人文科学とコンピュータ　49－7
（２００１．１．１９）

Web 公開のためのデータベース・エンジニアリング

及川昭文 †，山元啓史 ††

† 総合研究大学院大学，†† カリフォルニア大学サンディエゴ校

現在ホームページ上で，一般的な DBMS の類を利用することなく貝塚データベースを公開している。これは，データベース中の各レコードを 1 つの HTML ファイルとして格納し，検索や表示のための一群のプログラムを開発，実装することによって，従来の DBMS と同等の性能，機能を実現したものである。これまでのように使用する DBMS についての詳しい知識がなくても，ワープロ・ファイルやエクセル等の表形式ファイルを，簡単な手続きと手順で，Web 上にデータベースとして格納し，利用することができる仕組みを作り上げた。また，HTML の持っているリンク機能等を活用して，従来の DBMS では実現が困難であった機能も実現することができた。これらの一連の手法，手順，機能について貝塚データベースを例にして報告する。

The Kaizuka (Shell Mound) Database, which doesn't require DBMS, is now available to the public on the web. A user can retrieve and display any record from this database with almost the same performance and functions of an ordinary DBMS. We have designed these features so that each record arranges as one HTML file, and a series of tiny programs makes it possible to access simple tables for user search requests. Anyone with little knowledge about DBMS can upload files, which are recorded as a word-processor file or a spreadsheet file, to the web server as a unit of the database. We have also made external access, which is not possible under an ordinary DBMS, possible by using hyper-links which the HTML files contain. In this paper, a series of techniques, procedures, and features of the Kaizuka Database will be shown.

はじめに

コンピュータが計算する機械としてだけではなく，情報を処理する機械へと進化するとともに現れてきた DBMS という概念が大きく変わろうとしている。それはインターネットに代表される一連のネットワーク技術と，html という情報処理言語の出現によって引き起こされつつある。DBMS の進化をたどれば，まずホストコンピュータ，やがてワークステーション，そしてパソコンと，そのハードウェア環境はダウンサイジングされてきており，同時に運用も集団から個人へ，専門家から非専門家へとよりユーザーフレンドリーなものへと変化しつつある。以前は専門的な知識と十分な経験がなければ，DBMS を使いこなせなかったのが，ある程度の知識や技術があれば何とか Access 等に代表されるデータベース（以下，DB）ソフトを使えるような状況になってきている。

ただ，このある程度が問題である。筆者の勝手な憶測であるが，ワープロソフトにしろ DB ソフトにしろ，90%以上のユーザは，これらのソフトが有している機能の 20%も利用していないのではないかと思われる。すなわち，これらのソフトをフルに使いこなそうと思ったら，以前と同じように十分な知識と経験，そして技術を身につけなければ，それは叶わないというのが現実であろう。

初心者は初心者なりに使えるということは，確かに進歩といえるが，それはあくまで自分だけが利用する場合である。DB をサーバに格納し，一般に広く公開し，日常的な保守も行おうとしたら，DB ソフトだけでなく，サーバやネットワークについての十分な知識や技術が必要になり，そのようなユーザはほとんどいなくなってしまう。自分の持っている DB を広く公開したくてもできないでいるユーザでも，簡単な手続きと手順で Web 上に DB を公開できる BB-DB Engineering を提案する。

1. BB-DB Engineering －その概念－

BB とは Bare Born の略で，simple resource すなわち，できるだけ単純な形態を保ちながら DB 化するということである。また，ここでいうエンジニアリングとは，情報工学的なことだけを指すのではなく，1 次資料から DB を作成し，Web で公開するに至る過程に必要な情報技術，技法（手法），手順等を総称したものとして使っている。

1.1 何が問題なのか

多くの有用な学術コンテンツが存在しているにもかかわらず，なかなか研究室から発信されてこない大きな理由としては，3 つのことが考えられる。

技術の問題：永らく紙と鉛筆の世界で過ごしてきた人文系研究者にとって，ワープロは何とか使うことができても，DB までは手がでないということがある。あるいは，何とか DB 化はできても，それを Web で公開するだけの知識も技術もないし，そこまで学習する時間も意欲もないということがある。

人文系学問固有の問題：多くの人文系学問の場合，1 次資料（史料）の収集，データシートの作成に膨大な時間と人手を要し，苦労して作った DB をそう簡単に公開したくないという心情的な要素があり，なかなか DB 公開に至らないという側面は否定できない。

質と量の問題：DB を公開する以上は，それに値するだけの質，量が備わっていなければならない。とくにデータの Quality Control は，その利用価値を左右するものであり，十分な配慮が必要である。しかし，一般的な DBMS は Quality Control について無関心で，全ては作成者の責任になっている。この負担は研究者にとっては非常に大きいものがある。

これらの問題は，Web 公開に至るまでの 2 つの障壁として存在することになる。すなわち，まず第一はデータを DB 化できない。第二は DB 化までは何とかなるが，自ら使うのみで公開はできない。この 2 つの壁を取り除くためには，情報技術の開発だけではなく，それらの技術を活用するためのエンジニアリングが必要である。

1.2 何が必要なのか

多くの研究者は研究に必要な資料やデータをカードにしたり，図表にしたり，あるいはファイルフォルダーにまとめたりして整理している。DB 化とはこれらの資料やデータをコンピュータ上の仮想空間に移し替える作業に他ならない。言いかえれば，それはコンピュータにとって理解しやすい，処理しやすい，管理しやすい形態への変換作業である。このことは，その形態がどのようなものかを熟知していないと，DB 化に失敗する恐れが大きいことを意味しており，そのことが DB 化を難しいものにしている。したがって，その変換作業が研究者が日常的に行っている資料やデータの整理作業の延長として行えれば，DB 化は困難なものでなくなってくる。すなわち第一の壁は消滅する。

いまや DB の検索は，Web のページ上で行うのが主流となりつつあるが，そこでは多くの場合 DBMS が使用されている。したがって，利用者は DBMS に習熟している必要はないが，提供する側にはそれらの DBMS を熟知している技術者を必要とする。一方，ホームページ作成は，さまざまなアプリケーションソフトやツールが開発されており，容易な作業となっている。したがって，ホームページ作成と同じような手順で，DB をサーバ上に構築し，DBMS と同じような検索や表示が可能なツール群が用意されれば，DBMS を知らなくても，Web での DB 公開が比較的簡単に可能になる。第二の壁の消滅である。

1.3 BB-DB とは

ここで DBMS を利用することのメリット，デメリットを論じる余裕はないが，BB-DB はそれぞれの DB に対応した小規模なプログラム群から構成される各種ツールを用意し，DBMS を利用することなく同等の機能を実現しようとするものである。

BB-DB の最大の特徴は，**1R1H**，すなわち 1 record を 1 HTML file としてサーバ上に格納するところにある。このことによって従来の DBMS 以上の柔軟性と拡張性を実現している。

例えば，ある特定のレコードに関して例外的な処理（貝塚 DB の場合，図表や写真の項目は設定していないが，ある遺跡についてのみ，写真等を追加したいといった処理）を必要としても，その処理を行うページにリンクを張るだけでよい。そのリンクも，あるディレクトリに例外処理のページを格納しておけば，自動的に対象とする遺跡のページとリンクを張れるようなツールを貝塚 DB では用意した。

BB-DB は，現在開発中のシステムであり，その詳細を体系的に示すことはできないが，大まかな概念は図 1 のとおりである。ここでは DB 管理者，DB エンジニア，サーバエンジニアの 3 つの役割が必要になる。もちろん 1 人が全てを兼ねる場合もあろうが，通常は DB 管理者（＝ DB エンジニア）とサーバエンジニアという構成になる。そして，DB 管理者（＝ DB エンジニア）が公開したい DB を持っている研究者ということになる。いったんシステムが確立されれば，その運用に必要なのは DB 管理者（＝研究者）のみとなる。運用は全て Web 上の運用管理ページから行えるようになっており，特別な処理や新しい機能の追加等を除けば，DBMS の素人である研究者でも容易に運用していくことが可能である。

このようなことが可能になっているのは，BB-DB が simple is best, small is beautiful という基本哲学のもとでシステム開発を進めているからである。すなわち，処理プログラムは可能な限り単純に，そして小さな単位で行うようになっており，保守が容易で拡張性に富んだものとなっている。

2. BB-DB －その適用例－

今回われわれの提唱する BB-DB を貝塚 DB に適用した。以下，サーバでの具体的な処理手順について述べる。なお，貝塚 DB の URL は http://koko.soken.ac.jp/groups/kaizuka である。

2.1 サーバへの実装上の基本原則

ここでは DB をサーバへ実装のための基本原則について解説する。実装のための処理は，大きく DB 公開とシステム管理の 2 つの側面がある。DB 公開には，前処理，CGI によるプログラム処理，システム管理には，更新システム，ユーザ管理システム，また，開発過程においては，DB の内容を担当する DB エンジニアリングと DB をサーバで公開するサーバエンジニアリングの 2 つの仕事が発生することになる。

前処理と場合分け：公開するデータは，可能な限りあらかじめ html に変換する。しかし，一つひとつのファイルを管理していたのでは，大変なので，これらは公開データとし，直接編集することはない。管理すべきファイルは，サー

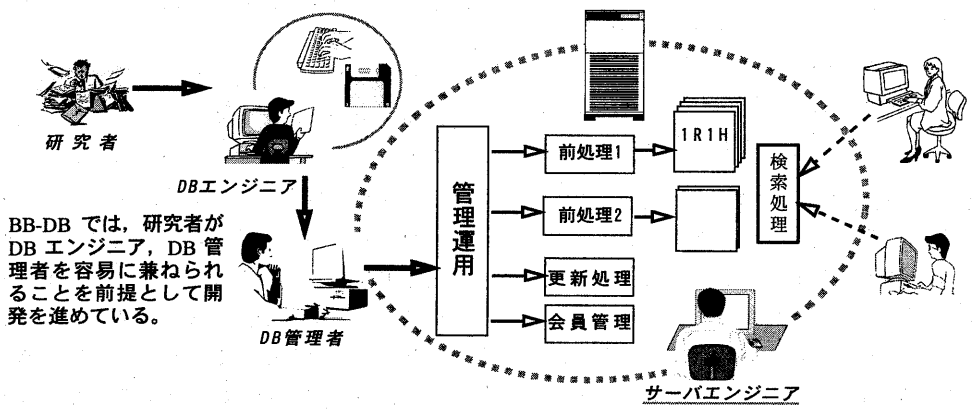

図 1 BB-DB システムの概念図

バにはおかれず，管理は個人パソコンの表ソフト等で管理更新を行う。サーバにはテキスト形式のファイルを転送し，そこからそれぞれのレコードの html ファイルを生成する。これを前処理と呼ぶ。

前処理では，公開に際して予測されるユーザの行動を考慮して，静的部分，動的部分に大きくわけ，静的部分については前処理で.html を出力し，動的部分については cgi を用意する。例えば，県別インデックス（47 県）のように数の固定されているものについては，静的処理として，前処理で ken.html を生成する。また，頻繁に検索が実施されると考えられるデータについては，あらかじめ前処理でテーブルを抽出しておき，そのテーブルを対象に検索を行う。さらに，対象によって，将来的に更新される部分と更新されない部分でプログラムを書き分ける。

プログラム開発：前処理の実行は大きな CPU 負荷を伴ったとしても，頻繁に行われる作業ではないので，できるだけシンプルに，それぞれのステップの入れ換えや書き換えが容易に行えるよう明示的に記述する。過度に高級な技術，あるいは処理のための処理といったメタ処理はできるだけ使わない。単純な技術の組合せを重視し，プログラムの可読性を高くし，プログラムはできるだけ小さい単位で作成する。また，運用中の機能の追加，機能別に独立したテスト等が行えるように，システム全体に影響を与えるようなプログラムの開発はしない。高速に，確実に，生のデータを直接アクセスすることで，管理効率のよいシステムを構築する。

ユーザインタフェース：クライアントの CPU やブラウザにパフォーマンスが依存しないように，サーバから転送するデータは html を基本とし，転送に負荷となるグラフィックスや java はできるだけ使わない。ユーザがそのまま html ファイルを保存した際，論文等に引用しやすいよう，それぞれ独立したレコードとして出力する。それぞれわかりやすいように，インデックス別，機能別にリンクを記述したトップページを用意する。

管理者インタフェース：管理者は，トップページの html ファイルの編集と利用者の管理，DB 全体の更新を行う。トップページは一般的な index.html であり，このトップページは，DB 管理者が関連情報をリンクしたり，おしらせ，利用マニュアルを追加できるように，単純な html で提供する。一方，利用者の管理，DB 全体の更新は，別のディレクトリに置かれたページから行う。http 標準実装の認証処理を行い，管理者以外はアクセスできない。このページには，管理作業別にリンクが施され，どういう手順で更新すればよいかは，プログラム実行中に明示され，作業に間違いのないよう確認メッセージを出す。

DB エンジニアとサーバエンジニア：DB エンジニアとサーバエンジニアの役割と目的について説明する。

DB エンジニアは，データシートのデザイン，データの入力，データの修正，データの表記一貫性の確保，またそのチェックプログラムの開発とデータチェックの実施，コンピュータで制約のある文字表現等の問題とその解決，原本と出典，１次データとの関連位置づけ，データシートの管理を行い，DB の目的，DB 利用方法，利用規定の明確化，ユーザとの情報交換等の作業を行う。

サーバエンジニアとは，データシートのデザイン及び対象 DB の特性をよく検討し，前処理の実施方法，検索の方法を検討する。また，プログラミングを行うだけでなく，データ出力のテストを行いつつ，対象つまり本稿における例では貝塚 DB の内容を十分に考慮した，集計方法，可視化の方法等を検討し，実装する。その上で，ユーザはどのような検索を望んでいるのか，どのような情報を必要とするか，どの程度の頻度及び範囲で更新を行うか，追加情報はど

のように取り扱うか，ユーザのデータ読み出しアクセスはどこまで許可するのか，ユーザのデータ書き換えを認めるか等を，DB エンジニアとサーバエンジニアの間であらかじめ話し合い，動的データ，静的データ，検索の範囲，データ更新の方法を決定する。さらに，品質を向上させるためには，利用者を増やすためにはどのようにすればよいかも検討し，ワークショップや研究発表等の計画も行う。

2.2 プログラムとツール群

前処理ツール：DB エンジニアからサーバエンジニアに引き渡されるファイルは表 1 にある 4 種で，それぞれのオリジナルは，DB エンジニアのコンピュータで管理され，サーバにはコピーが置かれ，それをもとに前処理を行う。したがって，DBMS では，それがクラックされた場合には，オリジナルファイルが危機に曝されるが，本システムにおいてはその心配はない。

表 2 の前処理プログラムにより，それぞれのファイルを生成するが，これら生成されたファイルは公開形式であって，最も重要なオリジナルではない。これらの更新プログラムは管理者ディレクトリに置かれ，管理者ページから WEB ブラウザを通して，実施される。どれがどの作業を行うかは，ページに記述してあるので，使用方法を迷うことはない。

表 1　前処理用ファイル

ファイル名	数	サイズ	内　　容
kai.db	5678	1.4MB	貝塚メインファイル
bib.db	2664	0.7MB	文献メインファイル
table.euc	146	2.6KB	時代遺跡遺構コードテーブル
kencode.tab	47	0.6KB	県コードテーブル

表 2　前処理用プログラム

プログラム名	行数	ソース	生 成 フ ァ イ ル
makekai.pl	200	kai.db	遺跡レコードファイル KZxxxxxx.html
makebib.pl	104	bib.db	文献レコードファイル KBxxxxxx.html
makeiko.pl	146	kai.db	遺構別インデックスファイル iko.html
makeiseki.pl	144	kai.db	遺跡別インデックスファイル iseki.html
makejidai.pl	143	kai.db	時代別インデックスファイル jidai.html
makeken.pl	134	kai.db	県別インデックスファイル ken.html
search_format.pl	119	kai.db	遺物検索テーブル search_item.tab

実行処理プログラム：ユーザが利用する CGI は，遺物検索のみである。貝塚 DB では，該当遺物の一覧は「完全一致」「部分一致」「かつ」「あるいは」の組合せにより，固定ではないため，その一覧生成に CGI を用いた。動的出力なので同時に該当件数の総数計算，県別集計，県別頻度色分け地図の出力を実装した。そのため，414 ステップスと他のプログラムよりも少々長いが，それでも 9KB 程度である。

- kz_search_f.pl:414(検索プログラム：検索し，結果を html で出力する。出力には，各レコード.html へのリンクが出力される。また，外部プログラム map をアクセスし，遺物頻度地図を出力する。)

公開アクセス管理ツール：利用は，検索，遺跡別一覧，遺構別一覧，県別一覧は，ユーザ制限なしでアクセスできるようにし，それぞれの遺跡レコードあるいは文献レコードのアクセスにはユーザ制限をもうけた。ユーザのアクセス制限は http 標準実装の認証を利用した。これらレコードにアクセスするためには，会員申請をする。ユーザが会員申請の受付フォームを使って会員申請を行うと，フォームの内容を regist.dat にアペンドする。

また，同じ内容を DB 管理者に，メールで自動的に送り，会員申込があったことを知らせる。データ更新・訂正報告フォームは，DB を利用している会員がデータの訂正を見つけた場合，それを報告するフォームで，同様に報告が行われると，その内容は，DB 管理者にメールで報告される。この場合，公開 DB のどこにも報告の内容は記録されない。会員アクセス許可あるいは削除，会員名簿更新ツールは DB 管理者のみが使えるプログラムで，会員アクセス許可プログラムは，許可実行とともに自動的に ID とパスワードが送られる。プログラムと生成ファイルは表 3,4 の

ようになる。

導入及び更新手順：サーバが立ち上がったら次の手順でDBの導入を行う。

1. 基本データの転送
   (kai.db,bib.db,ken.tab,table.euc)
2. 前処理の実施
   ・貝塚アップデータ(前処理/kai_update.pl:115)
   ・索引アップデータ(前処理/idx_update.pl:143)
   ・文献データアップデータ
   (前処理/bib_update.pl:105)
3. 検索システムの導入(kzsearch_f:414)
4. 外部プログラムの導入(jgrep,map)
5. アクセスシステムの導入
6. テスト

表3　会員管理プログラム

プログラム	行数	内　　容
kzregist.pl	388	会員申請受付フォーム
kzmember.pl	140	会員名簿公開プログラム
kztouroku.pl	329	会員アクセス許可ツール
kz_delete.pl	241	会員アクセス削除ツール
kz_correct.pl	219	会員名簿更新ツール
kzrepo.pl	313	データ更新・訂正報告フォーム ニュース・メイル転送システム

表4　会員管理用ファイル

ファイル名	内　　容
htacess	アクセスできるファイル，ディレクトリ，グループファイル，パスワードファイルのパスを記述。
group	貝塚グループでこのファイルに会員 ID が記述されてはじめて，レコードにアクセスできる。
passwd	各会員の encrypt パスワードが格納されている。
regist.dat	会員申請受付フォームから送られてきた内容がアペンドされるファイル。氏名，所属，電子メールアドレス，連絡先，自己紹介などが収集される。
id2email.dat	会員 ID と電子メールの関係テーブル

2.3　検索実行速度のテスト

DBMS を利用するとバイナリツリーあるいはハッシュ表等の検索テーブルを作成し，検索実行速度をあげる仕組みがあり，これを導入すると確実に飛躍的な改善が可能となる。しかしながら，その分プログラムの構造は複雑になり，処理のための処理，デバック，追加機能の実装にコストがかかる。そこで，本研究におけるシステムでは，どの程度の時間がかかっているかを計算し，プレーンテキストの検索が実際的であるかどうかを検討した。

具体的には，登録されている 5678(1382KB) レコードに対し，遺物とレコード ID だけを抽出したファイル，search_item.tab を前処理として作成した。これは 4424(839KB)レコードになった。この 4424 レコードを対象に検索した場合の CPU 時間を計算する。検索に利用したプログラムは jgrep(GNUgrep version2.0+multi-byte extension1.04)で，CGI で外部呼び出しをしているプログラムと同じプログラムである。計測に利用したプログラムは，Gnutime version1.7 で，実施した CPU は，Intel-MMX133MHz。OS は，Linux kernel2.0.34。総メモリは 96MB でいずれの実行環境においても，Swap は検出されなかった。出力は結果のテキスト量に依存するので，出力はいずれも/dev/null に送った。

それぞれ 50 回試行し，ユーザ CPU 時間，システム CPU 時間，実 CPU 時間の 3 種類の平均時間計算(単位は秒)を求めた。また，該当件数，部分一致，完全一致，「かつ，あるいは」等の条件によって変動が認められるか，サイズ 10 倍の疑似ファイルの場合はついても検討した。表 5 はその計測結果の一覧である。

「あるいは」以外いずれの条件においても，実時間は，0.12-0.18 秒程度であった。「あるいは」は jgrep を 2 回呼び出しているので，やや多いが，それでも，0.2 秒程度である。search_item.tab を 10 回コピーしたファイルを作成し，それに対し，CPU 時間を計算したところ，単純に CPU 時間は 10 倍となった。

search_item.tab を 10 回コピーしたファイルにおける結果でわかるように，5 万件のデータを仮定した場合，それに要する時間は 1.2 秒程度である。この速度は，html のデータをクライアントのリクエストにより，通信転送する時間を考慮した場合，この検索時間よりページの転送時間の方が大きくなるものと思われる。

表5　処理速度のテスト

U = user, S = system, R = real

部分一致「シジミ」該当件数 1877				
U	Min=0.100	Mean=0.136	Max=0.170	SD=0.015
S	Min=0.020	Mean=0.044	Max=0.080	SD=0.015
R	Min=0.170	Mean=0.173	Max=0.190	SD=0.006
完全一致「シジミ」該当件数 143				
U	Min=0.040	Mean=0.069	Max=0.090	SD=0.012
S	Min=0.000	Mean=0.023	Max=0.070	SD=0.013
R	Min=0.080	Mean=0.095	Max=0.280	SD=0.028
完全一致「ウシ」該当件数 236				
U	Min=0.070	Mean=0.093	Max=0.110	SD=0.012
S	Min=0.010	Mean=0.029	Max=0.050	SD=0.012
R	Min=0.120	Mean=0.122	Max=0.180	SD=0.010
部分一致「ウシ」該当件数 424				
U	Min=0.070	Mean=0.109	Max=0.140	SD=0.017
S	Min=0.000	Mean=0.038	Max=0.080	SD=0.017
R	Min=0.140	Mean=0.146	Max=0.360	SD=0.031
部分一致「ウシ」該当件数 4240 (search_item.tab の 10 倍ファイル)				
U	Min=0.900	Mean=0.977	Max=1.080	SD=0.042
S	Min=0.150	Mean=0.248	Max=0.330	SD=0.042
R	Min=1.210	Mean=1.223	Max=1.280	SD=0.011
完全一致「ウシ」かつ「ウマ」該当件数 149				
U	Min=0.060	Mean=0.096	Max=0.130	SD=0.016
S	Min=0.000	Mean=0.030	Max=0.060	SD=0.014
R	Min=0.170	Mean=0.178	Max=0.250	SD=0.011
完全一致「ウシ」あるいは「ウマ」該当件数 587				
U	Min=0.140	Mean=0.180	Max=0.210	SD=0.016
S	Min=0.010	Mean=0.041	Max=0.080	SD=0.017
R	Min=0.210	Mean=0.219	Max=0.230	SD=0.005

133MHz の CPU 速度でこの程度であるので，最近の 1.3GHz となると，さらに問題とならない速さといえよう。現在の貝塚 DB は，5 千件のデータであるが，現在作成中の遺跡 DB では，10 万件に及ぶものとされている。その場合でも，CPU 時間はかなり小さいものといえよう。

2.4 BB-DB の利点

ここでは，DB のレコード表現に html を採用した利点，外部プログラムとの組合せによる利点，通信プロトコルとして http を用いた利点について述べる。

html の利点：まず，出力形式を html とすることで，クライアントの画面サイズを気にすることなく，レコード出力を行うことができる。また，前処理出力時に，外部 html の参照を指定することで，レコードに外部の情報を付加することもできる。それらはとくにテキストである必要はなく，リンクに用いられるものであれば，何でもよい。実際に関連情報として，PhotoCD のファイルをリンクしているページもある。それぞれのレコードは html ファイルとして独立しており，外部のサーバ上にある，html からも参照することができる。

静的な html でなく，DBMS によってその都度出力するシステムではこういうことはできない。さらに，他のサーバから参照してもらいたいときには，タグを使い，サーチエンジンロボットに明示的に覚えさせれば，外部検索エンジンからも利用できる。

外部プログラムとの組合せによる利点：まず，UNIX にインストールされている，あるいはインストール可能なプログラムとの連携が簡単にとれることがあげられよう。例えば，貝塚 DB では，貝塚の遺物検索結果の県別頻度表示の可視化手法として MapOfJapan[*1] のライブラリー式を利用している。また，http 標準実装の認証を利用したユーザ管理システムを保有しており，同時にユーザ管理ファイルの電子メールアドレスを使って，sendmail により ML を構成したり，ニュースを一括送信したりすることができる。これらは，特に UNIX の技術としては新しいものでも特別なものでもないが，一般的な DBMS だけを使っていたのではこのようなことはできない。いずれにしても，DBMS だけ

*1 MapOfJapan (http://aoki2.si.gunma-u.ac.jp/PseudoFTP/UNIX/MapOfJapan/)は青木繁伸氏（群馬大学社会情報学部）作成の日本地図描画システムである。C プログラムソースコード及びデータ表現のマニュアルが添付されているので，データの塗り分けの方法，矩形の表現方法等の変更が可能である。一番興味深いのは，プログラムもさることながら，作者が地図データの更新を懸命に行っていることである。作者の努力に対して感謝します。

を操作するだけでは，インターネットの利点を十分にいかすことはできない。

http の利点：まず，プロトコルが広く使われている http であり，レンタルサーバを利用する場合でも，それがどのようなものであれ，http であるはずなので，DB は必ず公開できる。http はファイル名をアクセスするプロトコルであり，レコードのアクセススピードは即ファイルシステムのファイルアクセススピードである。つまり，DBMS 起動による CPU 負荷，そのための余計なメモリ負荷は皆無ということになる。また，アクセス許可はディレクトリあるいはファイル毎で可能で，それに用いられる認証システムは，http 標準のものが用いられるので，あらたに認証システムを用意する必要はない。さらに，http 標準のアクセスログファイルにより，ユーザがどのファイルを何回アクセスしたかを知ることができる。現在の貝塚 DB では，検索キーワードのログの集計も実装されている。

3. 課題と将来構想

すでに述べたように，BB-DB はまだ開発途上のシステムであり，今後検討していかなければならない課題も少なくない。具体的にいくつか挙げてみる。

更新頻度の高い DB：貝塚 DB は，新規レコードの一括追加や誤り訂正以外には，日常的な更新はほとんどない。学術研究用 DB には，この種のものが多いが，一方，頻繁な更新を必要とする DB も少なくない。このような DB に対しても BB-DB が対応可能かどうかについては，まだ十分に検討していない。今後具体的な DB を選んで，必要なツール等の開発を進めていくことを予定している。

Quality Control：DB の内容に関する品質管理は，ほとんどの場合作成者に任せられており，実際の DB 構築においてその負担は大きいものがある。コンピュータを使ってどのような品質管理が可能かは，対象とする DB に依存する場合が多いが，標準的な品質管理のためのツール群を開発することは十分に可能であると考えている。

マニュアル整備とツール群の体系化：現在 BB-DB を適用しているのは，貝塚 DB と小松左京コーパス（SF 作家小松左京氏の全作品のフルテキスト DB）の 2 つである。後者の URL は http://castelj.soken.ac.jp/groups/komatsu で，すでに一般に公開している。今後他の DB についても適用していく予定で，そこにおいても多くのツールが開発されると想定される。これらのツール群の体系化と，BB-DB を利用したい研究者向けのマニュアル作成が緊急の課題としてある。

CC-DB 構想：現在その実現を目指して検討を進めているのが，CC-DB 構想である。CC-DB とは Collaborative Creation of Database のことである。これまでの DB 作りは，1 人または少人数の研究者がこつこつと地味な作業を続けるというのが一般的であったが，これを 100 人あるいは 1000 人が Web 上での共同作業を通じて，大規模 DB を構築するという試みである。そのためには体系的な作業手順，参加者をマネージメントするための仕組み，データの品質管理等についての研究開発が必要である。

おわりに

BB-DB は，まだ生まれたばかりのシステムである。今後どのように発展していくのか，あるいは意図したような汎用性，拡張性，柔軟性を保持できるのか，未確定な要素が多々ある。今しばらく開発を続け，CC-DB を含めた体系化ができた時点で，改めて報告をしたい。

最後になるが，貝塚 DB について簡単に説明する。これは日本全国で発見されている遺跡のうち，動物遺存体を出土している遺跡を集め DB 化したもので，レコード数は約 5,600 である。この DB 作成に協力してくれた多くの方々に深く感謝する。