz/>.

[28] 馬場肇. 日本語全文検索システムの構築と活用. 東京, ソフトバンク, 1998, 258p. （ISBN 4-7973-0691-2）.

[29] 奈良先端科学技術大学院大学自然言語処理学講座. 日本語形態素解析ツール・茶筌. 2000-01-11.
URL: <http://cl.aist-nara.ac.jp/lab/nlt/chasen/>.

[30] Bib-1 Attribute Set. last update 1999-09-24.
URL: <http://lcweb.loc.gov/z3950/agency/defns/bib1.html>.

[31] 江草由佳. Z39.50 プロトコルを用いた検索クライアントの開発. つくば, 図書館情報大学, 1998. 卒業論文.

[32] 江草由佳, 真野泰久, 宇陀則彦, 石塚英弘. Z39.50 プロトコルによる日本語書誌データ情報検索システム. 第 6 回研究報告会講演論文集. 情報知識学会, 東京, 1998-05. 情報知識学会, 1998, p.29–36.

[33] 真野泰久. Z39.50 プロトコルを用いた検索サーバの開発. つくば, 図書館情報大学, 1998. 卒業論文.

[34] Hasebe Kigen; Nakamoto Ken'iti; Yamamoto Takeo. An information retrieval system on internet for languages without obvious word delimiters. Proceedings of International Symposium on Digital Libraries 1995. Tsukuba, University of Library and Information Science, 1995-08. p. 181–185.

[35] 馬場肇. 日本語全文検索エンジンソフトウェアのリスト. last update 2000-01-11.
URL: <http://www.kusastro.kyoto-u.ac.jp/~baba/wais/other-system.html>.

[36] Ken Lunde. （春遍雀來, 鈴木武生 訳） 日本語情報処理. 東京, ソフトバンク, 1999, 496p. （ISBN 4-89052-708-7）.

[37] Character Set and Language Negotiation （2）. 1998.
URL: <http://lcweb.loc.gov/z3950/agency/defns/charsets.html>.

[38] 杉本重雄. Dublin Core Metadata Element Set : 現在の状況と利用例. ディジタル図書館. No. 14, 1999, p. 3–18.

[39] Eric Miller; Paul Miller; Dan Brickley. Guidance on expressing the Dublin Core within the Resource Description Framework （RDF）. last update 1999-07-01.
URL: <http://www.ukoln.ac.uk/metadata/resources/dc/datamodel/WD-dc-rdf/>.

[40] Tim Bray; Jean Paoli; C. M. Sperberg-McQueen, ed. Extensible Markup Language（XML）1.0. World Wide Web Consortium, 1998, REC-xml-19980210.
URL: <http://www.w3.org/TR/1998/REC-xml-19980210>.

[41] Ralph LeVan. Dublin Core and Z39.50. Draft Version 1.2. last update 1998-02-02.
URL: <http://purl.org/DC/documents/notes-levan-19980202.htm>.

[42] Z39.50 Attribute Architecture. Version 1.1. last update 1999-07-09.
URL: <http://lcweb.loc.gov/z3950/agency/attrarch/arch.html>.

[43] The Z39.50 Cross-Domain Attribute Set. Version 1.4. last update 1999-08-04.
URL: <http://www.oclc.org/~levan/docs/crossdomainattributeset.html>.

[44] 石田茂. Z39.50 と日本語書誌目録の連携に関する考察. 情報処理学会情報学基礎・データベースシステム研究会研究報告. Vol. 99, No. 39, 1999, p. 81–88.

[45] 石田茂. Z39.50 JAPAN/MARC 対応関連仕様と評価用試作システムの説明. ディジタル図書館. No. 15, 1999, p. 40–56.

[46] 齋藤ひとみ, 宇陀則彦, 石塚英弘. Dublin Core Metadata Element Set による複数メタデータの検索. ディジタル図書館. No. 11, 1998, p. 48–55.

参考文献

- 図書館情報学ハンドブック編集委員会 編. “7. 情報検索システム”. 図書館情報学ハンドブック. 東京, 丸善, 1988, p. 593–701. （ISBN 4-621-03232-1）
- 山本毅雄, 橋爪宏達, 神門典子, 清水美都子. （学術情報センター 編）全文検索：技術と応用. 東京, 丸善, 1998, 124p. （ISBN 4-621-04525-3）
- Ricardo Baeza-Yates; Berthier Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley Longman, 1999, 513p. （ISBN 0-201-39829-X）
- G. G. Chowdhury. The Internet and Information Retrieval Research : A Brief Review. Journal of Documentation. Vol. 55, No. 2, 1999, p. 209–225.
- Peter Evans. Z39.50: Part 1- an overview. last update 1999-11-05.
  URL: <http://www.biblio-tech.com/html/z39_50.html>
- Peter Evans. Z39.50: Part 2 - Technical Details. last update 1999-11-05.
  URL: <http://www.biblio-tech.com/html/z39_50_part_2.html>
- Randal L. Schwartz; Tom Christiansen. （近藤嘉雪 訳）初めての Perl. 第 2 版. 東京, オーム社, 1998, 318p. （ISBN 4-900900-81-8）
- Larry Wall; Tom Christiansen; Randal L. Schwartz. （近藤嘉雪 訳）プログラミング Perl 改訂版. 東京, オーム社, 1997, 759p. （ISBN 4-900900-48-6）
- XML/SGML サロン. 標準 XML 完全解説. 東京, 技術評論社, 1998, 354p. （ISBN 4-7741-0584-8）

付録

以下、本研究で利用したプログラムおよびデータを付録として添付する。
また、これらのプログラムの動作には以下の環境が必要である。

- Perl
- YAZ
- 全文検索システム Namazu
- nkf
- 茶筌
- XML::Parser （Perl モジュール）
- Convert::EBCDIC （Perl モジュール）

付 録A　Z39.50-JPシステム

jmarcserver.c ： JAPAN/MARC データの検索を行なう Z39.50 サーバ（C プログラム）。

jmarcsesrver.h ： 上記 C プログラム用ヘッダファイル。

jmarcfilter.pl ： JAPAN/MARC データをテキストに変換する Perl スクリプト。

jmarc-namazu-2.1.patch ： Namazu で JAPAN/MARC テキストファイルを Indexing できるようにするパッチ。

A.1 jmarcserver.c

```
/*
 * $Id: jmarcserver.c,v 1.2 2000/01/14 10:32:57 masao Exp $
 */
/*
 * Demonstration of simple server
 */

/*
 * Modified by Masao Takaku.
 * For Japan/MARC server.
 */

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <assert.h>

#include <backend.h>
#include <xmalloc.h>
#include <proto.h>
#include <log.h>

#include "jmarcserver.h"

int bend_sort (void *handle, bend_sortrequest *req, bend_sortresult *res)
{
    res->errcode = 1;
    res->errstring = "Sort not implemented";
    res->sort_status = Z_SortStatus_failure;
    return 0;
}

bend_initresult *bend_init(bend_initrequest *q)
{
    bend_initresult *r = odr_malloc (q->stream, sizeof(*r));
    static char *dummy = "Hej fister";

    r->errcode = 0;
    r->errstring = 0;
    r->handle = dummy;
    q->bend_sort = bend_sort;    /* register sort handler */
    return r;
}

/*
 * Z_Term をもらって文字列を返す。
```

```
 */
void get_term(char *buf, Z_Term *term, bend_searchresult *r)
{
  buf[0] = '\0';

  switch (term->which) {
  case Z_Term_general:
    memcpy(buf, term->u.general->buf, term->u.general->len);
    buf[term->u.general->len] = '\0';
    return;
    break;
  case Z_Term_numeric:
    logf(LOG_FATAL, "Numeric Term Not Support.");
    r->errcode = 229;
    r->errstring = "Numeric Term Not support.";
    return;
    break;
  case Z_Term_characterString:
    logf(LOG_FATAL, "Character String Not Support.");
    r->errcode = 229;
    r->errstring = "Character String Not support.";
    return;
    break;
  case Z_Term_oid:
    logf(LOG_FATAL, "OID Term Not Support.");
    r->errcode = 229;
    r->errstring = "OID Term Not Support.";
    return;
    break;
  case Z_Term_dateTime:
    logf(LOG_FATAL, "Date-Time Term Not Support.");
    r->errcode = 229;
    r->errstring = "Date-Time Term Not Support.";
    return;
    break;
  case Z_Term_external:
    logf(LOG_FATAL, "External Term Not Support.");
    r->errcode = 229;
    r->errstring = "External Term Not Support.";
    return;
    break;
  case Z_Term_integerAndUnit:
    logf(LOG_FATAL, "Integer & Unit Term Not Support.");
    r->errcode = 229;
    r->errstring = "Integer & Unit Term Not Support.";
    return;
    break;
```

```
  case Z_Term_null:
    logf(LOG_FATAL, "NULL Term Not Support.");
    r->errcode = 229;
    r->errstring = "NULL Term Not Support.";
    return;
    break;
  default:
    logf(LOG_FATAL, "Unknown Term");
    r->errcode = 229;
    r->errstring = "Unknown Term";
    return;
    break;
  }
}

/*
 * Z_Operand を文字列に変換して返す。
 * まだ general 以外には未対応。
 */
void get_operand(char *term, Z_Operand *op, bend_searchresult *r)
{
  int i = 0;
  char buf[BUFSIZE];
  char buf_term[BUFSIZE];
  Z_AttributeElement *element;

  term[0]='\0';

  switch (op->which) {
  case Z_Operand_APT:
    if (! op->u.attributesPlusTerm->num_attributes) {
      get_term(term, op->u.attributesPlusTerm->term, r);
      return;
    }
    /* AttributeList を見る */
    while (i < op->u.attributesPlusTerm->num_attributes) {
      element = op->u.attributesPlusTerm->attributeList[i];
      switch (*element->attributeType) {
      case 1:
        switch (element->which) {
        case Z_AttributeValue_numeric:
          switch (*element->value.numeric) {
          case 1016: /* 全部 (Any) */
            get_term(buf_term, op->u.attributesPlusTerm->term, r);
            strcat(term, buf_term);
            logf(LOG_DEBUG, "This is Any search");
            break;
```

```
  case 4:    /* 書名 (Title) */
    get_term(buf_term, op->u.attributesPlusTerm->term, r);
    sprintf(buf, "+title:%s", buf_term);
    strcat(term, buf);
    logf(LOG_DEBUG, "This is Title-field search");
    break;
  case 1003: /* 著者名 (Author) */
    get_term(buf_term, op->u.attributesPlusTerm->term, r);
    sprintf(buf, "+author:%s", buf_term, buf_term);
    strcat(term, buf);
    logf(LOG_DEBUG, "This is Author-field search");
    break;
  case 7:    /* ISBN */
    get_term(buf_term, op->u.attributesPlusTerm->term, r);
    sprintf(buf, "+isbn:%s", buf_term);
    strcat(term, buf);
    logf(LOG_DEBUG, "This is ISBN-field search");
    break;
  case 21:   /* 件名 (Subject Headings) */
    get_term(buf_term, op->u.attributesPlusTerm->term, r);
    sprintf(buf, "+subject:%s", buf_term, buf_term);
    strcat(term, buf);
    logf(LOG_DEBUG, "This is SubjectHeadings-field search");
    break;
  case 1018: /* 出版者 (Publisher) */
    get_term(buf_term, op->u.attributesPlusTerm->term, r);
    sprintf(buf, "+publisher:%s", buf_term, buf_term);
    strcat(term, buf);
    logf(LOG_DEBUG, "This is Publisher-field search");
    break;
  case 63:  /* 注記 (Note) */
    get_term(buf_term, op->u.attributesPlusTerm->term, r);
    sprintf(buf, "+note:%s", buf_term, buf_term);
    strcat(term, buf);
    logf(LOG_DEBUG, "This is Note-field search");
    break;
  default: /* Non-support */
    logf(LOG_LOG, "This is Non-Support-field search");
    r->errcode = 114;
    sprintf(r->errstring, "%d", *element->value.numeric);
    break;
  }
  break;
case Z_AttributeValue_complex:
  logf(LOG_LOG, "This is Complex AttributesValue");
  r->errcode = 246;
  break;
```

```
        default:
          r->errcode = 108;
          logf(LOG_DEBUG, "Unknown AttributeValue");
          break;
        }
        break;
      }
      i++;
    }
    break;
  case Z_Operand_resultSetId:
    logf(LOG_FATAL, "ResultSetID Not Support.");
    r->errcode = 18;
    r->errstring = "ResultSetID Not Support.";
    break;
  case Z_Operand_resultAttr:
    logf(LOG_FATAL, "ResultSetPlusAttributes Not Support.");
    r->errcode = 245;
    r->errstring = "ResultSetPlusAttributes Not Support.";
    break;
  default:
    logf(LOG_FATAL, "Unknown Operand");
    r->errcode = 3;
    r->errstring = "Unknown Operand";
    break;
  }
}

/*
 * Z_Operatorを文字列に変換する。
 */
void get_operator(char *query, Z_Operator *op)
{
  switch(op->which) {
  case Z_Operator_and:
    strcpy(query, "and");
    break;
  case Z_Operator_or:
    strcpy(query, "or");
    break;
  case Z_Operator_and_not:
    strcpy(query, "not");
    break;
  case Z_Operator_prox:
    logf(LOG_FATAL, "Proximity operator not support");
    query = NULL;
    break;
```

```
  default:
    logf(LOG_FATAL, "Unknown operator");
    query = NULL;
    break;
  }
}

/*
 * parse_rpn_structure() 関数
 *
 * 説明: RPN を解析して, 結果集合の Query を含むかを判定する。
 * 返値: (int)
 *         1   : 結果集合を含む。
 *         0   : 結果集合を含まない。
 * 引数:
 *         (Z_RPNStructure *) rpn : 解析すべき RPN の検索式
 */
int parse_rpn_structure(Z_RPNStructure *rpn)
{
  switch(rpn->which) {
  case Z_RPNStructure_simple:
    if (rpn->u.simple->which == Z_Operand_APT) {
      return 0;
    } else {
      return 1;
    }
    break;
  case Z_RPNStructure_complex:
    if (parse_rpn_structure(rpn->u.complex->s1)) {
      return 1;
    } else {
      return parse_rpn_structure(rpn->u.complex->s2);
    }
    break;
  default:     /* don't go to here from anywhere. */
    break;
  }
}

/*
 * rpn_to_namazu_query() 関数
 *
 * 説明: RPN の検索式を解析して,Namazu 用の検索式に変換する。
 * 返値: (void)
 * 引数:
 *         (Z_RPNStructure *)rpn : 解析すべき検索式。
 *        (char *) query : Namazu 用の検索文字列。
```

```
 *        (bend_searchresult *) r : Z39.50での検索結果(エラー処理用)
 */
void rpn_to_namazu_query(Z_RPNStructure *rpn, char *query, bend_searchresult *r)
{
  char left_query[BUFSIZE], right_query[BUFSIZE], operator[BUFSIZE];

  switch (rpn->which) {
  case Z_RPNStructure_complex:
    rpn_to_namazu_query(rpn->u.complex->s1, left_query, r);

    rpn_to_namazu_query(rpn->u.complex->s2, right_query, r);

    get_operator(operator, rpn->u.complex->roperator);

    sprintf(query, "( %s %s %s )", left_query, operator, right_query);
    break;
  case Z_RPNStructure_simple:
    get_operand(query, rpn->u.simple, r);
    break;
  default:
    strcpy(query, "!!!UNKNOWN-RPN!!!");
    logf(LOG_FATAL, "Unknown RPN Structure");
    break;
  }
}

/*
 * 名前: longvowel_to_hyphen() 関数
 *
 * 説明: query中の長音(ー)を全てハイフン(−)に置換する。
 *         ※ EUC以外のコードが来ると結果は不定なので注意!
 *         Japan/MARCデータがもともと半角カナで記述されているため。
 *         (いわゆる「インターネット」問題への措置)
 * 返値: (void)
 * 引数:
 *        char *query : 検索式
 */
void longvowel_to_hyphen(char *query)
{
  int i, len = strlen(query);

  for (i = 0; i < len; i++) {
    if ((*(query+i) == '\xa1') && (*(query+i+1) == '\xbc')) {
      *(query+i) = '\xa1';
      *(query+i+1) = '\xdd';
      i++;
    }
```

```
  }
}

/*
 * 名前: rpn_to_resultset() 関数
 *
 * 説明: RPN を階層的に検索して結果集合を求める。
 * 返値: (void)
 * 引数:
 *        Z_RPNStructure *rpn : RPN
 *        char *resultfile: 結果集合ファイル
 *         bend_searchresult *r : (エラー処理用)
 */
void rpn_to_resultset(Z_RPNStructure *rpn, char *resultfile, bend_searchresult *r)
{
  char cmd_string[BUFSIZE];
  char operator[BUFSIZE];
  char query[BUFSIZE], left_query[BUFSIZE], right_query[BUFSIZE];
  char left_resultfile[BUFSIZE], right_resultfile[BUFSIZE];

  switch (rpn->which) {
  case Z_RPNStructure_complex:

    get_operator(operator, rpn->u.complex->roperator);

    if (parse_rpn_structure(rpn)) {
      sprintf(left_resultfile, "%s.l", resultfile);
      rpn_to_resultset(rpn->u.complex->s1, left_resultfile, r);
      sprintf(right_resultfile, "%s.r", resultfile);
      rpn_to_resultset(rpn->u.complex->s2, right_resultfile, r);
      if (!strcmp(operator, "and")) {
        sprintf(cmd_string, "sort %s %s | uniq -d > %s",
                left_resultfile, right_resultfile, resultfile);
      } else if (!strcmp(operator, "or")) {
        sprintf(cmd_string, "sort %s %s | uniq > %s",
                left_resultfile, right_resultfile, resultfile);
      } else if (!strcmp(operator, "not")) {
        sprintf(cmd_string, "sort %s %s %s | uniq -u > %s",
                left_resultfile, right_resultfile, right_resultfile, resultfile);
      }
      system(cmd_string);
      logf(LOG_LOG, "%s", cmd_string);
    } else {
      rpn_to_namazu_query(rpn->u.complex->s1, left_query, r);
      rpn_to_namazu_query(rpn->u.complex->s2, right_query, r);
      sprintf(query, "( %s %s %s )", left_query, operator, right_query);
      longvowel_to_hyphen(query);
```

```
      sprintf(cmd_string,
              "%s %s \"%s\" %s > %s",
              NAMAZU_PATH, NAMAZU_OPTION, query, NAMAZU_DATABASE, resultfile);
      system(cmd_string);
      logf(LOG_LOG, "%s", cmd_string);
    }
    break;
  case Z_RPNStructure_simple:
    if (parse_rpn_structure(rpn)) {
      sprintf(cmd_string, "ln -s %s-%d.%s %s",
             RESULTFILE, getpid(), rpn->u.simple->u.resultSetId, resultfile);
      system(cmd_string);
      logf(LOG_LOG, "%s", cmd_string);
    } else {
      get_operand(query, rpn->u.simple, r);
      longvowel_to_hyphen(query);
      sprintf(cmd_string,
              "%s %s \"%s\" %s > %s",
              NAMAZU_PATH, NAMAZU_OPTION, query, NAMAZU_DATABASE, resultfile);
      system(cmd_string);
      logf(LOG_LOG, "%s", cmd_string);
    }
    break;
  default:
    logf(LOG_FATAL, "Unknown RPN Structure");
    break;
  }
}

/*
 * 名前: process_rpn_query()
 *
 * 説明: Type1 & Type101 Query (RPN query) を処理する。
 * 返値: (void)
 * 引数:
 *        bend_searchresult *r : Z39.50 の検索結果。
 *        Z_RPNQuery *rpn_query : RPN Query
 *        char *setname : 現在の結果集合の名前
 */
void process_rpn_query(bend_searchresult *r, Z_RPNQuery *rpn_query, char *setname)
{
  char buf[BUFSIZE];
  char resultfile[BUFSIZE];
  int line_num;
  FILE *fp;

  sprintf(resultfile, "%s-%d.%s", RESULTFILE, getpid(), setname);
```

```
  rpn_to_resultset(rpn_query->RPNStructure, resultfile, r);

  /* ヒット数のカウント */
  line_num = 0;
  if ((fp = fopen(resultfile, "r")) == NULL ) {
    r->errcode=1;
    logf(LOG_FATAL, "%s cannot open", resultfile);
    return;
  }
  while (fgets(buf, BUFSIZE, fp) != NULL) {
    line_num++;
  }
  r->hits = line_num;
  logf(LOG_LOG, "%d Hits", r->hits);
  fclose(fp);
}

bend_searchresult *bend_search(void *handle, bend_searchrequest *q, int *fd)
{
    bend_searchresult *r = odr_malloc (q->stream, sizeof(*r));

    r->errcode = 0;
    switch ( q->query->which ) {
    case Z_Query_type_1:
    case Z_Query_type_101:
      process_rpn_query(r, q->query->u.type_1, q->setname);
      break;
    case Z_Query_type_2:
      logf(LOG_FATAL, "Type-2 Query Not Support.");
      r->errcode = 107;
      r->errstring = "Type-2 Query Not Support.";
      break;
    default:
      logf(LOG_FATAL, "Unknown Query");
      r->errcode = 107;
      r->errstring = "Unknown Query";
      break;
    }

    return r;
}

static int atoin (const char *buf, int n)
{
    int val = 0;
    while (--n >= 0)
    {
```

```
        if (isdigit(*buf))
            val = val*10 + (*buf - '0');
        buf++;
    }
    return val;
}

char *marc_read(FILE *inf)
{
    char length_str[5];
    size_t size;
    char *buf;

    if (fread (length_str, 1, 5, inf) != 5)
        return NULL;
    size = atoin (length_str, 5);
    if (size <= 6)
        return NULL;
    if (!(buf = xmalloc (size+1)))
        return NULL;
    if (fread (buf+5, 1, size-5, inf) != (size-5))
    {
        xfree (buf);
        return NULL;
    }
    memcpy (buf, length_str, 5);
    buf[size] = '\0';
    return buf;
}

bend_fetchresult *bend_fetch(void *handle, bend_fetchrequest *q, int *num)
{
  bend_fetchresult *r = odr_malloc (q->stream, sizeof(*r));
  static char *bbb = 0;
  char buf[BUFSIZE];
  char recordfile[BUFSIZE];
  char record[BUFSIZE * 10];
  char resultfilename[BUFSIZE];
  FILE *fp;
  int i;

  r->errcode = 0;
  r->errstring = 0;
  r->basename = DBNAME;
  r->last_in_set = 0;
  if (bbb) {
    xfree(bbb);
```

```
    bbb = 0;
  }

  if (q->format != VAL_SUTRS) {
      r->errcode = 238;
      r->errstring = "SUTRS";
      logf(LOG_DEBUG, "%s : alternative-suggested syntax=SUTRS", diagbib1_str(238));
      return r;
  } else {
      r->format = q->format;
  }

  /* 結果集合のファイルリスト */
  sprintf(resultfilename, "%s-%d.%s", RESULTFILE, getpid(), q->setname);
  if ((fp = fopen(resultfilename, "r")) == NULL) {
    logf(LOG_FATAL, "cannot open %s", resultfilename);
    r->errcode=30;
    sprintf(r->errstring, "%s", q->setname);
    return r;
  }
  for ( i = 0; i < q->number; i++) {
    fgets(buf, BUFSIZE, fp);
  }
  fclose(fp);

  /* 実際のレコードファイル */
  strncpy(recordfile, buf, strlen(buf)-1);
  recordfile[strlen(buf)-1] = '\0';
  if ((fp = fopen(recordfile, "r")) == NULL) {
    logf(LOG_FATAL, "cannot open %s", recordfile);
    r->errcode=14;
    sprintf(r->errstring, "%s.%d", q->setname, q->number);
    return r;
  }
  record[0] = '\0';
  while (fgets(buf, BUFSIZE, fp) != NULL ) {
    strcat(record, buf);
  }
  fclose(fp);

  assert(r->record = bbb = xmalloc(strlen(record)+1));
  strcpy(bbb, record);
  r->len = strlen(record);

  return r;
}
```

```
bend_deleteresult *bend_delete(void *handle, bend_deleterequest *q, int *num)
{
    return 0;
}

/*
 * silly dummy-scan what reads words from a file.
 */
bend_scanresult *bend_scan(void *handle, bend_scanrequest *q, int *num)
{
    bend_scanresult *r = odr_malloc (q->stream, sizeof(*r));
    static FILE *f = 0;
    static struct scan_entry list[200];
    static char entries[200][80];
    int hits[200];
    char term[80], *p;
    int i, pos;

    r->errstring = 0;
    r->entries = list;
    r->status = BEND_SCAN_SUCCESS;
    if (!f && !(f = fopen("dummy-words", "r")))
    {
        perror("dummy-words");
        exit(1);
    }
    if (q->term->term->which != Z_Term_general)
    {
            r->errcode = 229; /* unsupported term type */
        return r;
    }
    if (q->term->term->u.general->len >= 80)
    {
            r->errcode = 11; /* term too long */
        return r;
    }
    if (q->num_entries > 200)
    {
            r->errcode = 31;
        return r;
    }
    memcpy(term, q->term->term->u.general->buf, q->term->term->u.general->len);
    term[q->term->term->u.general->len] = '\0';
    for (p = term; *p; p++)
            if (islower(*p))
            *p = toupper(*p);
```

```
    fseek(f, 0, 0);
    r->num_entries = 0;
    for (i = 0, pos = 0; fscanf(f, " %79[^:]:%d", entries[pos], &hits[pos]) == 2;
        i++, pos < 199 ? pos++ : (pos = 0))
    {
            if (!r->num_entries && strcmp(entries[pos], term) >= 0) /* s-point fnd */
        {
            if ((r->term_position = q->term_position) > i + 1)
            {
                    r->term_position = i + 1;
                r->status = BEND_SCAN_PARTIAL;
            }
            for (; r->num_entries < r->term_position; r->num_entries++)
            {
                    int po;

                po = pos - r->term_position + r->num_entries + 1; /* find pos */
                if (po < 0)
                    po += 200;
                list[r->num_entries].term = entries[po];
                list[r->num_entries].occurrences = hits[po];
            }
        }
        else if (r->num_entries)
        {
            list[r->num_entries].term = entries[pos];
            list[r->num_entries].occurrences = hits[pos];
            r->num_entries++;
        }
        if (r->num_entries >= q->num_entries)
            break;
    }
    if (feof(f))
            r->status = BEND_SCAN_PARTIAL;
    return r;
}

void bend_close(void *handle)
{
    char rm_resultset[BUFSIZE];

    sprintf(rm_resultset, "rm -f %s-%d.*", RESULTFILE, getpid());
    system(rm_resultset);
    return;
}

int main(int argc, char **argv)
```

```
{
    return statserv_main(argc, argv);
}
```

A.2 jmarcsesrver.h

```
/* バッファサイズ */
#define BUFSIZE 10240

/* Namazu の index のあるディレクトリ */
#define NAMAZU_DATABASE "/project/jmarc/"

/* 検索コマンド namazu のパス */
#define NAMAZU_PATH         "/home/masao/namazu/bin/namazu"
/* namazu に与えるオプション引数 */
#define NAMAZU_OPTION        "-uSa"

/* Search 時の結果ファイルの prefix */
#define RESULTFILE "/home/masao/Z39.50/search-result"

/* Database Name */
#define DBNAME "JPMARC"

void process_rpn_query(bend_searchresult *r, Z_RPNQuery *rpn_query, char *setname);
void get_operand(char *term, Z_Operand *o, bend_searchresult *r);
void get_operator(char *query, Z_Operator *o);
int parse_rpn_structure(Z_RPNStructure *rpn);
void longvowel_to_hyphen(char *query);
void rpn_to_namazu_query(Z_RPNStructure *rpn, char *query, bend_searchresult *r);
void rpn_to_resultset(Z_RPNStructure *rpn, char *resultfile, bend_searchresult *r);
```

A.3 jmarcfilter.pl

```
#!/usr/local/bin/perl -w
#
# $Id: jmarcfilter.pl,v 1.2 2000/01/14 10:32:57 masao Exp $
#

use strict;
use Convert::EBCDIC;

my $DEBUG = 0;
$| = 1;

&main();

sub debug_print($) {
    my ($str) = @_;

    if ($DEBUG) {
        print "$str";
    }
}

# JIS X 0208 漢字をエスケープコードを含めて返す。
# 同時に、外字の処理も行なう。
sub escape_kanji($) {
    my ($data) = @_;

    # 一応、長音のローマ字形 (a^i^u^e^o^, etc.) は通常のアルファベットに戻す。
    my %GAIJI = ("\x2a\x23" => "#A", # Ａ
                 "\x2a\x2f" => "#I", # Ｉ
                 "\x2a\x39" => "#U", # Ｕ
                 "\x2a\x2b" => "#E", # Ｅ
                 "\x2a\x34" => "#O", # Ｏ
                 "\x2a\x43" => "#a", # ａ
                 "\x2a\x56" => "#i", # ｉ
                 "\x2a\x6c" => "#u", # ｕ
                 "\x2a\x50" => "#e", # ｅ
                 "\x2a\x5e" => "#o"  # ｏ
                 # "\x22\x31" => "\x21\x21" # ／
                 );

    my $str = '';

    while (length($data)) {
        die "Length is Odd.\n" if (length($data) == 1);
        my $kanji = substr($data, 0, 2);
        $data = substr($data, 2);
```

```
        foreach my $key (keys %GAIJI) {
            if ($kanji eq "$key") {
                $kanji = $GAIJI{$key};
            }
        }

        # その他の外字は全て全角空白にする。
        $kanji =~ s/[\x30-\x7e][\xa1-\xfe]/\x21\x21/;
        $kanji =~ s/[\x29-\x2f][\x21-\x7e]/\x21\x21/;
        $kanji =~ s/[\x22][\x2f-\x68]/\x21\x21/;

        $str .= $kanji;
    }

    # 直前の文字がカタカナ(or 平仮名)の場合、
    ## 例: データベース、あっかんべー etc.
    # マイナス(ー: \x21\x5d)を長音(ー: \x21\x3c)に戻す。
    $str =~ s/([\x24-\x25][\x21-\x76])\x21\x5d/$1\x21\x3c/g;

    # JIS X 0208-1978 ( JIS C 6328 97 ) escape sequence.
    return "\x1b\x24\x40$str\x1b\x28\x42";
}

sub main {
    my @tmp = <>;
    my $contents = join('', @tmp);
    my @records = split(/\x1d/, $contents);

    my $t = new Convert::EBCDIC;

    debug_print("Total $#records records found.\n");

    foreach my $record (@records) {
        my $length = length $record;
        my $label = $t->toascii(substr($record, 0, 24));
        my $data = substr($record, 24);
        my @fields = split(/\x1e/, $data);
        my $directory = $t->toascii(shift(@fields));

        debug_print "Record : $length ($#fields fields)\n";
        debug_print "Label: $label\n";
        debug_print "Directory: $directory\n\n";

        foreach my $field (@fields) {
            if (length($directory) < 12) {
                debug_print "ERROR: Directory is mismatch.\n";
            }
```

```
            my $field_id = substr($directory, 0, 3);
            my $field_len = substr($directory, 3, 4)+0;
            my $field_start = substr($directory, 7, 5)+0;
            $directory = substr($directory, 12);

            debug_print "ID: $field_id, Length: $field_len, Offset: $field_start\n";
            if ($field !~ /^\x1f/) {
                print "$field_id  ".$t->toascii($field)."\n";
            } else {                # This field is SubField
                my @subfields = split(/\x1f/, $field);
                debug_print "Subfield: $#subfields\n";
                foreach my $subfield (@subfields) {
                    if (length($subfield) < 5) {
                        debug_print "ERROR: Length of subfield ($subfield) is mismatch.\n";
                        next;
                    }
                    my $sub_id = $t->toascii(substr($subfield, 0, 1));
                    my $mode = $t->toascii(substr($subfield, 4, 1));
                    $subfield = substr($subfield, 5, length($subfield));
                    if ($mode eq "1") {
                        print "$field_id \$$sub_id  ".$t->toascii($subfield)."\n";
                    } elsif ($mode eq "2") {
                        print "$field_id \$$sub_id  ".escape_kanji($subfield)."\n";
                    } else {
                        debug_print "mode recognized mismatch: $mode\n";
                    }
                }
            }
#            print "\n";
        }
        print "\n";
    }
}
```

A.4 jmarc-namazu-2.1.patch

```
diff -ruN namazu-1.4.0.0-beta-8.orig/ChangeLog.jmarc namazu-1.4.0.0-beta-8/ChangeLog.jmarc
--- namazu-1.4.0.0-beta-8.orig/ChangeLog.jmarc
+++ namazu-1.4.0.0-beta-8/ChangeLog.jmarc        Thu Sep  9 21:32:27 1999
@@ -0,0 +1,7 @@
+Thu Sep  9 20:48:47 1999  Masao Takaku  <masao@ulis.ac.jp>
+
+[jmarc-namazu-2.1]
+
+        * lib/filter.pl: 部分的に著者名を抽出できていないバグを直した。
+
+        * namazu-1.4.0.0-beta-8 をベースに作り直した。
diff -ruN namazu-1.4.0.0-beta-8.orig/Makefile.am namazu-1.4.0.0-beta-8/Makefile.am
--- namazu-1.4.0.0-beta-8.orig/Makefile.am        Sat May  1 17:15:51 1999
+++ namazu-1.4.0.0-beta-8/Makefile.am        Thu Sep  9 20:55:18 1999
@@ -1,7 +1,7 @@
 ## Process this file with automake to produce Makefile.in

 @SET_MAKE@
-EXTRA_DIST = README-ja
+EXTRA_DIST = README-ja README.jmarc
 EXTRA_DIRS = lib
 # EXTRA_DIRS = contrib doc lib misc
 SUBDIRS = src
diff -ruN namazu-1.4.0.0-beta-8.orig/Makefile.in namazu-1.4.0.0-beta-8/Makefile.in
--- namazu-1.4.0.0-beta-8.orig/Makefile.in        Mon May  3 17:52:36 1999
+++ namazu-1.4.0.0-beta-8/Makefile.in        Thu Sep  9 20:52:10 1999
@@ -74,7 +74,7 @@
 SCORING = @SCORING@
 VERSION = @VERSION@
 ZCAT = @ZCAT@
-EXTRA_DIST = README-ja
+EXTRA_DIST = README-ja README.jmarc
 EXTRA_DIRS = lib
 # EXTRA_DIRS = contrib doc lib misc
 SUBDIRS = src
diff -ruN namazu-1.4.0.0-beta-8.orig/README.jmarc namazu-1.4.0.0-beta-8/README.jmarc
--- namazu-1.4.0.0-beta-8.orig/README.jmarc
+++ namazu-1.4.0.0-beta-8/README.jmarc        Thu Sep  9 20:44:13 1999
@@ -0,0 +1,23 @@
+README for namazu-jmarc
+
+この配布パッチは全文検索システム Namazu を Japan/MARC の検索に利用するため
+のパッチです。
+
+現在、namazu-1.4.0.0-beta-8 に対する差分を配布しています。
+まず、http://openlab.ring.gr.jp/namazu/などから namazu-1.4.0.0-beta-8.tar.gz
```

```
+を取得して下さい
+次のようにパッチをあてます。
+
+% gzip -cd namazu-1.4.0.0-beta-8.tar.gz | tar xvf -
+% cd namazu-1.4.0.0-beta-8
+% patch -p1 < jmarc-namazu-2.1.patch
+
+インデクサ mknmz には、Japan/MARC のアクセスポイント情報などを抽出する
+ための -j オプションが追加されます。
+検索コマンド namazu は、フィールド名の置換えを常に抑制します。
+
+※ なお、インデックスの対象となる Japan/MARC レコードは安斎さん作の
+jmarcfilter で ISO-2022 に準拠した形でテキスト化し、一レコードは一ファイ
+ルになっていることが前提です。
+--
+たかくまさお（高久雅生）/ masao@ulis.ac.jp / 1999
diff -ruN namazu-1.4.0.0-beta-8.orig/configure namazu-1.4.0.0-beta-8/configure
--- namazu-1.4.0.0-beta-8.orig/configure        Mon May  3 17:51:48 1999
+++ namazu-1.4.0.0-beta-8/configure        Thu Sep  9 20:50:39 1999
@@ -707,9 +707,9 @@
 fi


-PACKAGE=namazu
+PACKAGE=jmarc-namazu

-VERSION=1.4.0.0-beta-8
+VERSION=2.1

 if test "`cd $srcdir && pwd`" != "`pwd`" && test -f $srcdir/config.status; then
   { echo "configure: error: source directory already configured; run "make distclean" there
diff -ruN namazu-1.4.0.0-beta-8.orig/configure.in namazu-1.4.0.0-beta-8/configure.in
--- namazu-1.4.0.0-beta-8.orig/configure.in        Mon May  3 17:51:37 1999
+++ namazu-1.4.0.0-beta-8/configure.in        Thu Sep  9 20:50:21 1999
@@ -9,7 +9,7 @@

 dnl Process this file with autoconf to produce a configure script.
 AC_INIT(src/cgi.c)
-AM_INIT_AUTOMAKE(namazu, 1.4.0.0-beta-8)
+AM_INIT_AUTOMAKE(jmarc-namazu, 2.1)

 dnl Do you wish?
 dnl PATH=/bin:/usr/bin:/usr/sbin:/usr/local/bin:$PATH
diff -ruN namazu-1.4.0.0-beta-8.orig/lib/conf.pl.in namazu-1.4.0.0-beta-8/lib/conf.pl.in
--- namazu-1.4.0.0-beta-8.orig/lib/conf.pl.in        Mon May  3 17:29:20 1999
+++ namazu-1.4.0.0-beta-8/lib/conf.pl.in        Thu Sep  9 20:39:31 1999
@@ -68,6 +68,7 @@
```

```
       -O (dir) : インデックスファイルの出力先を指定する
       -T (dir) : NMZ.{head,foot,body}.* のディレクトリを指定する
       -t (regex): 対象ファイルの正規表現を指定する
+      -j: JPMARC のフィールド情報などを抽出する

 EOFusage

@@ -108,6 +109,7 @@
       -O (dir) : specify a directory to output the index
       -T (dir) : specify a directory where NMZ.{head,foot,body}.* are
       -t (regex): specify a regex for target files
+      -j: extract JPMARC fields

 EOFusage

@@ -369,6 +371,7 @@
 $NoDeleteProcessing = 0;
 $NoUpdateProcessing = 0;
 $HtaccessExcludeOpt = 0;
+$JPMARCOpt = 0;

 %CheckPoint = ("on" => undef, "continue" => undef, "pid" => $$, "fid" => 0);

diff -ruN namazu-1.4.0.0-beta-8.orig/lib/filter.pl namazu-1.4.0.0-beta-8/lib/filter.pl
--- namazu-1.4.0.0-beta-8.orig/lib/filter.pl        Mon May  3 16:37:03 1999
+++ namazu-1.4.0.0-beta-8/lib/filter.pl        Thu Sep  9 21:30:44 1999
@@ -52,6 +52,7 @@
         mailnews_filter($contents, $weighted_str, $title, $fields);
         mailnews_citation_filter($contents, $weighted_str);
     }
+    jmarc_filter($contents, $fields) if $conf::JPMARCOpt;
     line_adjust_filter($contents) unless $conf::NoLineAdOpt;
     line_adjust_filter($weighted_str) unless $conf::NoLineAdOpt;
     white_space_adjust_filter($contents);
@@ -67,6 +68,10 @@
     util::dprint("-- content --\n$$contents\n");
     util::dprint("-- weighted_str: --\n$$weighted_str\n");
     util::dprint("-- headings --\n$$headings\n");
+    util::dprint("-- fields --\n");
+    foreach my $key (keys %{$fields}) {
+        util::dprint("-- field : $key --\n $fields->{$key}\n");
+    }
 }

 # Adjust white spaces
@@ -434,6 +439,75 @@
 sub analize_rcs_stamp()
```

```
 {
 }
+
+# For masao's own use.
+# JMARC filter
+sub jmarc_filter($\%) {
+    my ($contents, $fields) = @_;
+    my ($line, @tmp);
+
+    @tmp = split(/\n/, $$contents);
+    while (@tmp) {
+       $line = shift(@tmp);
+#      print "Line: " . $line . "\n";
+       if ( $line =~ /^[\n]/ ) {            # レコードの区切り。
+           next;
+       } elsif ( $line =~ /^001(.*)/ ) { # ユニークな番号が入っている
+           $$contents = $1;
+#          print "ID : " . $1  . "\n";
+       } elsif ( $line =~ /^010\$A(.*)$/ ) { # ISBN (1=7)
+           $$contents .= " " . $1;
+           $fields->{isbn} .= $1 . " ";
+#          print "Title : " . $1;
+       } elsif ( $line =~ /^55[1-9]\$[AXD](.*)$/ || $line =~ /^25[1-9]\$[ABD](.*)$/ ) {
+           # 書名 Title (1=4)
+           # 251 ～ 259 : 記述フィールド
+           #              $A : 書名
+           #              $B : 副書名
+           #              $D : 巻次等
+           # 551 ～ 559 : 書名アクセスポイント
+           #              $A : カタカナ形
+           #              $X : ローマ字形
+           #              $B : 漢字形（251 フィールドなどを指しているだけ）
+           #              $D : 巻次の読み
+           $$contents .= " " . $1;
+           $fields->{title} .= $1 . " ";
+#          print "Title : " . $1;
+       } elsif ( $line =~ /^7[59][1-9]\$[AXB](.*)$/ ) {
+           # 著者名 Author (1=1003)
+           # 751 ～ 759 : 著者名アクセスポイント
+           #              $A : カタカナ形
+           #              $X : ローマ字形
+           #              $B : 漢字形
+           $$contents .= " " . $1;
+           $fields->{author} .= $1 . " ";
+#          print "Author : " . $1;
+       } elsif ( $line =~ /^65[08]\$[AXB](.*)$/ ) {
+           # 件名 Subject Headings (1=21)
```

```
+           # 650 : 個人名件名標目
+           # 658 : 一般件名標目
+           #       $A : カタカナ形
+           #       $X : ローマ字形
+           #       $B : 漢字形
+           $$contents .= " " . $1;
+           $fields->{subject} .= $1 . " ";
+#          print "Subject Headings :" . $1;
+       } elsif ( $line =~ /^270\$B(.*)$/ ) { # 出版社 Publisher (1=1018)
+           $$contents .= " " . $1;
+           $fields->{publisher} .= $1 . " ";
+#          print "Publisher : " . $1;
+       } elsif ( $line =~ /^3\d{2}\$[AB](.*)$/ ) { # 注記 Note (1=63)
+           $$contents .= " " . $1;
+           $fields->{note} .= $1 . " ";
+#          print "Note : " . $1;
+       } elsif ( $line =~ /^\d{3}\$.(.*)$/ ) {     # その他 Any (1=1016)
+           $$contents .= " " . $1;
+       } else {                        #  not come here
+           print "Err: ". $line . " : cannot extract Japan/MARC field info.\n";
+       }
+    }
+}
+


 1;
diff -ruN namazu-1.4.0.0-beta-8.orig/src/conf.c namazu-1.4.0.0-beta-8/src/conf.c
--- namazu-1.4.0.0-beta-8.orig/src/conf.c        Sat May  1 17:15:57 1999
+++ namazu-1.4.0.0-beta-8/src/conf.c        Thu Sep  9 20:39:34 1999
@@ -42,8 +42,9 @@
   * LOGGING    : %s\n\
   * LANGUAGE   : %s\n\
   * SCORING    : %s\n\
+  * MAXHIT     : %d\n\
 ", DEFAULT_DIR, BASE_URL, Logging ? "ON" : "OFF",
-           Lang, TfIdf ? "TFIDF" : "SIMPLE");
+           Lang, TfIdf ? "TFIDF" : "SIMPLE", IgnoreHit);

     {
         REPLACE list = Replace;
@@ -193,6 +194,9 @@
             for (n = 5; buf[n] == '\t'; n++);
             strncpy(Lang, &buf[n], 2);
             initialize_message();
+        } else if (!strncmp(buf, "MAXHIT\t", 7)) {
+            for (n = 7; buf[n] == '\t'; n++);
```

```
+            IgnoreHit = atoi(&buf[n]);
         }
     }
     fclose(fp);
diff -ruN namazu-1.4.0.0-beta-8.orig/src/hlist.c namazu-1.4.0.0-beta-8/src/hlist.c
--- namazu-1.4.0.0-beta-8.orig/src/hlist.c        Sat May  1 17:15:57 1999
+++ namazu-1.4.0.0-beta-8/src/hlist.c        Thu Sep  9 20:39:35 1999
@@ -333,7 +333,7 @@
             fprintf(stderr, "idf: %f (N:%d, n:%d)\n", idf, AllDocumentN, n/2);
     }

-    if (n >= IGNORE_HIT * 2) {
+    if (n >= IgnoreHit * 2) {
         /* '* 2' means NMZ.i contains a file-ID and a score. */
         hlist.n = TOO_MUCH_HIT;
     } else {
diff -ruN namazu-1.4.0.0-beta-8.orig/src/messages.c namazu-1.4.0.0-beta-8/src/messages.c
--- namazu-1.4.0.0-beta-8.orig/src/messages.c        Sat May  1 17:15:57 1999
+++ namazu-1.4.0.0-beta-8/src/messages.c        Thu Sep  9 20:39:35 1999
@@ -85,7 +85,8 @@
      -F        : <FORM> ... </FORM> の部分を強制的に表示する\n\
      -R        : URL の置き換えを行わない\n\
      -U        : plain text で出力する時に URL encode の復元を行わない\n\
-     -L (lang) : メッセージの言語を設定する ja または en\n";
+     -L (lang) : メッセージの言語を設定する ja または en\n\
+     -u        : フィールド名の置き換えを行わない (Japan/MARC インデックス用)\n";

         /* output messages (Japanese message should be outputed by
            euctojisput function */
@@ -138,7 +139,8 @@
      -F        : force <FORM> ... </FORM> region to output.\n\
      -R        : do not replace URL string.\n\
      -U        : do not decode URL encode when plain text output.\n\
-     -L (lang) : set output language (ja or en)\n";
+     -L (lang) : set output language (ja or en)\n\
+     -u        : unalias field name ( for Japan/MARC index )\n";

         MSG_TOO_LONG_KEY = (uchar *)
             "        <H2>Error!</H2>\n<P>Too long query.</P>\n";
diff -ruN namazu-1.4.0.0-beta-8.orig/src/mknmz.pl.in namazu-1.4.0.0-beta-8/src/mknmz.pl.in
--- namazu-1.4.0.0-beta-8.orig/src/mknmz.pl.in        Mon May  3 17:45:21 1999
+++ namazu-1.4.0.0-beta-8/src/mknmz.pl.in        Thu Sep  9 20:39:36 1999
@@ -580,6 +580,11 @@
         $conf::WAKATI = $conf::CHASEN_MORPH, $conf::MorphOpt = 1 if $argv[0] =~ /m/;
         $conf::UuencodeOpt = 1 if $argv[0] =~ /u/;
         $conf::MailNewsOpt = 1 if $argv[0] =~ /h/;
+        if ($argv[0] =~ /j/) {
```

```
+            $conf::JPMARCOpt = 1;
+            $conf::SEARCH_FIELD = "Title|Subject|ISBN|Author|Publisher|Note";
+            %conf::FIELD_ALIASES = ();
+        }
         if ($argv[0] =~ /r/) {
             $conf::ManOpt      = 1;
             $conf::TARGET_FILE = '.*\.\d.*';
@@ -1703,7 +1708,7 @@
       $conf::MHONARC_MESSAGE_FILE, $conf::DENY_FILE, $conf::INTSIZE,
       $conf::MailNewsOpt, $conf::NoLineAdOpt, $conf::CHASEN_MORPH,
       $conf::UuencodeOpt, $conf::MHONARC_HEADER, $conf::CHASEN,
-      $conf::KAKASI, $conf::OkuriganaOpt, $TARGET_DIR,
+      $conf::KAKASI, $conf::OkuriganaOpt, $TARGET_DIR, $conf::JPMARCOpt,
       $conf::HiraganaOpt, $conf::ROBOTS_EXCLUDE_URLS, $conf::DEFAULT_FILE,
       $conf::HTML_SUFFIX, $conf::USAGE_EN, $conf::NoHeadAbstOpt,
       $conf::SUMMARY_HEADER, $conf::DebugOpt, $conf::NoEncodeURL,
diff -ruN namazu-1.4.0.0-beta-8.orig/src/namazu.h namazu-1.4.0.0-beta-8/src/namazu.h
--- namazu-1.4.0.0-beta-8.orig/src/namazu.h        Sat May  1 17:16:02 1999
+++ namazu-1.4.0.0-beta-8/src/namazu.h        Thu Sep  9 20:45:33 1999
@@ -40,7 +40,6 @@
 #define SCORE_LINE      1        /* スコア表示をする行の番号 */
 #define ABSTRACT_LINE   2        /* 文書の頭の方 (要約) を表示する行の番号 */
 #define PAGE_MAX       20        /* 検索結果出力のページの最大数 */
-#define IGNORE_HIT  10000        /* ヒットした項目がこれより多いと無視する */
 #define IGNORE_MATCH 1000        /* これより単語が多くマッチしたら無視する */
 #define HLIST_MAX_MAX  100        /* 一度につきの結果表示の最大数の最大数 */
 #define DB_MAX          64       /* データベースの最大数 */
@@ -117,6 +116,7 @@
 extern int AllDocumentN;
 extern int TfIdf;
 extern int NoReference;
+extern int IgnoreHit;

 extern uchar KeyTable[];
 extern uchar DbName[];
diff -ruN namazu-1.4.0.0-beta-8.orig/src/re_match.c namazu-1.4.0.0-beta-8/src/re_match.c
--- namazu-1.4.0.0-beta-8.orig/src/re_match.c        Sat May  1 17:15:58 1999
+++ namazu-1.4.0.0-beta-8/src/re_match.c        Thu Sep  9 20:39:36 1999
@@ -65,7 +65,7 @@

     if (field_mode) {
         malloc_hlist(&val, size += STEP);
-        max = IGNORE_HIT;
+        max = IgnoreHit;
         if (!strcmp(field, "url")) {
             url_mode = 1;
         }
```

```
@@ -98,7 +98,7 @@
             }
             if (!field_mode) {
                 tmp = get_hlist(i);
-                if (tmp.n > IGNORE_HIT) {
+                if (tmp.n > IgnoreHit) {
                     free_hlist(val);
                     val.n = -1;
                     break;
@@ -115,7 +115,7 @@
             if (!field_mode) {
                 val = ormerge(val, tmp);
             }
-            if (val.n > IGNORE_HIT) {
+            if (val.n > IgnoreHit) {
                 free_hlist(val);
                 val.n = -1;
                 break;
diff -ruN namazu-1.4.0.0-beta-8.orig/src/search.c namazu-1.4.0.0-beta-8/src/search.c
--- namazu-1.4.0.0-beta-8.orig/src/search.c        Sat May  1 17:16:01 1999
+++ namazu-1.4.0.0-beta-8/src/search.c        Thu Sep  9 20:39:37 1999
@@ -185,13 +185,13 @@
         chop(buf);
         if (!strncmp(key, buf, n)) {
             tmp = get_hlist(i);
-            if (tmp.n > IGNORE_HIT) {
+            if (tmp.n > IgnoreHit) {
                 free_hlist(val);
                 val.n = TOO_MUCH_MATCH;
                 break;
             }
             val = ormerge(val, tmp);
-            if (val.n > IGNORE_HIT) {
+            if (val.n > IgnoreHit) {
                 free_hlist(val);
                 val.n = TOO_MUCH_MATCH;
                 break;
@@ -575,6 +575,8 @@
     }
     *tmp = '\0';

+/* この部分はJAPAN/MARCの検索では不要なのでコメントにした。 */
+/*
     if (strcmp(field, "title") == 0) {
         strcpy(field, "subject");
     } else if (strcmp(field, "author") == 0) {
@@ -582,6 +584,7 @@
```

```
     } else if (strcmp(field, "path") == 0) {
         strcpy(field, "url");
     }
+*/
 }

 void get_expr(uchar *expr, uchar *str)
@@ -884,6 +887,11 @@
     int i;

     strcpy(query_orig, query); /* save */
+
+    /* JAPAN/MARC を変換するときの EBCDIC コードの扱いの都合上、いったん
+        'ー' を '-' に一括置換する。 */
+    longvowel_to_hyphen(query);
+
     split_query(query);

     if (!HitCountOnly && !MoreShortFormat && !NoReference && !Quiet) {
diff -ruN namazu-1.4.0.0-beta-8.orig/src/util.c namazu-1.4.0.0-beta-8/src/util.c
--- namazu-1.4.0.0-beta-8.orig/src/util.c        Sat May  1 17:16:02 1999
+++ namazu-1.4.0.0-beta-8/src/util.c        Thu Sep  9 20:39:37 1999
@@ -241,3 +241,27 @@
     return tmp;
 }

+/*
+ * 名前: longvowel_to_hyphen() 関数
+ *
+ * 説明: query 中の長音（ー）を全てハイフン（−）に置換する。
+ *       ※ EUC 以外のコードが来ると結果は不定なので注意！
+ *       Japan/MARC データがもともと半角カナで記述されているため。
+ *       （いわゆる「インターネット」問題への措置）
+ * 返値: (void)
+ * 引数:
+ *      char *query : 検索式
+ */
+void longvowel_to_hyphen(char *query)
+{
+  int i, len = strlen(query);
+
+  for (i = 0; i < len; i++) {
+    if ((*(query+i) == '\xa1') && (*(query+i+1) == '\xbc')) {
+      *(query+i) = '\xa1';
+      *(query+i+1) = '\xdd';
+      i++;
+    }
```

```
+  }
+  zen2han(query);
+}
diff -ruN namazu-1.4.0.0-beta-8.orig/src/util.h namazu-1.4.0.0-beta-8/src/util.h
--- namazu-1.4.0.0-beta-8.orig/src/util.h        Sat May  1 17:16:02 1999
+++ namazu-1.4.0.0-beta-8/src/util.h        Thu Sep  9 20:39:37 1999
@@ -39,6 +39,7 @@
 void decode_url_string();
 void tolower_string();
 void delete_backslashes();
+void longvowel_to_hyphen(char *query);        /* For JAPAN/MARC */
 int get_unpackw();
 int read_unpackw();

diff -ruN namazu-1.4.0.0-beta-8.orig/src/values.c namazu-1.4.0.0-beta-8/src/values.c
--- namazu-1.4.0.0-beta-8.orig/src/values.c        Sat May  1 17:16:02 1999
+++ namazu-1.4.0.0-beta-8/src/values.c        Thu Sep  9 20:39:38 1999
@@ -96,6 +96,7 @@
 #endif

 int AllDocumentN = 0;  /* number of all of documents */
+int IgnoreHit = 10000;        /* MAXHIT */

 uchar KeyTable[BUFSIZE];                /* table which saving query */
 uchar *KeyItem[KEY_ITEM_MAX + 1];        /* pointers of items of query */
```

付 録B　Z39.50-DCシステム

B.1　jmarc2dc.pl

```
#!/usr/local/bin/perl -w
#
# This program convert JAPAN/MARC raw data to Dublin Core/RDF data.
#

use strict;
use Convert::EBCDIC;
use Cwd;
use IO::File;
use Getopt::Long;

my $DEBUG = 0;
$| = 1;

# 出力ファイルのディレクトリ
my $OUTDIR = cwd();

# 複数の書誌項目を Bag で Container 化するか?
## -B or --bag で制御する。
my $BagOpt = 0;

main();

sub debug_print($) {
    my ($str) = @_;

    if ($DEBUG) {
        print "$str";
    }
}

# JIS X 0208 漢字をエスケープコードを含めて返す。
# 同時に、外字の処理も行なう。
# 一応、長音のローマ字形 (a^i^u^e^o^, etc.) はアルファベットに戻す。
sub escape_kanji($) {
    my ($data) = @_;
    my %GAIJI = ("\x2a\x23" => "#A", # A
```

```
                "\x2a\x2f" => "#I", # Ⅰ
                "\x2a\x39" => "#U", # Ⅴ
                "\x2a\x2b" => "#E", # Ⅹ
                "\x2a\x34" => "#O", # Ⅼ
                "\x2a\x43" => "#a", # ⅰ
                "\x2a\x56" => "#i", # ⅱ
                "\x2a\x6c" => "#u", # ⅲ
                "\x2a\x50" => "#e", # ⅳ
                "\x2a\x5e" => "#o", # ⅴ
                "\x22\x31" => "\x21\x21" # ／
                );

    my $str = '';

    while (length($data)) {
        die "Length is Odd.\n" if (length($data) == 1);
        my $kanji = substr($data, 0, 2);
        $data = substr($data, 2);
        foreach my $key (keys %GAIJI) {
            if ($kanji eq "$key") {
                $kanji = $GAIJI{$key};
            }
        }
        $kanji =~ s/[\x30-\x7e][\xa1-\xfe]/\x21\x21/;
        $kanji =~ s/[\x29-\x2f][\x21-\x7e]/\x21\x21/;
        $kanji =~ s/[\x22][\x2f-\x68]/\x21\x21/;

        $str .= $kanji;
    }

    # 直前の文字がカタカナの場合、
    # 平仮名も追加。(例: あっかんべー)
    # マイナス (ー: \x21\x5d) を長音 (ー: \x21\x3c) に戻す。
    $str =~ s/([\x24-\x25][\x21-\x76])\x21\x5d/$1\x21\x3c/g;

    # JIS X 0208-1978 ( JIS C 6328 97 ) escape sequence.
    return "\x1b\x24\x40$str\x1b\x28\x42";
}

sub register_data($$$$) {
    my ($hashref, $fid, $subid, $string) = @_;

    my $name = "$fid\$$subid";
    $name = "$fid" if ($subid eq '');

    if (defined $hashref->{$name}) {
        $hashref->{$name} .= " ".$string;
```

```
    } else {
        $hashref->{$name} = $string;
    }
}

sub parse_options() {
    Getopt::Long::config('bundling');
    GetOptions(
               'O|outdir=s'        => \$OUTDIR,
               'B|bag'                => \$BagOpt,
               'd|debug'        => \$DEBUG,
               );
}

sub main {
    # read JAPAN/MARC data from stdin.
    my $contents = join('', <>);
    my @records = split(/\x1d/, $contents);

    my $t = new Convert::EBCDIC;

    debug_print("Total $#records records found.\n");

    parse_options();
    print "File output directory is ... $OUTDIR\n";

    foreach my $record (@records) {
        my %data = ();
        my $length = length $record;
        my $label = $t->toascii(substr($record, 0, 24));
        my @fields = split(/\x1e/, substr($record, 24));
        my $directory = $t->toascii(shift(@fields));

        debug_print "Record : $length ($#fields fields)\n";
        debug_print "Label: $label\n";
        debug_print "Directory: $directory\n\n";

        foreach my $field (@fields) {
            if (length($directory) < 12) {
                debug_print "ERROR: Directory is mismatch.\n";
            }
            my $field_id = substr($directory, 0, 3);
            my $field_len = substr($directory, 3, 4)+0;
            my $field_start = substr($directory, 7, 5)+0;
            $directory = substr($directory, 12);

            debug_print "ID: $field_id, Length: $field_len, Offset: $field_start\n";
```

```
            if ($field !~ /^\x1f/) {
#                print "$field_id  ".$t->toascii($field)."\n";
                register_data(\%data, $field_id, '', $t->toascii($field));
            } else {                # This field is SubField
                my @subfields = split(/\x1f/, $field);
                debug_print "Subfield: $#subfields\n";
                foreach my $subfield (@subfields) {
                    if (length($subfield) < 5) {
                        debug_print "ERROR: Length of subfield ($subfield) is mismatch.\n";
                        next;
                    }
                    my $sub_id = $t->toascii(substr($subfield, 0, 1));
                    my $mode = $t->toascii(substr($subfield, 4, 1));
                    $subfield = substr($subfield, 5, length($subfield));
                    if ($mode eq "1") {
#                        print "$field_id \$$sub_id  ".$t->toascii($subfield)."\n";
                        register_data(\%data, $field_id, $sub_id, $t->toascii($subfield));
                    } elsif ($mode eq "2") {
#                        print "$field_id \$$sub_id  \x1b\x24\x42".$subfield."\x1b\x28\x42\n
                        my $kanji = escape_kanji($subfield);
                        #my $kanji = "\x1b\x24\x42$subfield\x1b\x28\x42";
                        register_data(\%data, $field_id, $sub_id, $kanji);
                    } else {
                        debug_print "mode recognized mismatch: $mode\n";
                    }
                }
            }
#            print "\n";
        }
        dc_write(%data);
#        print "\n";
    }
}

# 実際に Dublin Core データを書き出す。
sub dc_write(%) {
    my (%data) = @_;

    my %DC_MAP = ('010$A' => 'Identifier',
                  '251$A' => 'Title',
                  '251$F' => 'Creator',
                  '270$B' => 'Publisher',
                  '101$A' => 'Language',
                  '270$D' => 'Date');

    # DC element names have to be lower case. (eg. title, not 'Title')
    my %DC_MAP2 = ('title' => ['251$A', '251$B', '251$D', # タイトル
```

```
'252$A', '252$B', '252$D', # タイトル
'253$A', '253$B', '253$D', # タイトル
'254$A', '254$B', '254$D', # タイトル
'255$A', '255$B', '255$D', # タイトル
'256$A', '256$B', '256$D', # タイトル
'257$A', '257$B', '257$D', # タイトル
'258$A', '258$B', '258$D', # タイトル
'259$A', '259$B', '259$D', # タイトル
'280$A', '280$B', '280$D', '280$F', # 叢書名
'281$A', '281$B', '281$D', '281$S', '281$T', '281$X', # シ
リーズ
'282$A', '282$B', '282$D', '283$S', '282$T', '282$X', # シ
リーズ
'283$A', '283$B', '283$D', '282$S', '283$T', '283$X', # シ
リーズ
'291$A', '291$B', '291$D', # 多巻ものの各巻のタイ
トル
'292$A', '292$B', '292$D', # 多巻ものの各巻のタイ
トル
'293$A', '293$B', '293$D', # 多巻ものの各巻のタイ
トル
'294$A', '294$B', '294$D', # 多巻ものの各巻のタイ
トル
'295$A', '295$B', '295$D', # 多巻ものの各巻のタイ
トル
'296$A', '296$B', '296$D', # 多巻ものの各巻のタイ
トル
'297$A', '297$B', '297$D', # 多巻ものの各巻のタイ
トル
'298$A', '298$B', '298$D', # 多巻ものの各巻のタイ
トル
'299$A', '299$B', '299$D', # 多巻ものの各巻のタイ
トル
'354$A', # 原タイトル注記
'551$A', '551$X', # タイトル標目
'552$A', '552$X', # タイトル標目
'553$A', '553$X', # タイトル標目
'554$A', '554$X', # タイトル標目
'555$A', '555$X', # タイトル標目
'556$A', '556$X', # タイトル標目
'557$A', '557$X', # タイトル標目
'558$A', '558$X', # タイトル標目
'559$A', '559$X', # タイトル標目
'580$A', '580$X', # 叢書名標目
'581$A', '581$X', # シリーズのタイトル標目
'582$A', '582$X', # シリーズのタイトル標目
'583$A', '583$X', # シリーズのタイトル標目
```

```
            '591$A', '591$X', # 多巻ものの各巻のタイトル標目
            '592$A', '592$X', # 多巻ものの各巻のタイトル標目
            '593$A', '593$X', # 多巻ものの各巻のタイトル標目
            '594$A', '594$X', # 多巻ものの各巻のタイトル標目
            '595$A', '595$X', # 多巻ものの各巻のタイトル標目
            '596$A', '596$X', # 多巻ものの各巻のタイトル標目
            '597$A', '597$X', # 多巻ものの各巻のタイトル標目
            '598$A', '598$X', # 多巻ものの各巻のタイトル標目
            '599$A', '599$X'  # 多巻ものの各巻のタイトル標目
           ],
'creator' => ['251$F', # 責任表示
              '252$F', # 責任表示
              '253$F', # 責任表示
              '254$F', # 責任表示
              '255$F', # 責任表示
              '256$F', # 責任表示
              '257$F', # 責任表示
              '258$F', # 責任表示
              '259$F', # 責任表示
              '751$A', '751$B', '751$X', # 著者標目
              '752$A', '752$B', '752$X', # 著者標目
              '753$A', '753$B', '753$X', # 著者標目
              '754$A', '754$B', '754$X', # 著者標目
              '755$A', '755$B', '755$X', # 著者標目
              '756$A', '756$B', '756$X', # 著者標目
              '757$A', '757$B', '757$X', # 著者標目
              '758$A', '758$B', '758$X', # 著者標目
              '759$A', '759$B', '759$X'  # 著者標目
             ],
'subject' => ['650$A', '650$B', '650$X', # 個人件名
              '658$A', '658$B', '658$X', # 一般件名
              '677$A', # NDC 分類記号
              '685$A', '685$X' # NDC 分類記号 (カナ・ローマ字付)
             ],
'description' => ['350$A', # 一般注記
                  '377$A'  # 内容注記
                 ],
'publisher' => ['270$B' # 出版者
                ],
'contributor' => ['281$F', # シリーズに関する責任表示
                  '282$F', # シリーズに関する責任表示
                  '283$F', # シリーズに関する責任表示
                  '291$F', # 多巻ものの各巻の責任表示
                  '292$F', # 多巻ものの各巻の責任表示
                  '293$F', # 多巻ものの各巻の責任表示
                  '294$F', # 多巻ものの各巻の責任表示
                  '295$F', # 多巻ものの各巻の責任表示
```

```
                                ’296$F’, # 多巻ものの各巻の責任表示
                                ’297$F’, # 多巻ものの各巻の責任表示
                                ’298$F’, # 多巻ものの各巻の責任表示
                                ’299$F’, # 多巻ものの各巻の責任表示
                                ’781$A’, ’781$B’, ’781$X’, # シリーズの著者
標目
                                ’782$A’, ’782$B’, ’782$X’, # シリーズの著者
標目
                                ’783$A’, ’783$B’, ’783$X’, # シリーズの著者
標目
                                ’791$A’, ’791$B’, ’791$X’, # 多巻ものの各巻
著者標目
                                ’792$A’, ’792$B’, ’792$X’, # 多巻ものの各巻
著者標目
                                ’793$A’, ’793$B’, ’793$X’, # 多巻ものの各巻
著者標目
                                ’794$A’, ’794$B’, ’794$X’, # 多巻ものの各巻
著者標目
                                ’795$A’, ’795$B’, ’795$X’, # 多巻ものの各巻
著者標目
                                ’796$A’, ’796$B’, ’796$X’, # 多巻ものの各巻
著者標目
                                ’797$A’, ’797$B’, ’797$X’, # 多巻ものの各巻
著者標目
                                ’798$A’, ’798$B’, ’798$X’, # 多巻ものの各巻
著者標目
                                ’799$A’, ’799$B’, ’799$X’  # 多巻ものの各巻
著者標目
                               ],
                    ’date’ => [’270$D’ # 出版年月
                              ],
                    ’type’ => [],
                    ’format’ => [], # ’275$A’ 形態（資料の種別） :: どこへやるか不明???
                    ’identifier’ => [ # ’001’, :: レコード識別番号（全国書誌番号と
同一なので削除）
                                    ’010$A’, # ISBN
                                    ’020$B’, # 全国書誌番号
                                    ’905$A’, # NDL 請求記号
                                    ’906$A’  # NDL 印刷番号
                                    ],
                    ’source’ => [],
                    ’language’ => [’101$A’  # 著作の言語
                                  ],
                    ’relation’ => [],
                    ’coverage’ => [’270$A’ # 出版地
                                  ],
                    ’rights’ => []);
```

```
    my $DC_HEAD = <<EOF;
<?xml version="1.0" encoding="EUC-JP"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.0/">

<rdf:Description about="">
EOF

    my $DC_FOOT = <<EOF;
</rdf:Description>
</rdf:RDF>
EOF

    my $fh = new IO::File;
    $fh->open("|nkf -e > $OUTDIR/$data{'001'}.rdf") || die "$data{'001'}: $!\n";
    print "open $data{'001'} ... ";

    print $fh $DC_HEAD;
#    foreach my $key (keys %data) {
#        if ($DC_MAP{$key}) {
#            print $fh "  <dc:$DC_MAP{$key}>$data{$key}</dc:$DC_MAP{$key}>\n";
#        }
#    }

    foreach my $element (keys %DC_MAP2) {
        my @tmp = ();
        foreach my $field (@{$DC_MAP2{$element}}) {
            if (defined $data{$field}) {
                # print "$element -> $field: $data{$field}\n";
                push @tmp, $data{$field};
            } else {
                # This field cannot map to DC.
                # so, this field located to JPMARC original element.
                ## print $fh "  <jpmarc:$field>$data{$field}</jpmarc:$field>\n";
                ### This is !!mistake!!
                ### ここを見てもDCにマップされてないフィールドは分からない。
            }
        }
        if ($BagOpt) {
            if ($#tmp == 0) {
                print $fh "  <dc:$element>$tmp[0]</dc:$element>\n";
            } elsif ($#tmp > 0) {
                print $fh "  <dc:$element>\n";
                print $fh "   <rdf:Bag>\n";
                foreach my $item (@tmp) {
                    print $fh "    <rdf:li>$item</rdf:li>\n";
```

```
                }
                print $fh "   </rdf:Bag>\n";
                print $fh "  </dc:$element>\n";
            } else {
                next;
            }
        } else {
            foreach my $item (@tmp) {
                print $fh "  <dc:$element>$item</dc:$element>\n";
            }
        }
    }
    print $fh $DC_FOOT;

    $fh->close;
    print " done.\n";
}
```

B.2　JAPAN/MARC レコード例（テキスト形式）

00198021725
020$AJP
020$B98021725
100$A19980413 1996　　0JPN 1312
251$A 研究会「ＮＤＣ新訂９版の適用をめぐって」
251$B 研究会の記録
251$D 第１回・第２回
251$F 図書館流通センターＰＲ室／編
270$A 東京
270$B 図書館流通センター
270$D １９９６．１０
275$A ２冊（資料編とも）
275$B ３０ｃｍ
350$A 会期　１９９６年6月5日・7月１１日
551$A ケンキュウカイ　ＮＤＣ　シンテイ　9ハン　ノ　テキヨウ　オ　メグッテ
551$X Ｋｅｎｋｙｕｕｋａｉ　《NDC》　ｓｉｎｔｅｉ　9ｈａｎ　ｎｏ　ｔｅｋｉｙｏｏ　ｏ　ｍｅｇｕｔｔｅ
551$B ２５１Ａ１
551$D １
551$A ケンキュウカイ　ノ　キロク
551$X Ｋｅｎｋｙｕｕｋａｉ　ｎｏ　ｋｉｒｏｋｕ
551$B ２５１Ｂ１
658$A ジッシンブンルイホウ
658$X Ｚｉｓｓｉｎｂｕｎｒｕｉｈｏｏ
658$B 十進分類法
677$A ０１４．４５
677$V ９
685$A ＵＬ６５５
751$A トショカン　リュウツウ　センター
751$X Ｔｏｓｙｏｋａｎ　ｒｙｕｕｔｕｕ　ｓｅｎｔａａ
751$B 図書館流通センター
905$A ＵＬ６５３－Ｇ５

B.3　変換後の RDF 表現の Dublin Core メタデータ

```
<?xml version="1.0" encoding="EUC-JP"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.0/">

<rdf:Description about="">
  <dc:subject>
   <rdf:Bag>
    <rdf:li>ジッシンブンルイホウ</rdf:li>
    <rdf:li>十進分類法</rdf:li>
    <rdf:li>Ｚｉｓｓｉｎｂｕｎｒｕｉｈｏ</rdf:li>
    <rdf:li>０１４．４５</rdf:li>
    <rdf:li>ＵＬ６５５</rdf:li>
   </rdf:Bag>
  </dc:subject>
  <dc:creator>
   <rdf:Bag>
    <rdf:li>図書館流通センターＰＲ室　編</rdf:li>
    <rdf:li>トショカン　リュウツウ　センター</rdf:li>
    <rdf:li>図書館流通センター</rdf:li>
    <rdf:li>Ｔｏｓｙｏｋａｎ　ｒｙｕｔｕ　ｓｅｎｔａ</rdf:li>
   </rdf:Bag>
  </dc:creator>
  <dc:title>
   <rdf:Bag>
    <rdf:li>研究会「ＮＤＣ新訂９版の適用をめぐって」</rdf:li>
    <rdf:li>研究会の記録</rdf:li>
    <rdf:li>第１回・第２回</rdf:li>
    <rdf:li>ケンキュウカイ　ＮＤＣ　シンテイ　９ハン　ノ　テキヨウ　オ　メグッテ ケンキュウカイ　ノ　キロク</rdf:li>
    <rdf:li>Ｋｅｎｋｙｕｋａｉ　《NDC》　ｓｉｎｔｅｉ　９ｈａｎ　ｎｏ　ｔｅｋｉｙｏ　ｍｅｇｕｔｔｅ Ｋｅｎｋｙｕｋａｉ　ｎｏ　ｋｉｒｏｋｕ</rdf:li>
   </rdf:Bag>
  </dc:title>
  <dc:identifier>
   <rdf:Bag>
    <rdf:li>98021725</rdf:li>
    <rdf:li>ＵＬ６５３－Ｇ５</rdf:li>
   </rdf:Bag>
  </dc:identifier>
  <dc:description>会期　１９９６年６月５日・７月１１日</dc:description>
  <dc:coverage>東京</dc:coverage>
  <dc:date>１９９６．１０</dc:date>
  <dc:publisher>図書館流通センター</dc:publisher>
</rdf:Description>
</rdf:RDF>
```

B.4 dc.pl

```
#
# -*- Perl -*-
# $Id: dc.pl,v 1.2 2000/01/16 06:53:16 masao Exp $
# Copyright (C) 1997-1999 Satoru Takabayashi ,
#               1999 NOKUBI Takatsugu All rights reserved.
#     This is free software with ABSOLUTELY NO WARRANTY.
#
#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either versions 2, or (at your option)
#  any later version.
#
#  This program is distributed in the hope that it will be useful
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to the Free Software
#  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA
#  02111-1307, USA
#
#  This file must be encoded in EUC-JP encoding
#

package dc;
use strict;
use XML::Parser;
require 'util.pl';
require 'gfilter.pl';

my $DC_URI = 'http://purl.org/dc/elements/1.0/';
my @DC_ELEMENTS = ('title',
                   'creator',
                   'subject',
                   'description',
                   'publisher',
                   'contributor',
                   'date',
                   'type',
                   'format',
                   'identifier',
                   'source',
                   'language',
                   'relation',
                   'coverage',
```

```
                  'rights');
my %dc_fields = ();
my $dc_text = '';

sub mediatype() {
    return ('application/rdf; x-type=dublin_core');
}

sub status() {
    return 'yes';
}

sub recursive() {
    return 0;
}

sub codeconv() {
    return 0;
}

sub filter ($$$$$) {
    my ($orig_cfile, $cont, $weighted_str, $headings, $fields)
      = @_;

    %dc_fields = ();
    $dc_text = '';

    my $parser = new XML::Parser(Namespaces => 1);
    $parser->setHandlers(Char => \&handle_char);
    $parser->parse($$cont);

    $$cont = $dc_text;
    foreach my $key (keys %dc_fields) {
        util::dprint "DC :: $key: $dc_fields{$key}\n";
        $fields->{$key} = $dc_fields{$key};
    }

    gfilter::white_space_adjust_filter($cont);
    gfilter::show_filter_debug_info($cont, $weighted_str,
                                    $fields, $headings);
    return undef;
}

sub handle_char($$)
{
    my ($p, $str) = @_;
```

```
    my $tmpfile = util::tmpnam('NMZ.dc');
    my $utfconvpath = util::checkcmd('lv');
    my $fh = util::efopen("| $utfconvpath -Iu8 -Oej > $tmpfile");
    print $fh $str;
    undef $fh;

    $fh = util::efopen("< $tmpfile");
    $str = util::readfile($fh);
    undef $fh;
    unlink($tmpfile);

    # Original mknmz do this...
    codeconv::toeuc(\$str);

    util::dprint "Tag is ".$p->current_element;
    util::dprint "  (".$p->generate_ns_name($p->current_element, $DC_URI).")\n";
    foreach my $element (@DC_ELEMENTS) {
        my $this_name = $p->generate_ns_name($element, $DC_URI);
        if ($p->within_element($this_name)) {
            # util::dprint "$element : $str\n";
            $dc_fields{$element} .= " $str";
        }
    }
    $dc_text .= " $str";
}

1;
```